Automating Malware Deobfuscation with Binary Ninja

Recon 2024

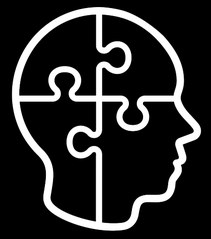
Download contents: https://github.com/Invoke-RE/workshops

Joshua Reynolds

Founder, Invoke RE

- Over ten years of security-related experience working for industry leading companies
- Spoken at RSA, DEF CON and Virus Bulletin on ransomware and malicious document analysis
- Co-developed malware analysis course taught at Southern Alberta Institute of Technology
- @jershmagersh / @InvokeReversing
- info@invokere.com

Obfuscation

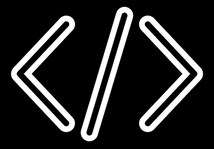


Makes malware difficult to understand and analyze

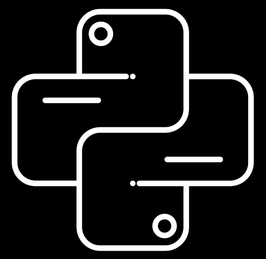
Hides the true purpose and behaviour of the malware

Binary Ninja

Powerhouse reverse engineering suite

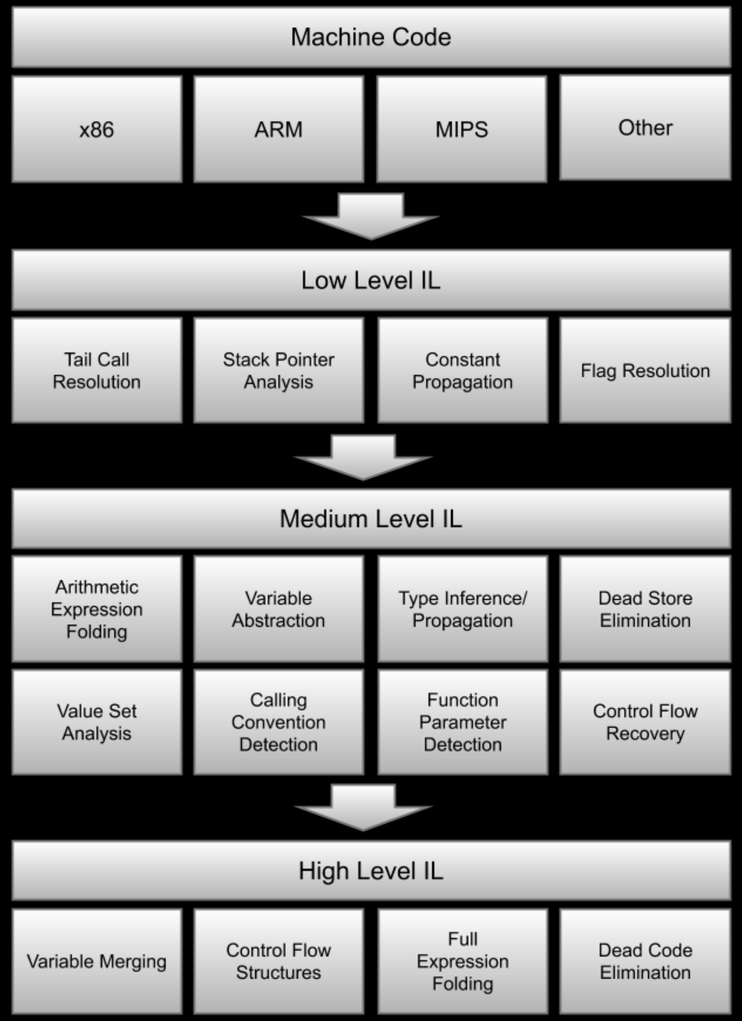


Intermediate languages (BNILs)



Robust Python API

Binary Ninja BNILs



Source: Binary Ninja User Documentation, https://docs.binary.ninja/dev/bnil-overview.html

Warning: Malware!

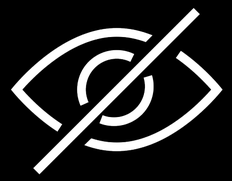
Real-world malware samples that may trigger antivirus. Please handle with care.

HLIL and Scripting with Binary Ninja

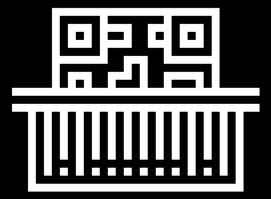
Unpacking Qakbot

Packers

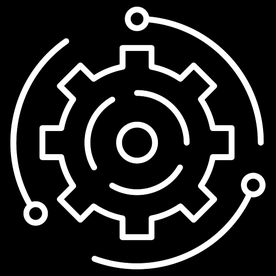
Broad terms to describe protecting original binary



Original binary not recognizable on disk

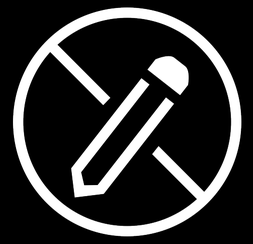


Typically use encryption, compression and encoding

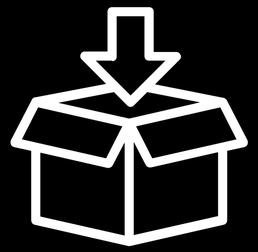


Stub generation and packing is typically automated

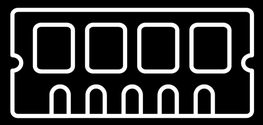
Packers Contd.



Do not modify original code

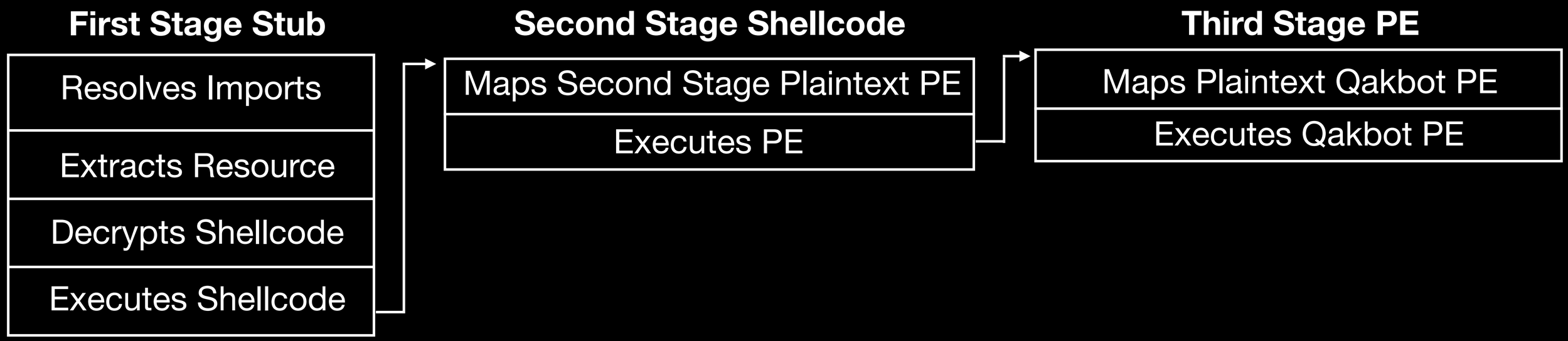


Crypters, packers, protectors

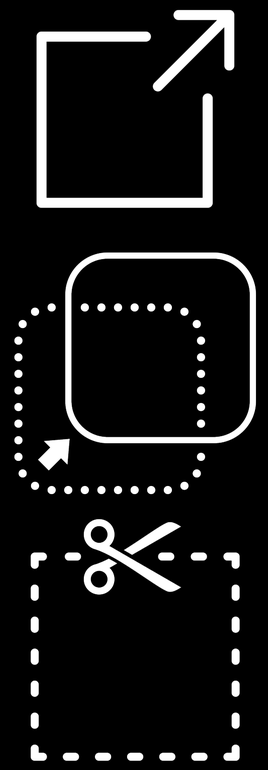


Execute binary in memory

Unpacking Process



Plan of Attack



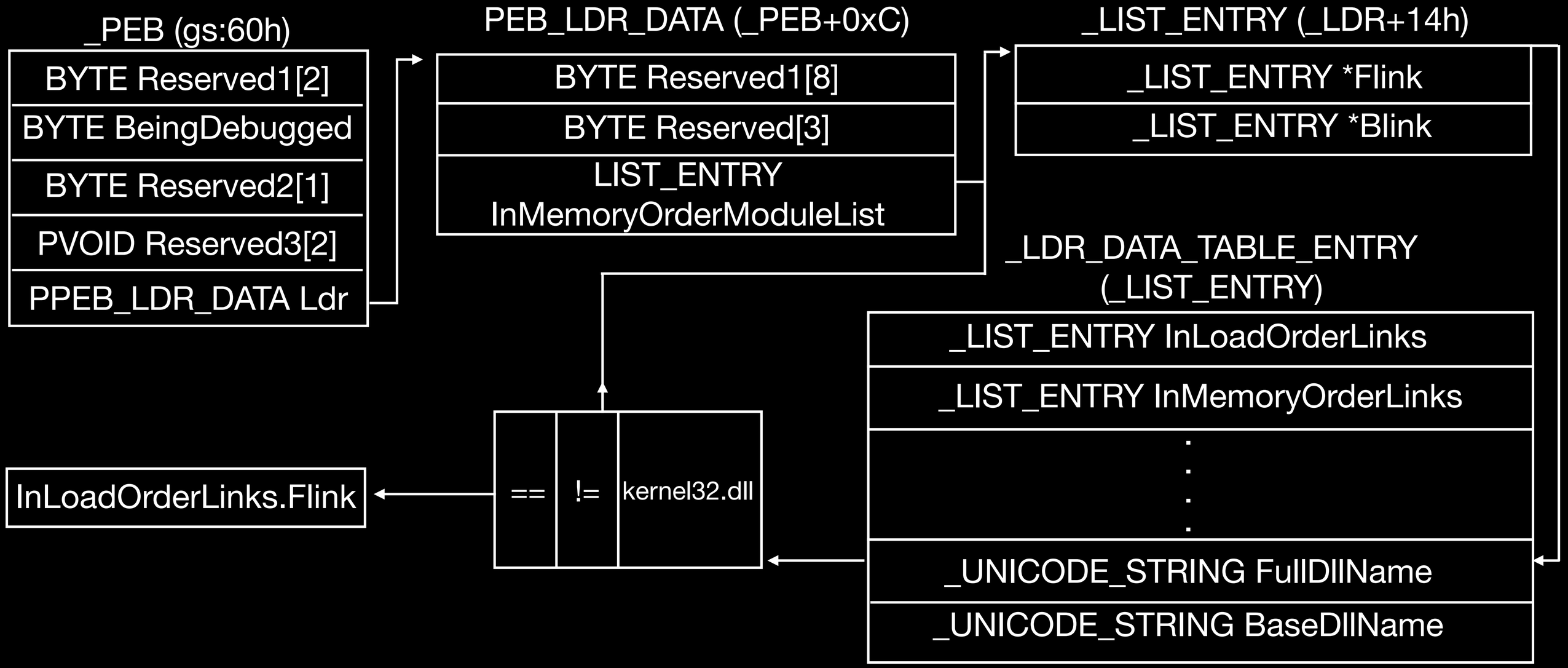
Extract needed info from stub using Binary Ninja

Extract resource (pefile) and decrypt it using info

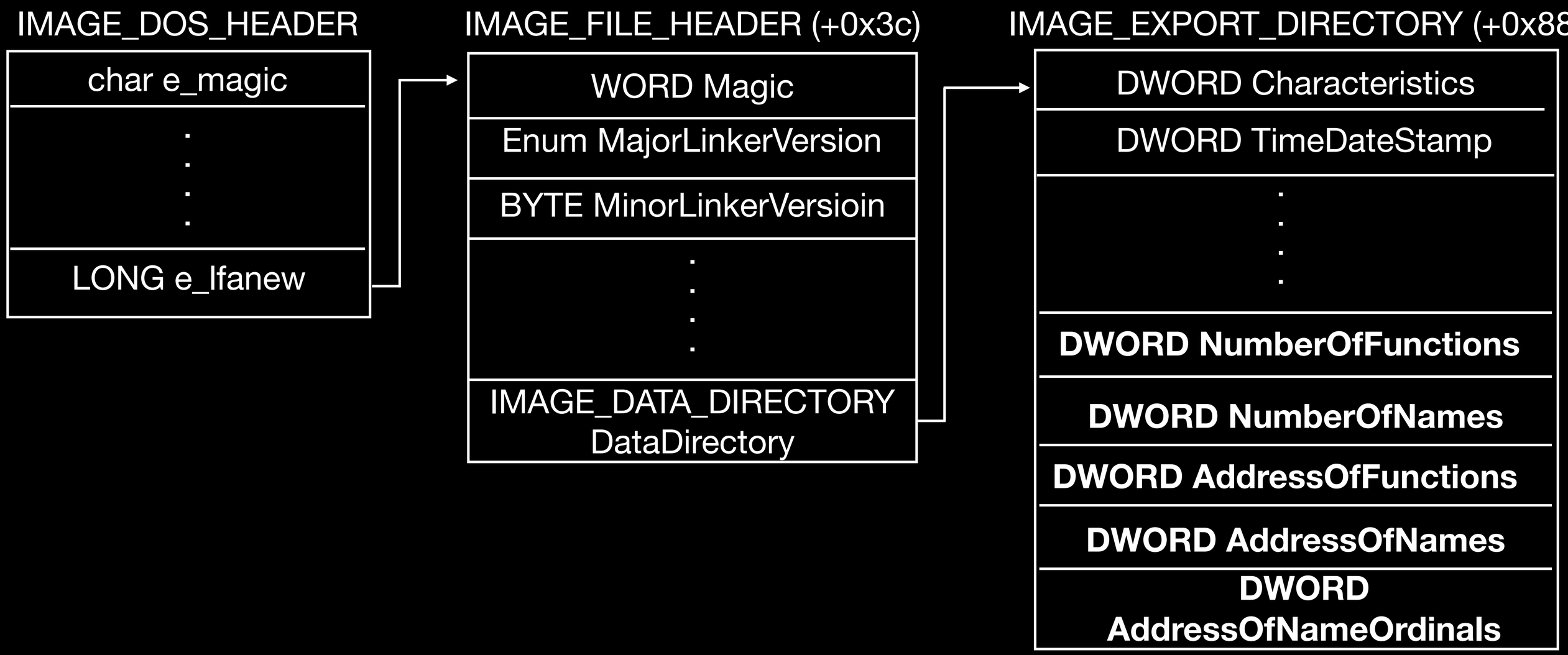
Carve plaintext PEs from shellcode (binary refinery)

Dynamic Function Resolution

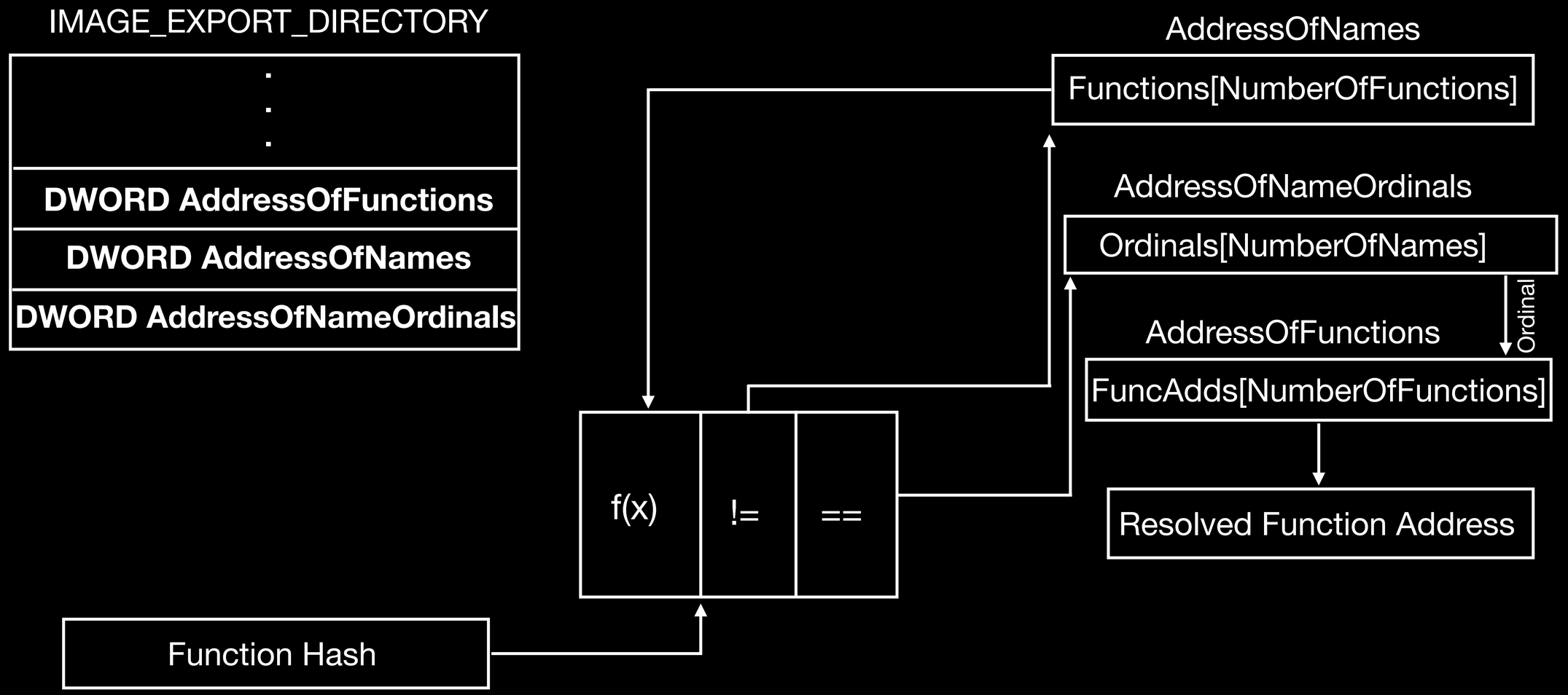
Dynamic Module Resolution

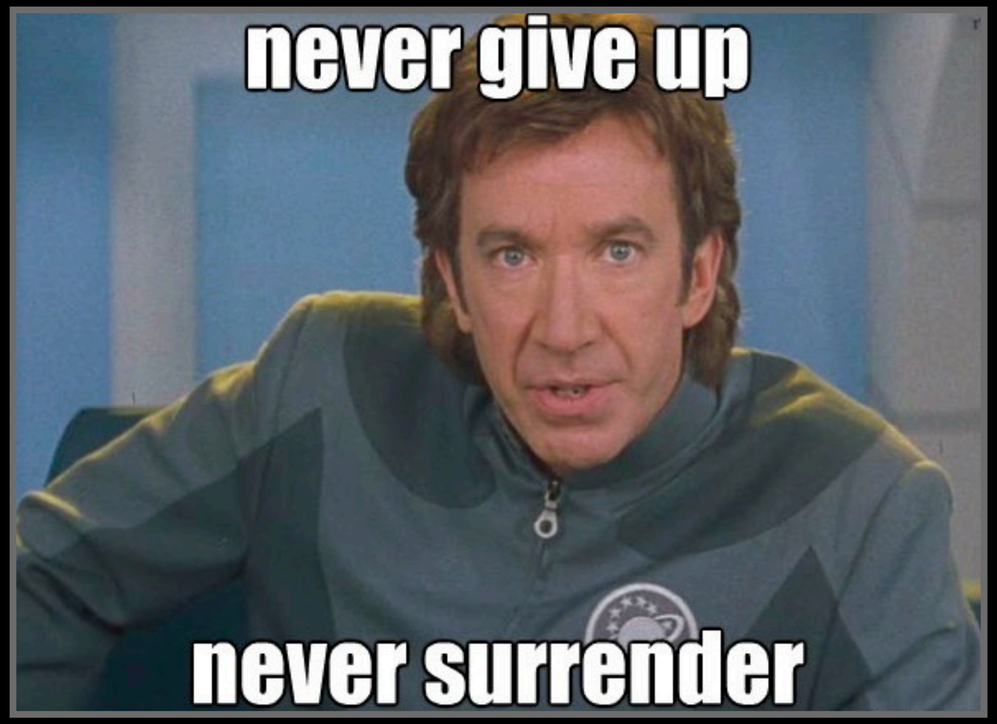


Dynamic Function Resolution



Dynamic Function Resolution





Contact

@jershmagersh
@InvokeReversing
info@invokere.com
invokere.com

Questions?

./RE