In Your Mems

Windows AV Evasion using In Memory Techniques

iDigitalFlame 2016

~$ whoami

iDigitalFlame

- Penetration Tester
- Digital Forensics Analyst

essextec
GETTING IT DONE TOGETHER

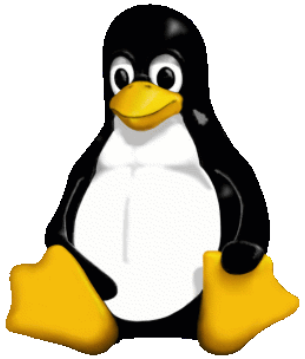
- Programmer
- Security Researcher
- Raspberry Pi Enthusiast

- Enjoys
  - Hacking Video Games
  - Popping Boxes
  - Sending Phishing Emails
- Python is Life

Overview

- The need to bypass Antivirus
- Ways Antivirus catches malware
- Current Antivirus bypasses
- Windows Memory Injection
- Windows API Methods
- AV Trust
- Antivirus Bypass
- Detection / Prevention
- Code & References

Antivirus



JULY 22 AAPL: 98.66 -0.77

Stagefright-style vulnerability discovered in OS X and iOS, update for protection

Apple patches zero-day vulnerabilities in Safari and OS X

Antivirus (cont.)

April 29, 2016

Incidents of Ransomware on the Rise

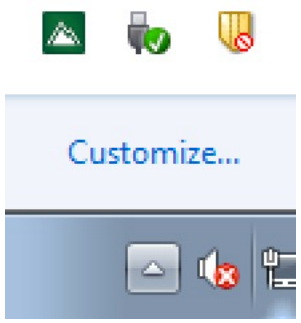
Protect Yourself and Your Organization

The need for AV bypass

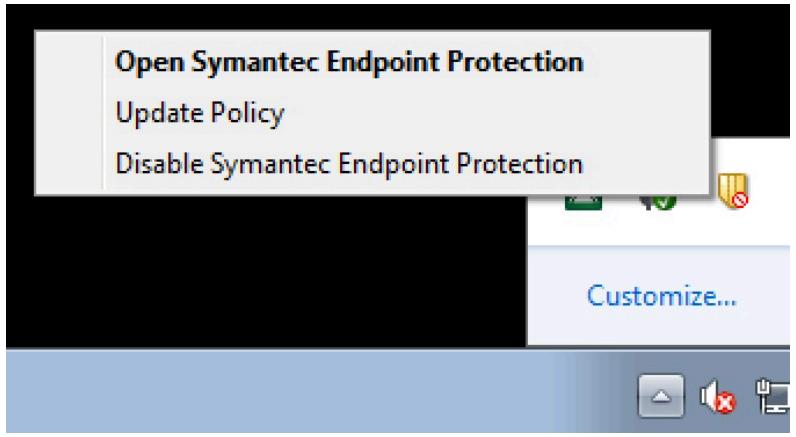
- Who uses it
  - Penetration Testers
  - Red Teams
  - “The Bad Guys”
- Why
  - Saves frustration
  - Successful delivery of payloads
  - Sometimes better then turning off AV

The need for AV bypass (cont.)

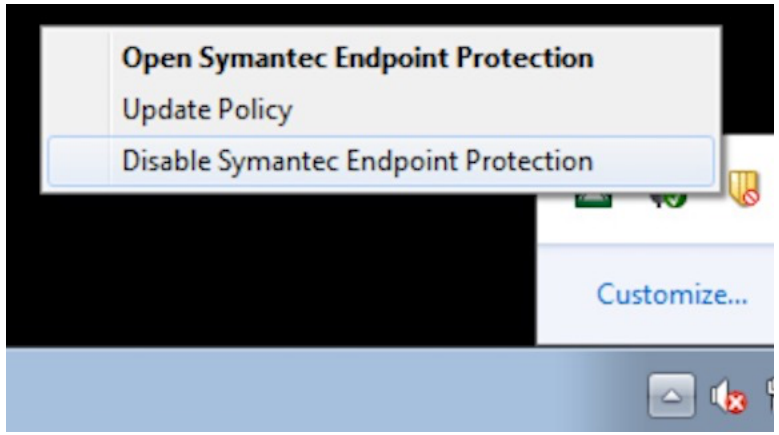
1



2

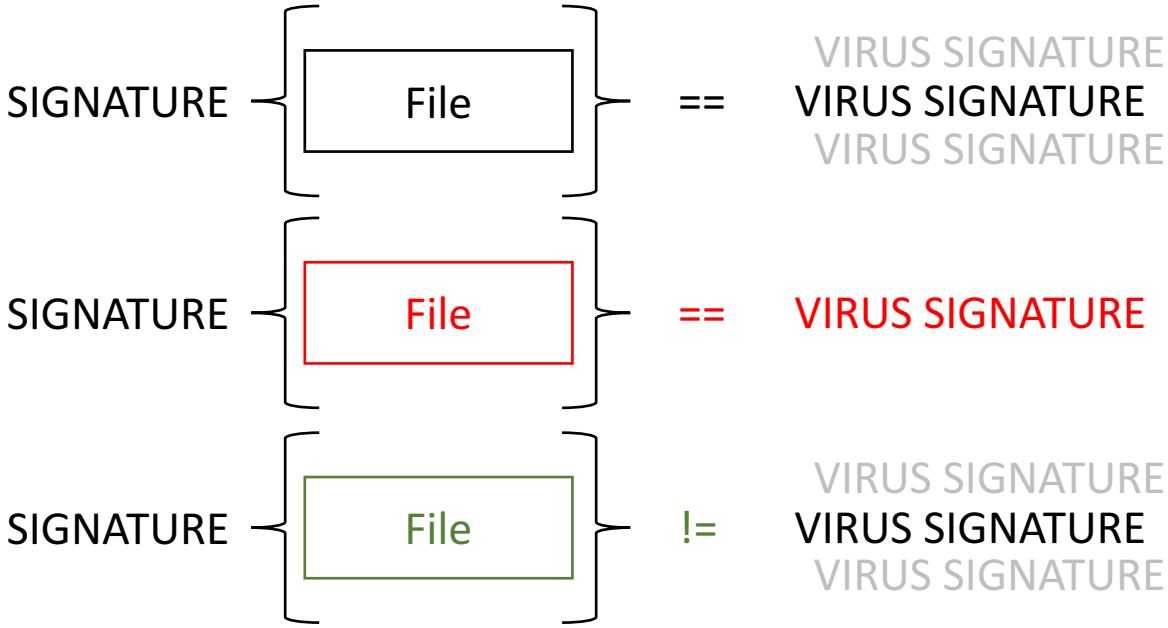


3



Getting Caught by Antivirus

- Antivirus works on signature detection
  - Signature === “Fingerprint” of malware
  - Hashes
  - File content



Getting Caught by Antivirus (cont.)

- Example of AV signature test file
  - EICAR test file

"X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*"

Getting Caught by Antivirus (cont.)

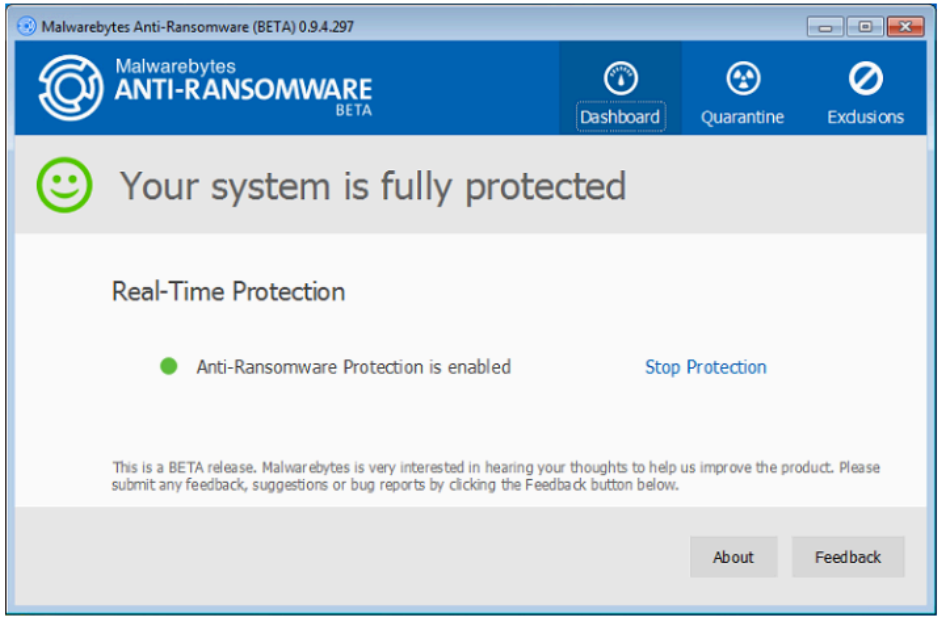
- Scans file system for malware
- Real-time scanning
  - Scans files as handles are generated
  - Intercepts malware being read or executed
- Not as reliable
- Only detects malware with known signatures

Getting Caught by Antivirus (cont.)

- Heuristic detection
  - “Actions” of processes
  - Network Connections
  - Resources Used
  - Uses “Threat Intelligence”
- Better method of detection
- Faster response to emerging threats
- Detects non ‘listed’ malware

Getting Caught by Antivirus (cont.)

- Ransomware detection
  - http://pub.idfla.me/links/iym1



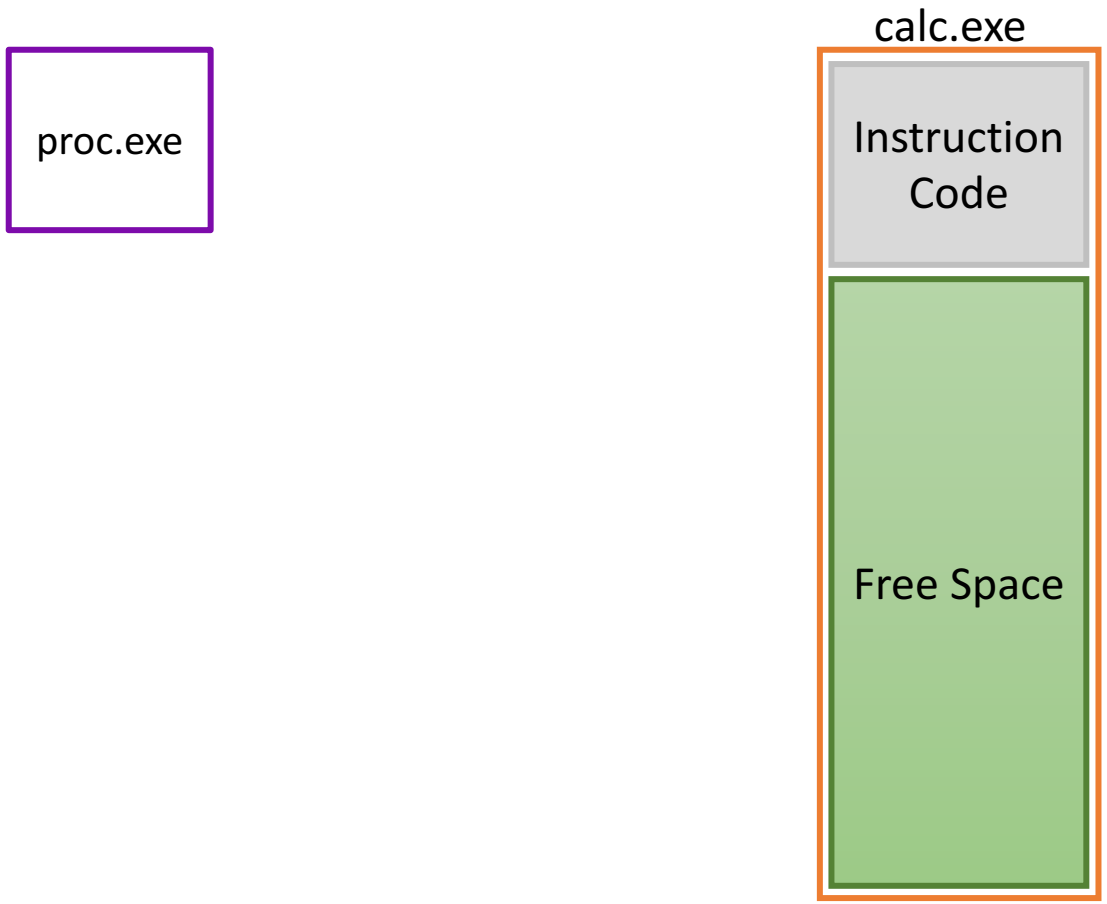
Ways of bypassing Antivirus

- Encoding / Encryption
  - Changes resulting hash
  - Same payload
- Loaders
  - Clean file
  - Downloads Malware when run
- Custom Written
  - Undetected until mainstream
- Veil Framework
  - Framework to pack malware
  - Many payloads available
  - Payloads in C, Go, PowerShell and others
  - Uses Memory Injection
  - Creates undetectable executables

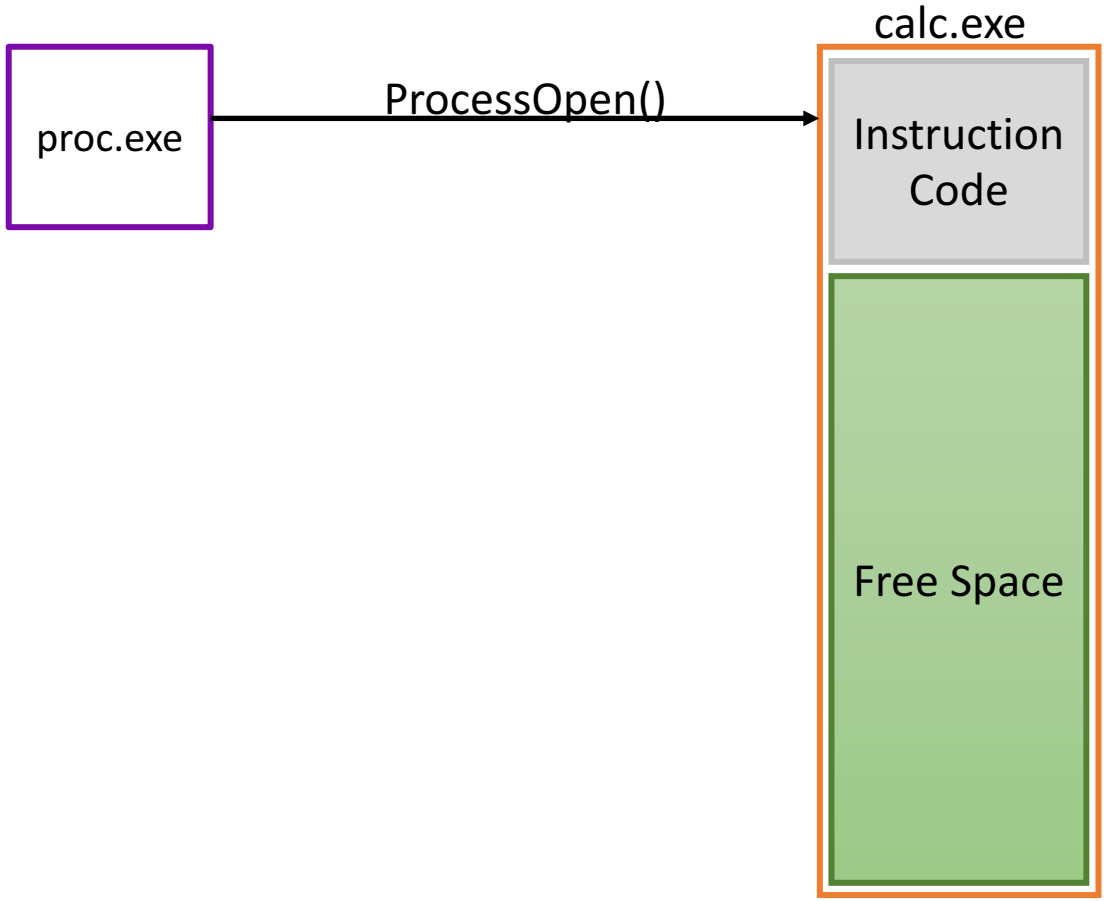
What is Memory Injection?

- Procedure of adding custom instructions to a process
- Not always Malicious
- Used by legitimate programs
  - PC Game Mods
  - Extensions to Windows Programs
  - Debuggers
- Custom instructions run in target process
  - Share same Process ID

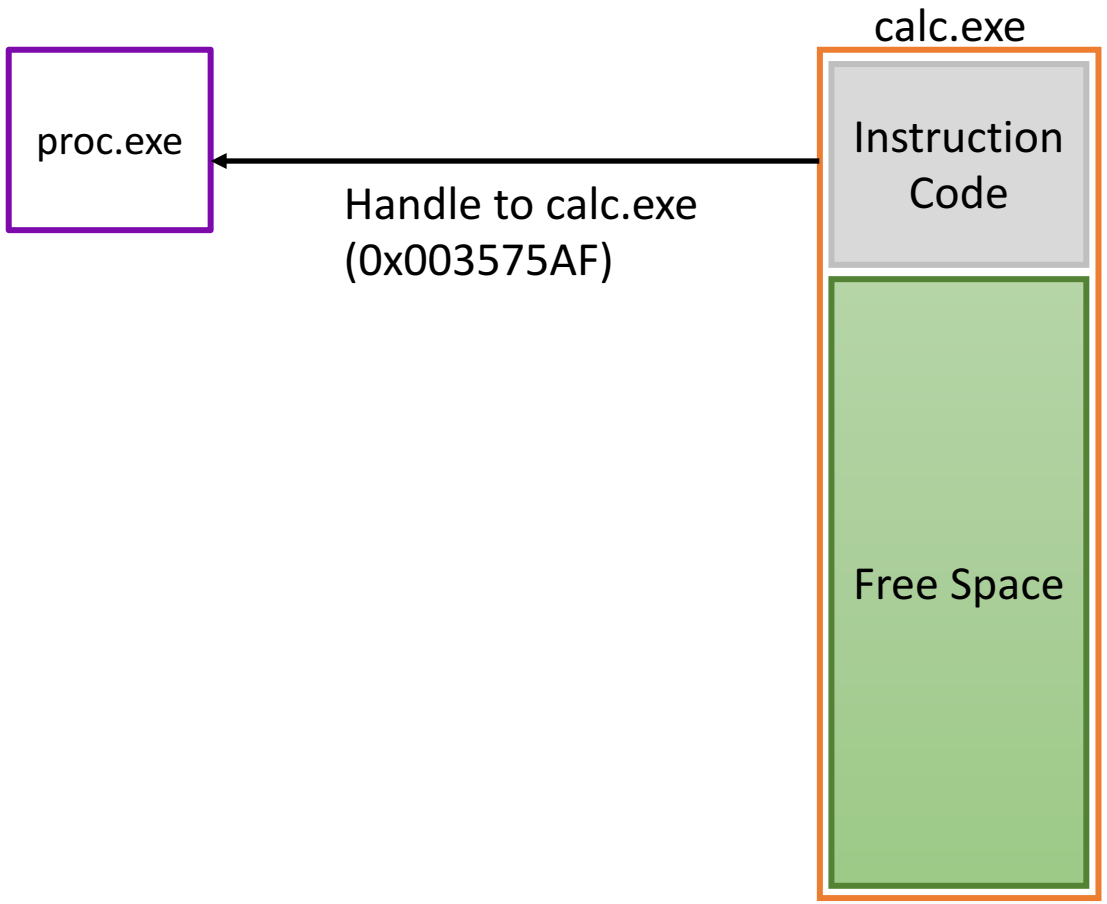
How memory Injection Works



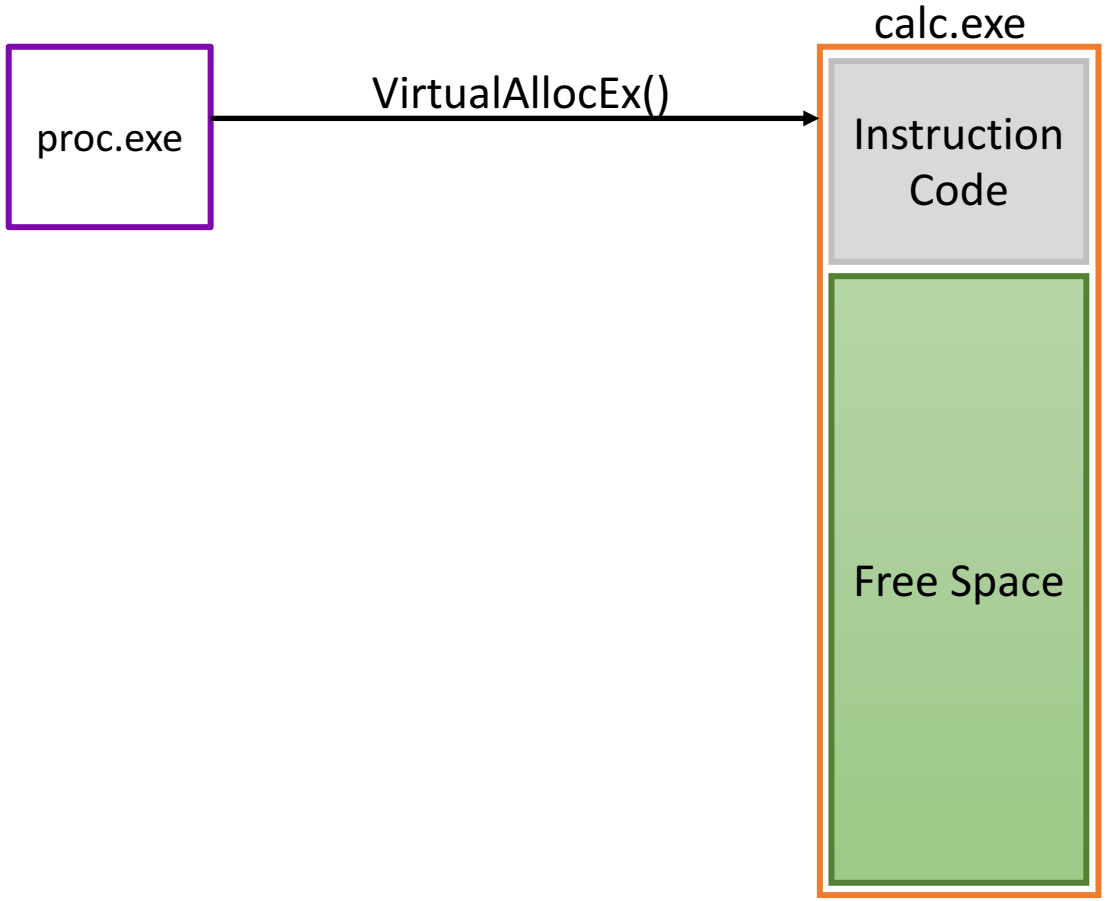
How memory Injection Works



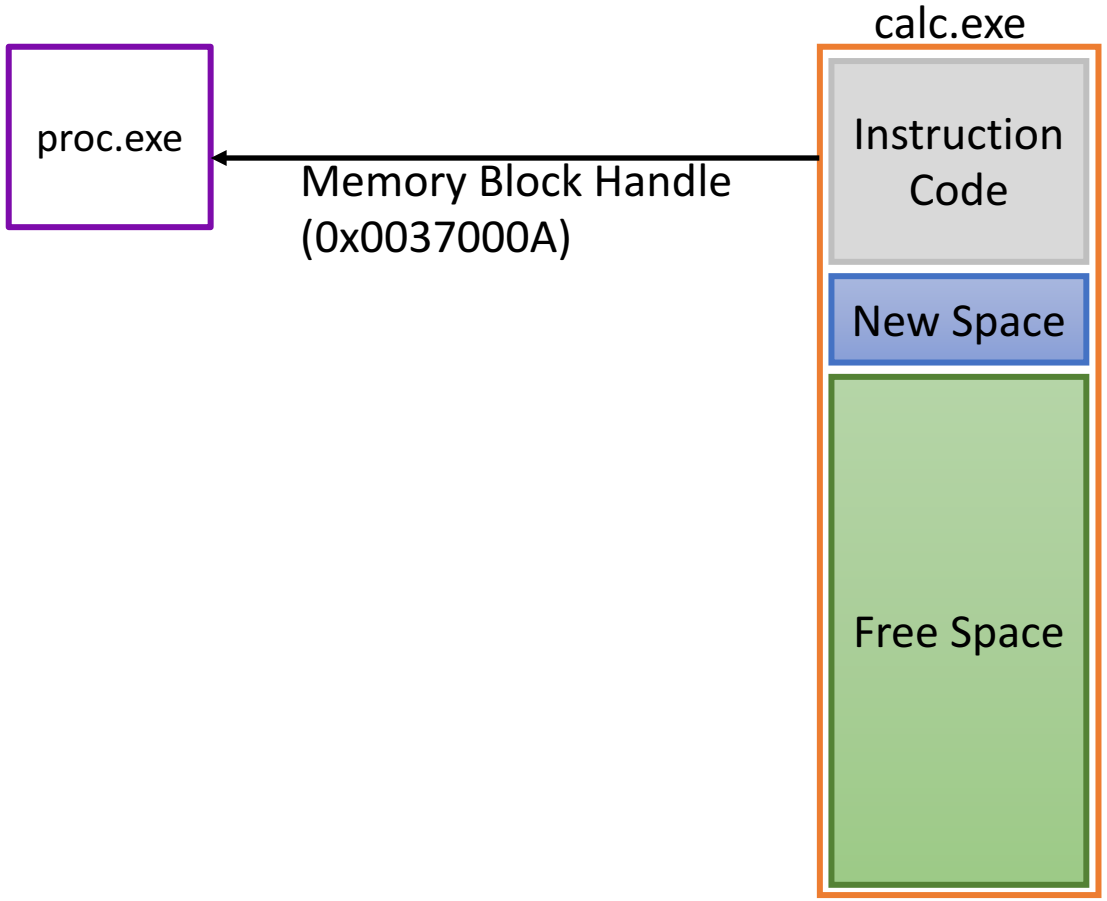
How memory Injection Works



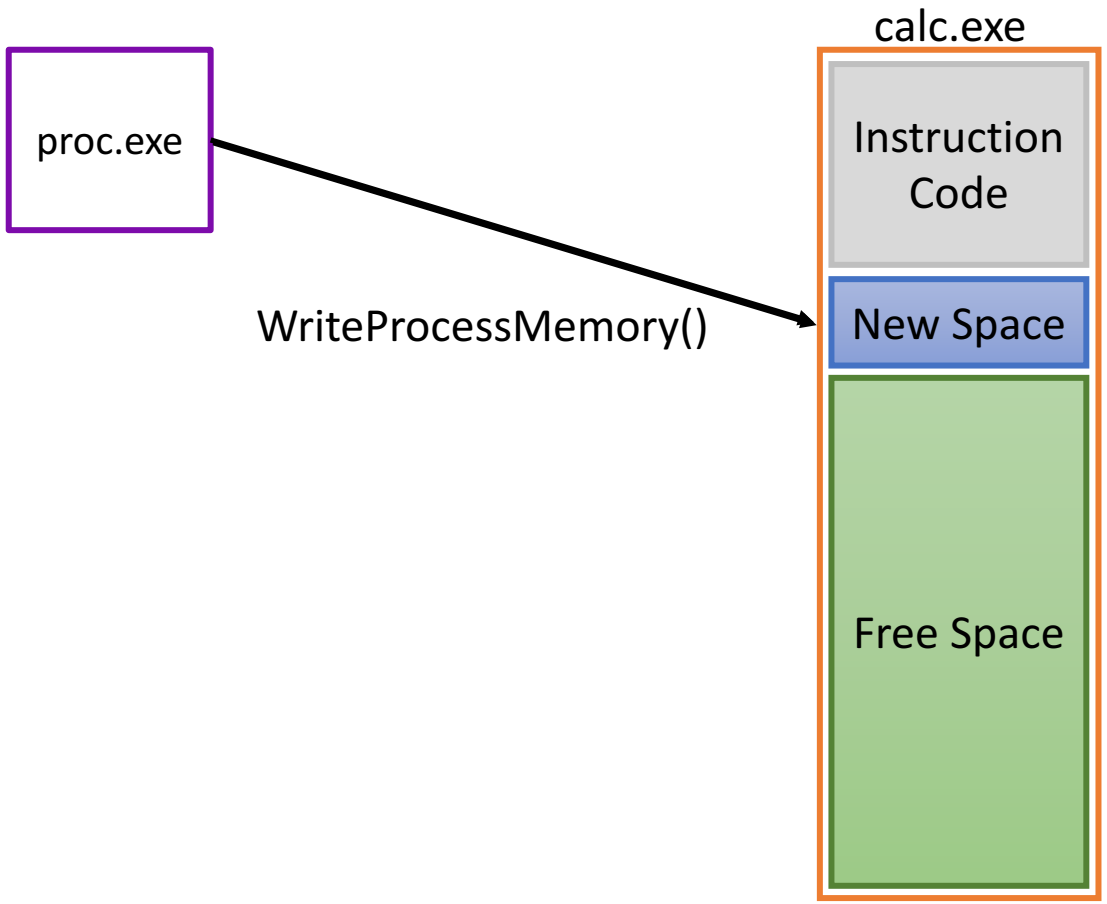
How memory Injection Works



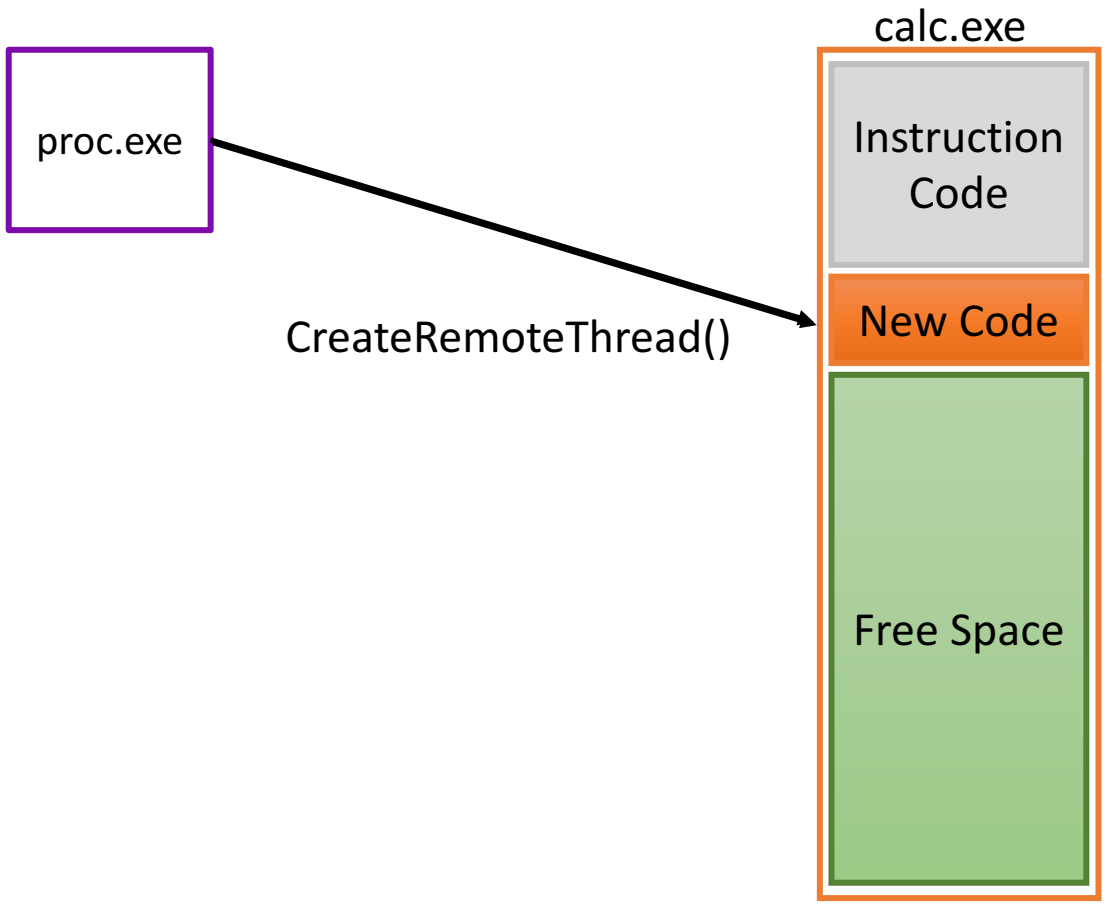
How memory Injection Works



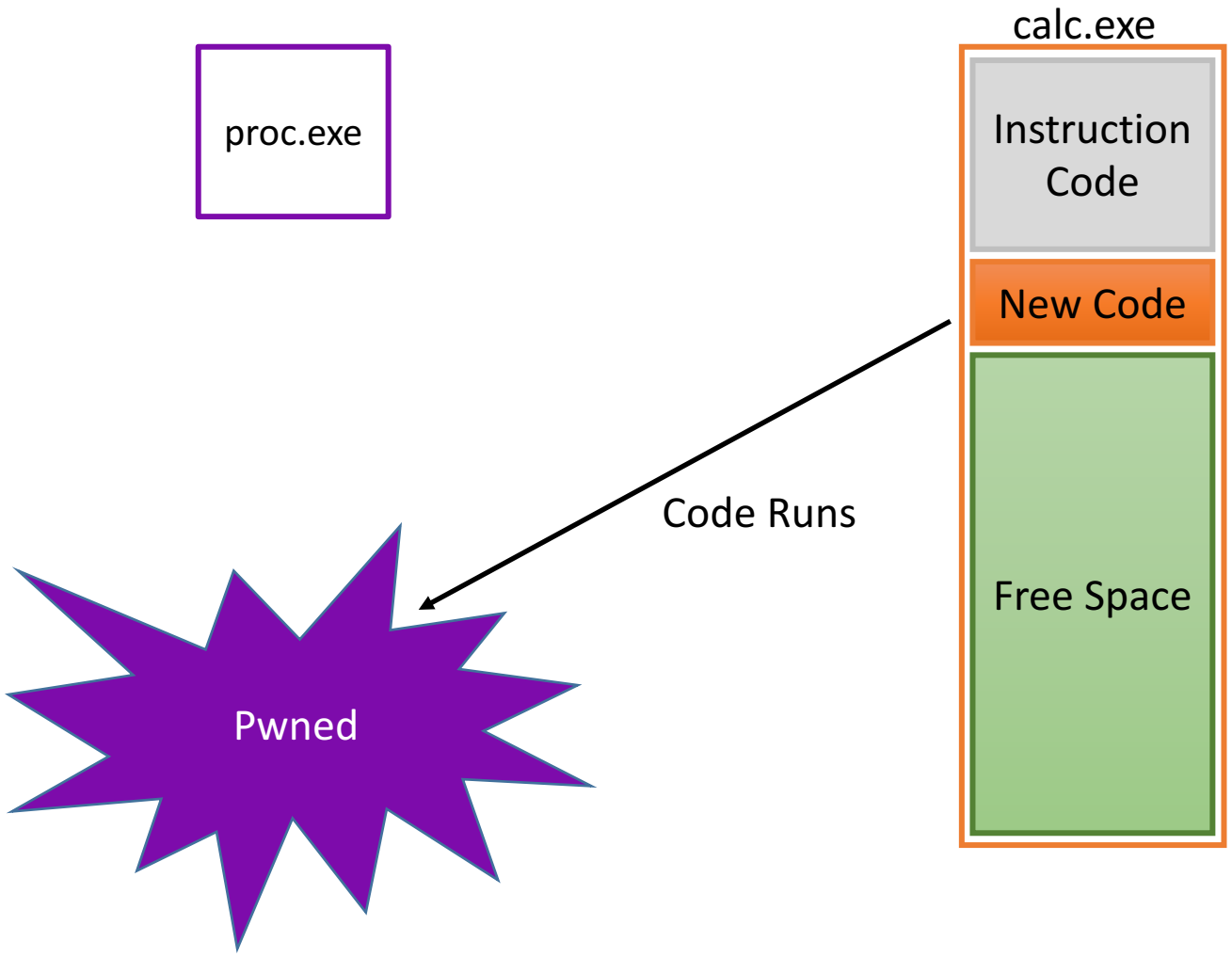
How memory Injection Works



How memory Injection Works



How memory Injection Works

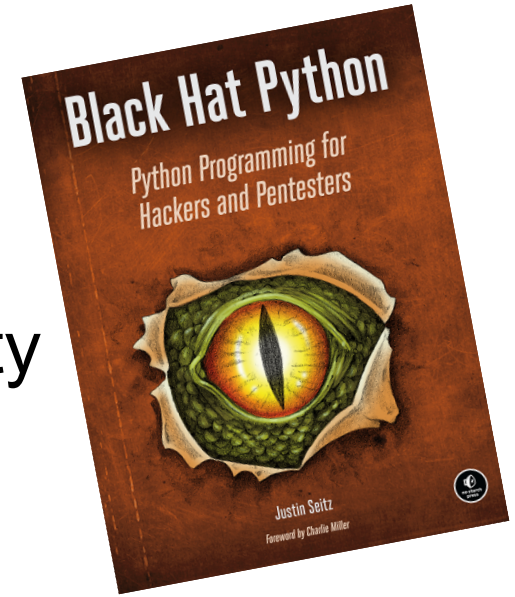


Importance of Memory Injection

- Become any process
  - That you have rights to
- Used in Malware
  - Called “Process Hijackers”
- Detected by Antivirus
- Evades file/real-time scanning
- Evades signature detection
- Nothing added to disk

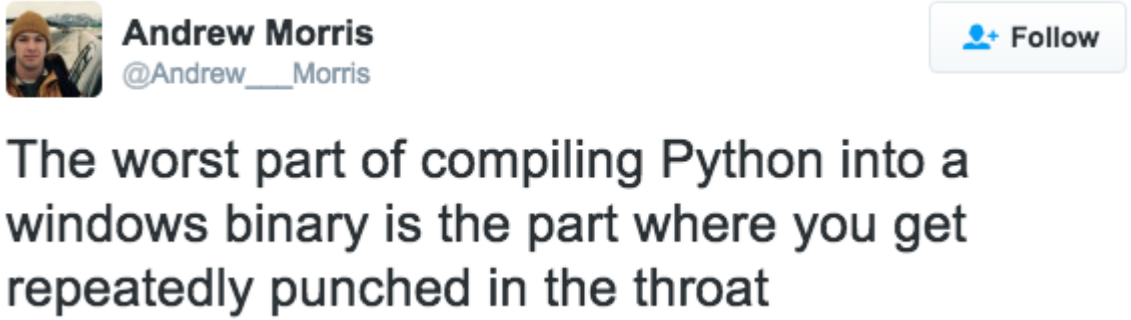
Memory Injection with Python

- Injection possible using python
- Covered in books
  - Black Hat Python
  - Gray Hat Python
- Only useful if Python is installed
- Issues with Python 3
- Can be compiled for compatibility



Memory Injection with Python (cont.)

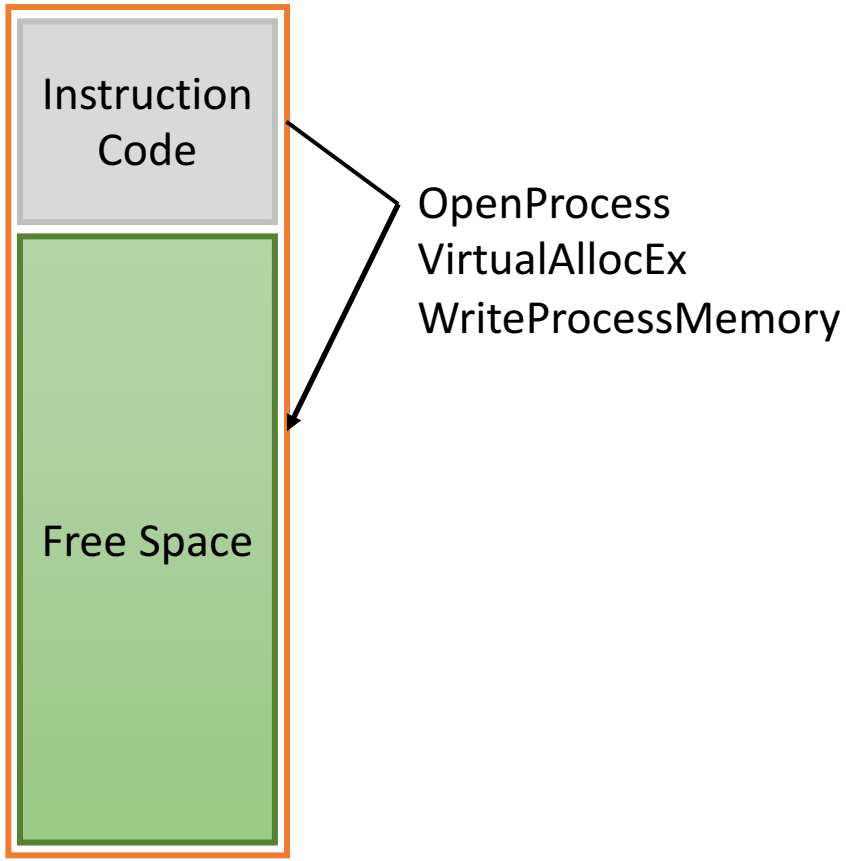
- Compiled Python can cause issues
  - Multiple files generated
  - Need a self extracting archive
  - Can look more suspicious



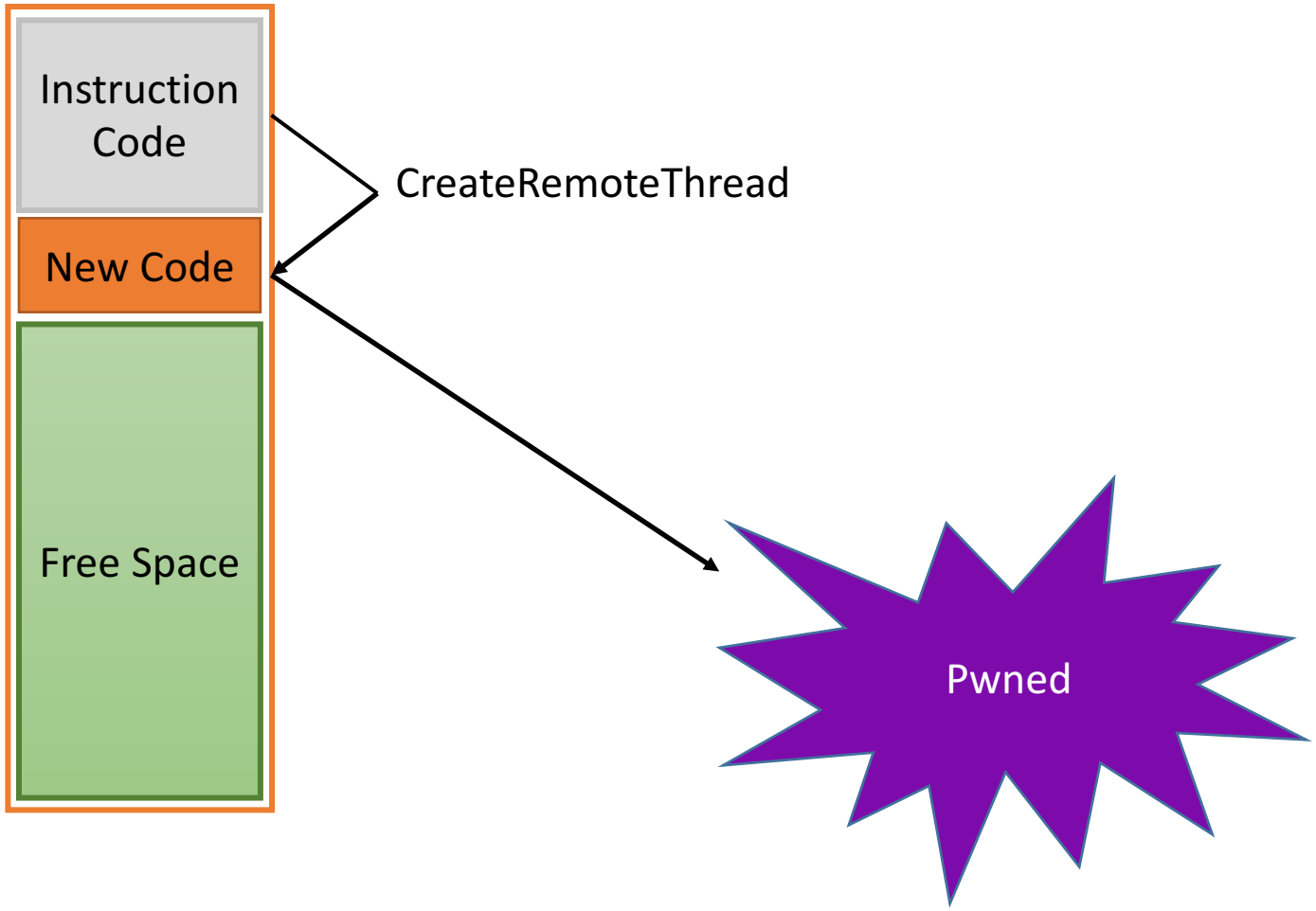
Process Local Memory Injection

- Malicious code can be hidden by using local injection
- Injection of code into current running process
- Not detected as a Process Hijacker
- Payload type in Veil Framework

Process Local Memory Injection



Process Local Memory Injection



Video 1: Meterpreter

virustotal

SHA256: 0355c1264c272e74ca5a004908d5e09155e3a1af8d7c25690ac3334babe79b92

File name: msf.exe

Detection ratio: 36 / 56

Analysis date: 2016-10-07 11:38:34 UTC (1 minute ago)

Analysis | File detail | Additional information | Comments | Votes

Antivirus	Result	Update
ALYac	Gen:Variant.Kazy.591095	20161007
AVware	Trojan.Win32.Generic!BT	20161007
Ad-Aware	Gen:Variant.Kazy.591095	20161007
AegisLab	Troj.W32.Jorik.Skor.IrUS	20161007
AhnLab-V3	Trojan/Win32.Swrort.C695042	20161007
Antiy-AVL	Trojan[:HEUR]/Win32.AGeneric	20161007
Arcabit	Trojan.Kazy.D904F7	20161007
Avast	Multi:Swrort-A [Trj]	20161007
Avira (no cloud)	TR/Crypt.XPACK.Gen	20161007
Baidu	Win32.Trojan.WisdomEyes.151026.9950.9999	20161001
BitDefender	Gen:Variant.Kazy.591095	20161007
Bkav	W32.eHeur.Virus02	20161007

Video 2: Local Injection

Why Local Injection Works

- Not touching any other process
- Contained in same process
- Process is ‘allowed’ to inject into itself
- Memory of process is not scanned

Why Local Injection Sucks

- Keeps the same process PID
- Runs in the same context as the payload executable
- Might not be trusted
  - Actions can be flagged by AV

AV Heuristic Detection

- Based on “Actions” of processes
  - Action Context
- Process signature
  - Is it a known process?
  - Is it a system process?
  - User running process?
  - Process execute path?
- Network connections made
- Sometimes labeled as “Threat Intelligence”

AV Heuristic Detection (cont.)

- Spot the bad processes

  C:\Windows\system32\svchost.exe as SYSTEM

  C:\Windows\explorer.exe as Bob

  C:\Windows\system32\notepad.exe as SYSTEM

  C:\Windows\regedit.exe as Bob

  C:\Windows\system32\cmd.exe as Bob

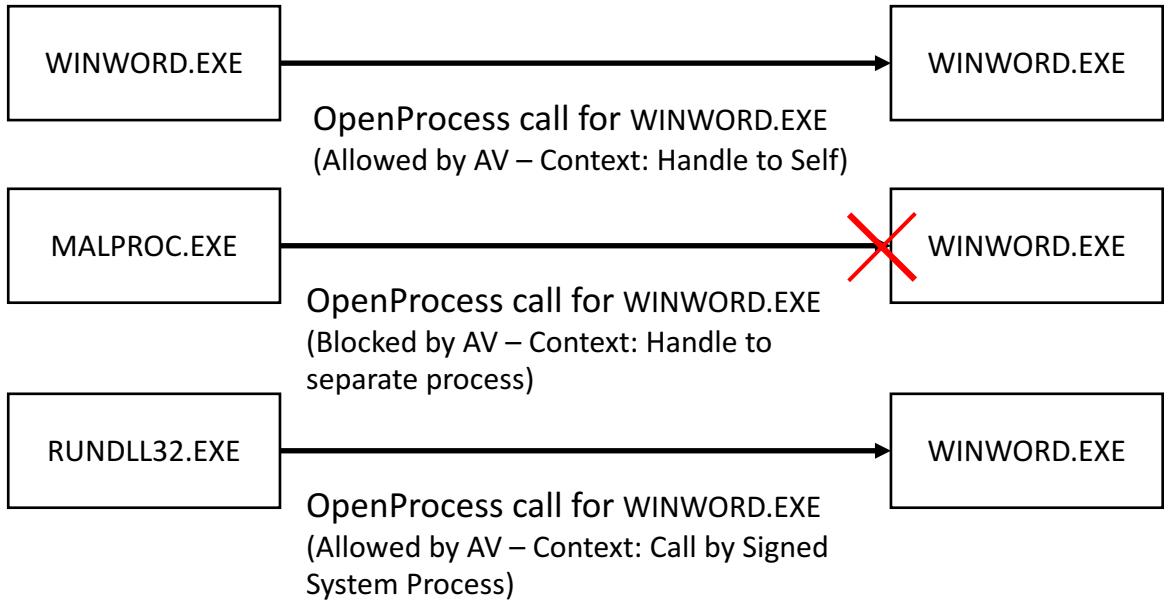
  C:\Users\Bob\AppData\Local\Temp\svchost.exe as Bob

AV Heuristic Detection (cont.)

- Bad Processes found
  - C:\Users\Bob\AppData\Local\Temp\svchost.exe as Bob
  - C:\Windows\system32\notepad.exe as SYSTEM
- Process Path
  - svchost is located in C:\Windows\system32 not Temp
- User
  - notepad running as SYSTEM

AV Heuristic Detection (cont.)

- Windows API Calls are also check for use by AV
- AV checks the context to see if the action looks malicious



Windows API Calls

- CreateRemoteThread
  - Creates thread in other process
- NTCreateThreadEx
  - Creates thread in other process
  - Undocumented!
- WriteProcessMemory
  - Write contents of process memory
- OpenProcess
  - Opens a handle and requests access to a process
- LoadLibraryA
  - Loads a DLL file into the current process

AV Trust

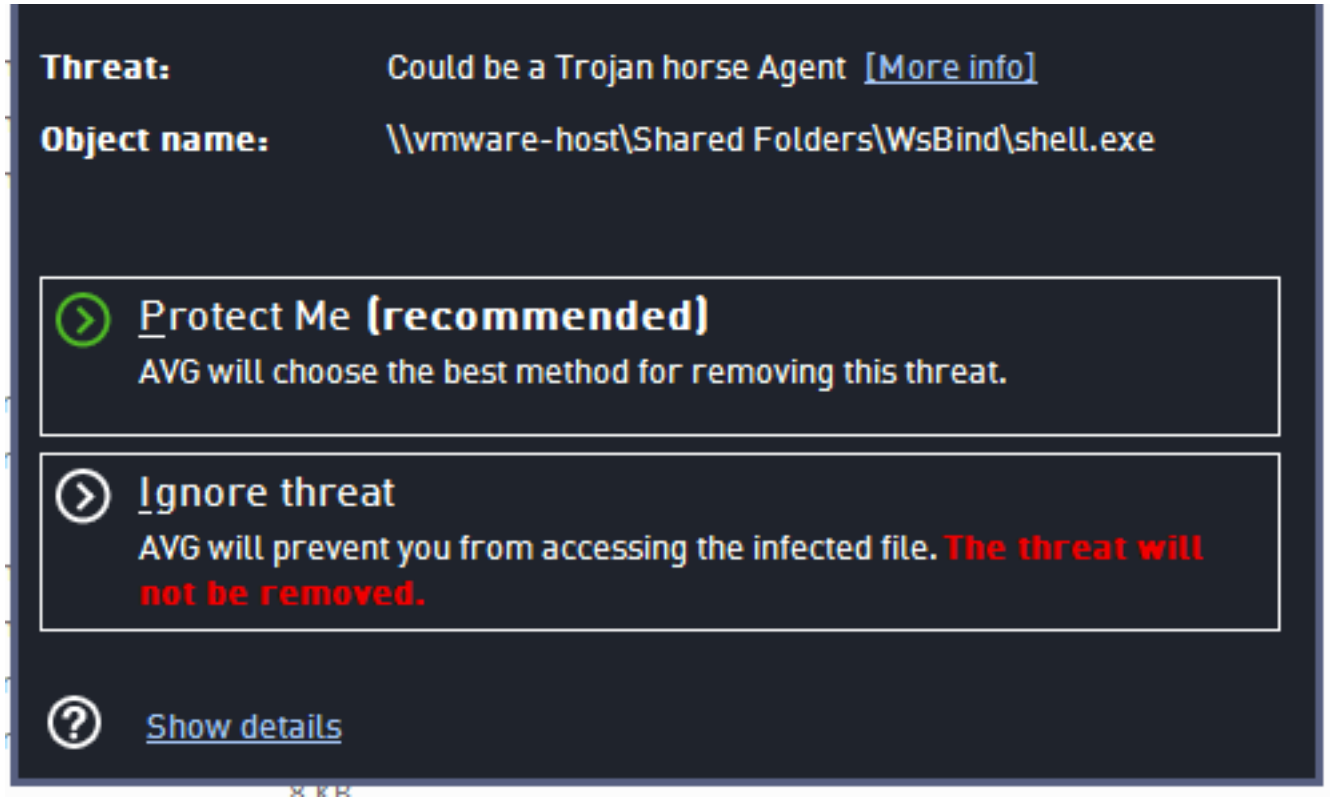
- Certain processes are trusted by AV
- Critical files and processes
  - System processes
  - AV data files
  - AV program files
- With no trust AV can damage an OS
  - AV software can break the OS
  - http://pub.idfla.me/links/iym2

Exploit Overview

- Memory Injection to hide Malware code
- AV trust in system processes
  - Actions by system processes are trusted
- System process to run our malicious code
  - While still being trusted
  - Without Admin access

Exploit Overview (cont.)

Answer is rundll32.exe


Threat:
Could be a Trojan horse Agent [More info]
Object name:
\\vmware-host\Shared Folders\WsBind\shell.exe
Protect Me (recommended)
AVG will choose the best method for removing this threat.
Ignore threat
AVG will prevent you from accessing the infected file. The threat will not be removed.
Show details

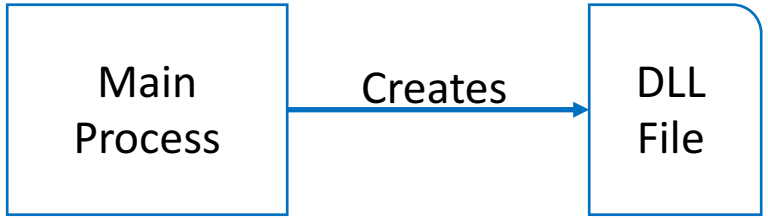
Video 3: Exploit

What happened?

- Step 1
  - Main process creates runnable DLL file with basic payload
  - Contains memory injection instructions and shellcode
- Step 2
  - Main process calls rundll32 to execute DLL file
- Step 3
  - Our DLL is now running as rundll32
  - Enumerates all running processes
  - Picks process we have access to
- Step 4
  - Inject shellcode into picked process
  - Seen by AV as rundll32, action is allowed
  - Shellcode runs

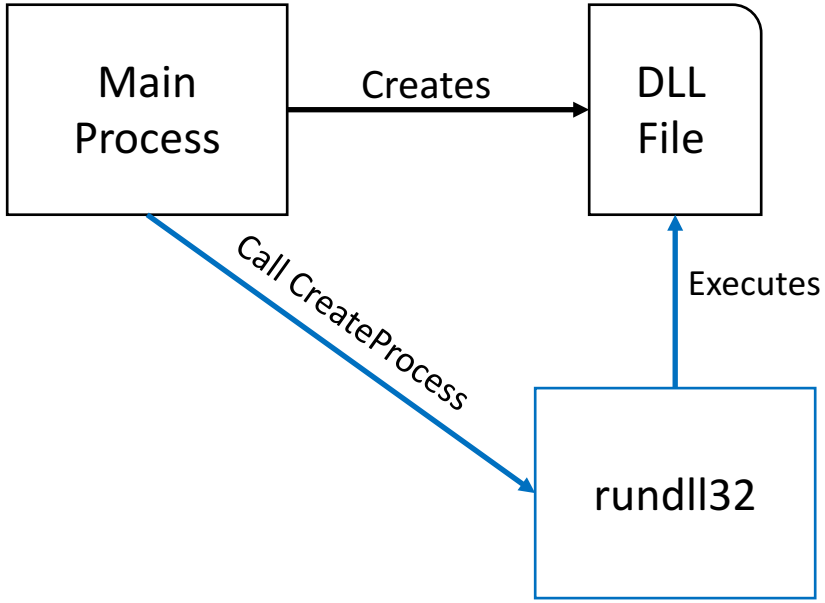
What happened? (cont.)

- Step 1
  - Main process creates runnable DLL file with basic payload
  - Contains memory injection instructions and shellcode



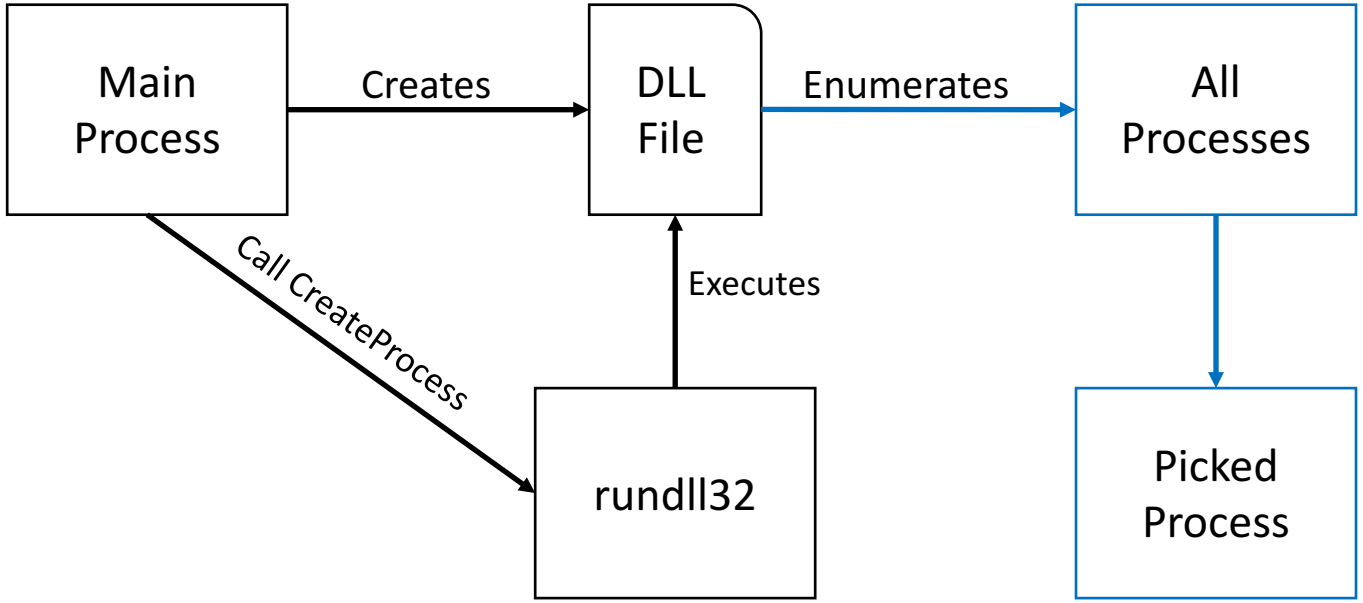
What happened? (cont.)

- Step 2
  - Main process calls rundll32 to execute DLL file



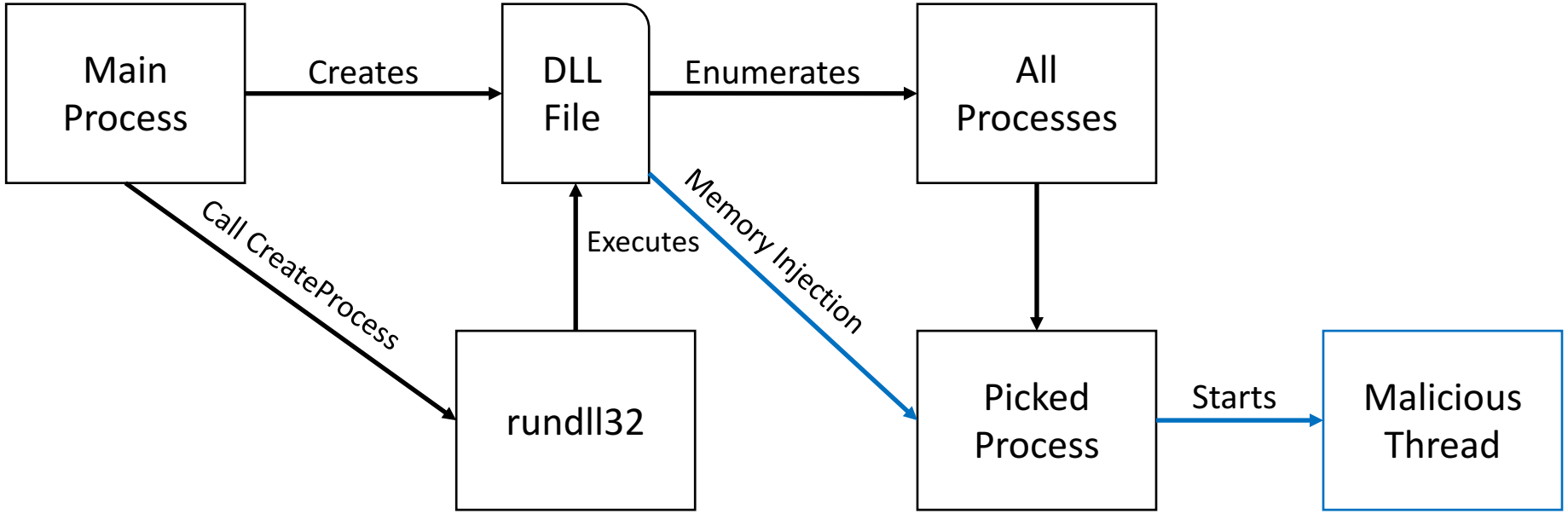
What happened? (cont.)

- Step 3
  - Our DLL is now running as rundll32
  - Enumerates all running processes
  - Picks process we have access to



What happened? (cont.)

- Step 4
  - Inject shellcode into picked process
  - Seen by AV as rundll32, action is allowed
  - Shellcode runs



Video 4: Bonus Video

Compatibility

- Works on Windows 7 & 10

```
[*] Command shell session 2 opened (192.168.253.130:443 -> 192.168.253.134:50188) at 2016-10-07 09:03:59 -0400

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

```
[*] Command shell session 7 opened (192.168.253.130:443 -> 192.168.253.136:49688) at 2016-10-07 09:50:15 -0400

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Detection and Prevention

- Detection
  - Redline
  - Volatility
  - Memory Dumps
- Prevention
  - Flag the DLL/Loader as infected
  - Enable loading signed DLLs only
  - Disable loading DLLs from non-program directories
  - rundll32 specify the DLL as the running application

Code and Downloads

- Code is currently Live on GitHub
  - https://github.com/iDigitalFlame/InYourMems
  - http://idfla.me/iym
- Links and References at
  - http://idfla.me/iym

Questions?