



Malware Analysis for Incident Responders

Agustin Gonzalez

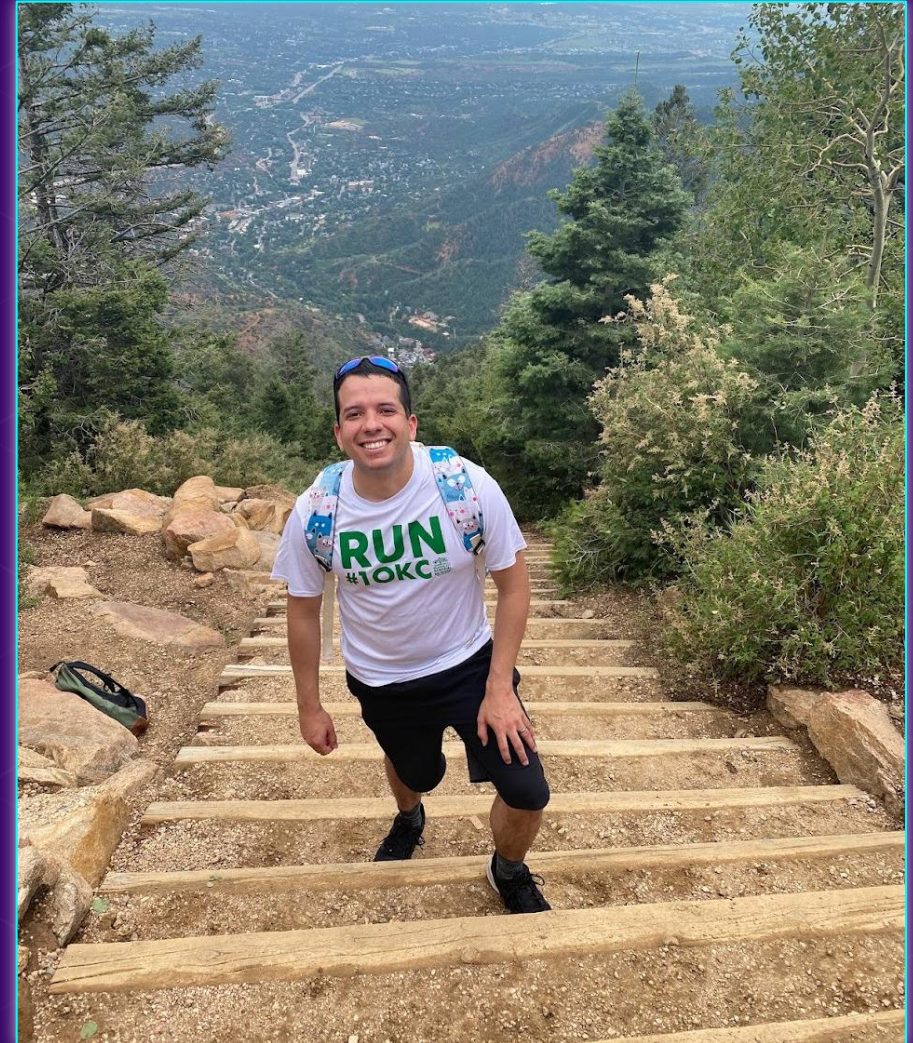


whoami

- Agustin Gonzalez – CISSP
- Puerto Rico
- Cyber Defense Engineer, U.S Air Force
- 8 years of cyber security experience
 - Security Analyst
 - Incident Responder
 - Malware Analyst
 - Red Team
 - Purple Team
- Master's Degree in Cybersecurity
- Certified in GIAC GREM,GCFA,GCIH and GPEN
- Love Tinkering with Computers

<https://www.linkedin.com/in/agustin227/>

<https://twitter.com/aqu227>



Overview

[Scenario](#)

[What is Malware Analysis?](#)

[Why Malware Analysis?](#)

[Malware Analysis Thought Process](#)

[Malware Analysis Lab Setup](#)

[Stages of Malware Analysis](#)

[Key Takeaways](#)

[Resources](#)



Scenario

- Suspicious program (executable) is found
- We don't know what it is or how it got there or what is doing
- What do we do?
 - Do we just delete it and be done with it?
 - How do we know the impact of that piece of software?
- OSINT seems to not have the answers
- We need IOCs





Now
What?



Malware Analysis
is the Answer!

What is Malware Analysis?

- Malware analysis is the process of studying malicious software (malware) to understand its functionality, origin, and potential impact. It is a critical part of cybersecurity, as it helps security teams to identify and mitigate threats before they can cause damage. (By Bard)
- Allows to Identify the type of malware



Why Malware Analysis?

- Malicious software is an integral component of most security incidents
- Most people don't understand what the malicious software does
- Understanding how to analyze malware enables you and your organization take control of incidents
 - Determine scope of incident
 - Understand the threat to your organization
 - What are the software capabilities?
 - Completely eradicate malicious artifacts across the enterprise (Threat Hunting?)
 - Fortify system and network defenses thanks to the discovery of IOCs and creation of signatures
- Contribute to the cyber intelligence community
 - What does the software reveal about the creator (adversary)



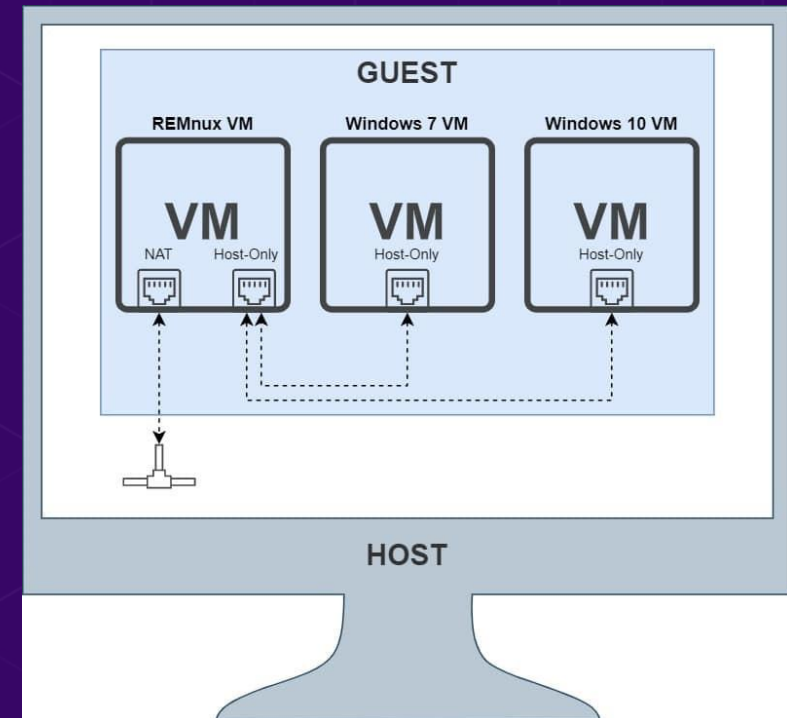
Malware Analysis Resources

- Use online resources like:
 - File Reputation- Virustotal, #totalhash, Malware Hash Registry
 - Datasets - Winbindx
 - Automated Malware Analysis Platforms - any.run, Hybid Analysis, Joe Sandbox, Falcon Sandbox
 - URL/IPvoid Research - vURL, Quttera, urlscan.io, urlvoid, Talos
 - Cyber Intelligence - Shodan.IO, OTX, RiskIQ, Talos, CISA, US-CERT, Mandiant.
- Warning - Use caution when utilizing external resources due to:
 - Attribution
 - Alert the adversary



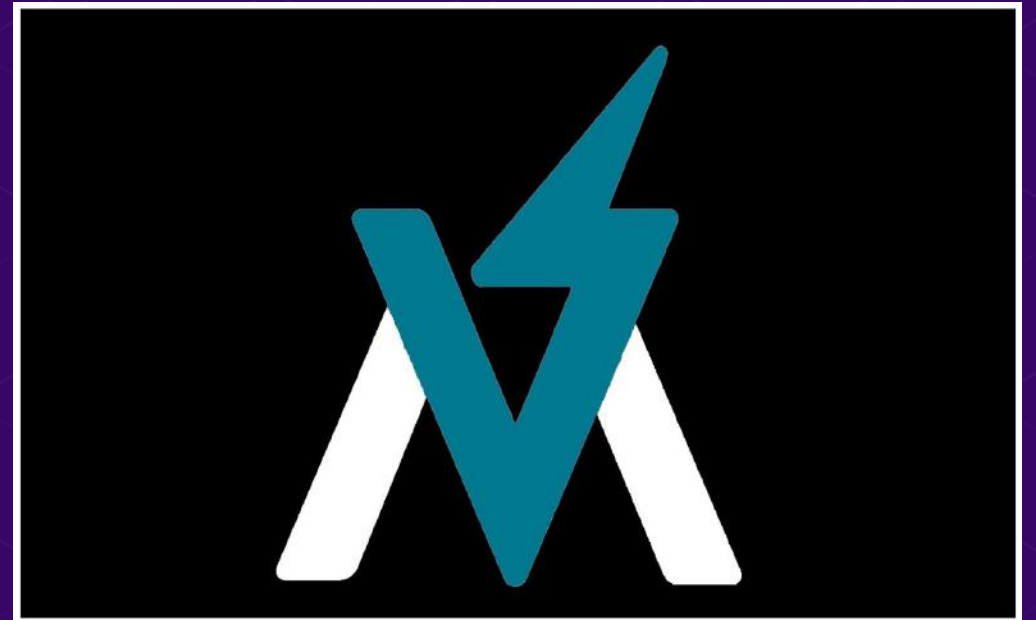
Malware Analysis Lab

- Isolate Lab from other networks
- Virtual Lab (VMs)
 - Convenient
 - Multiple hosts on one system
 - Internal Networking
 - Snapshots
 - VM Escape Exploits (Rare)
 - VM-Aware Malware
- Physical Systems are an option but outside the scope in here

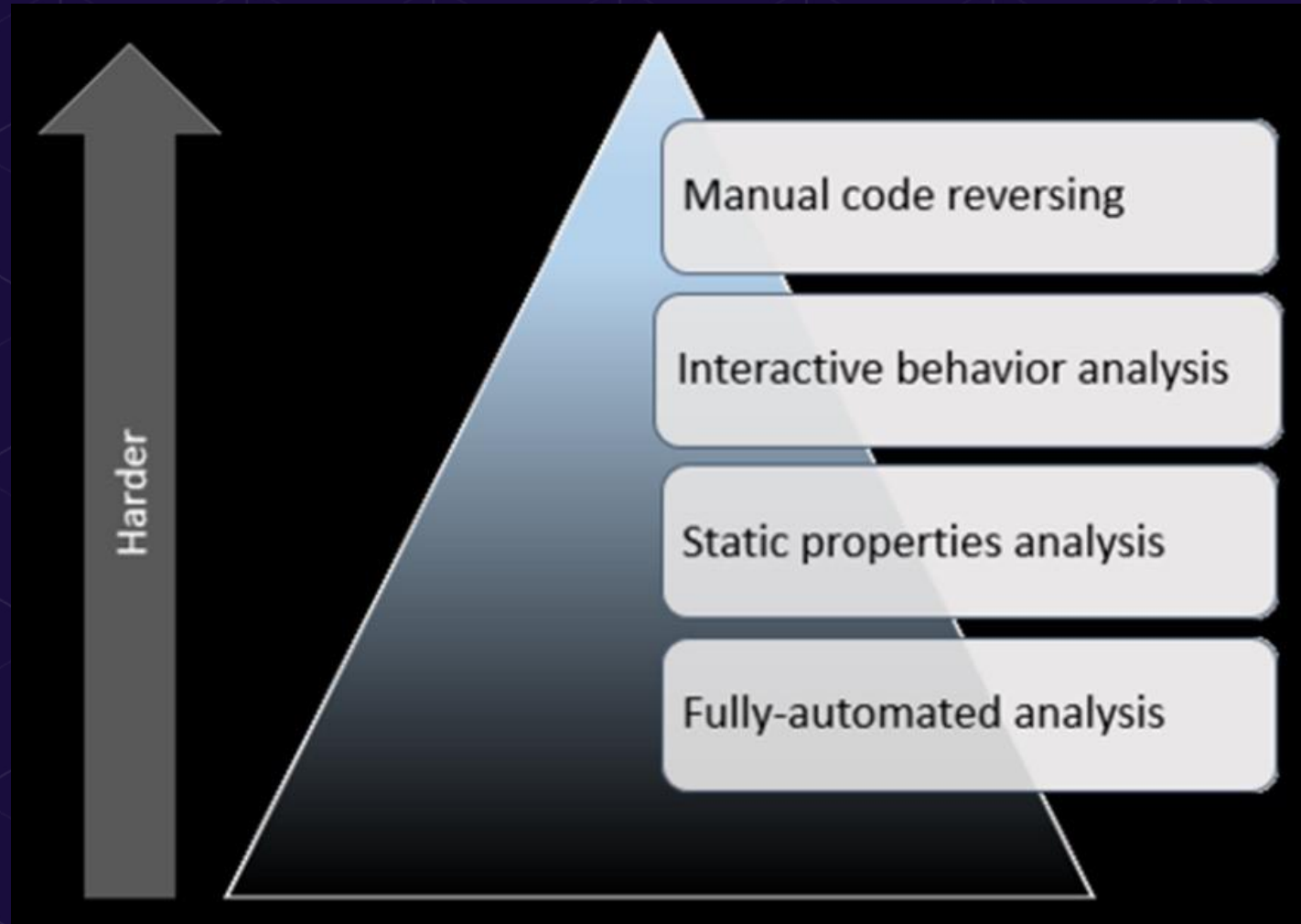


Malware Analysis Lab

- Lab should include these capes:
 - Static properties analysis tools
 - Behavioral analysis tools
 - Code Analysis Tools
- Distros
 - REMnux – Linux Based Malware Analysis Distro
 - Flare VM – Script that automates the installation of malware analysis and RE on a Windows VM.

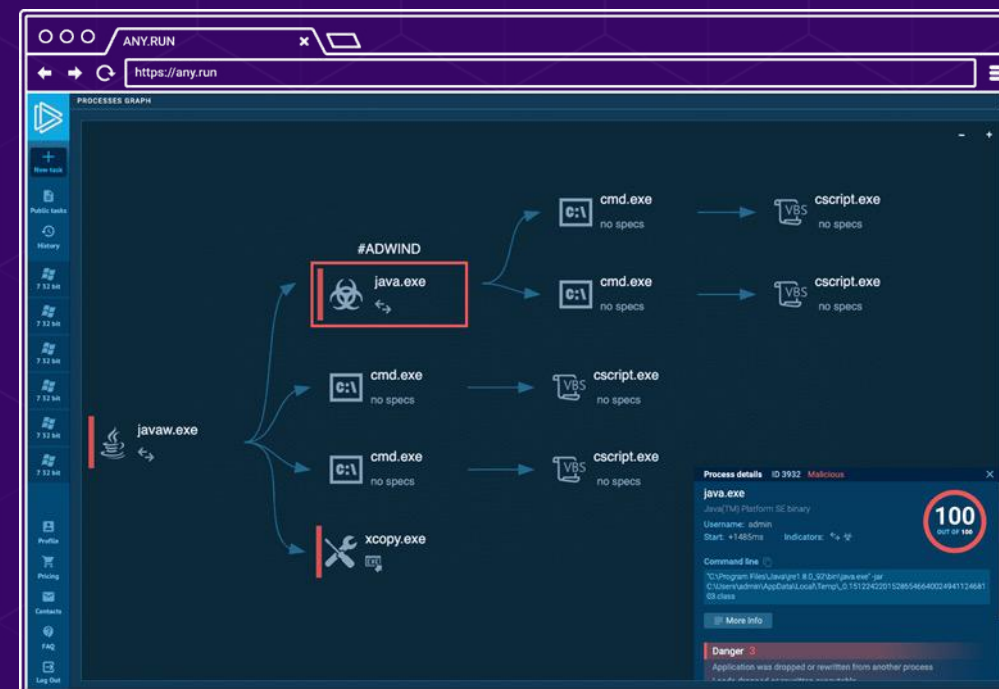


Malware Analysis Stages



Fully Automated Analysis

- Quick assessment
- Produces easy to understand reports
- Not as flexible as manual analysis
- Some malware will evade or refuse to run on automated platforms
- Examples
 - Cuckoo Sandbox
 - Joe Sandbox
 - Any.run
 - Hybrid analysis
 - Falcon Sandbox
 - Mockingbird



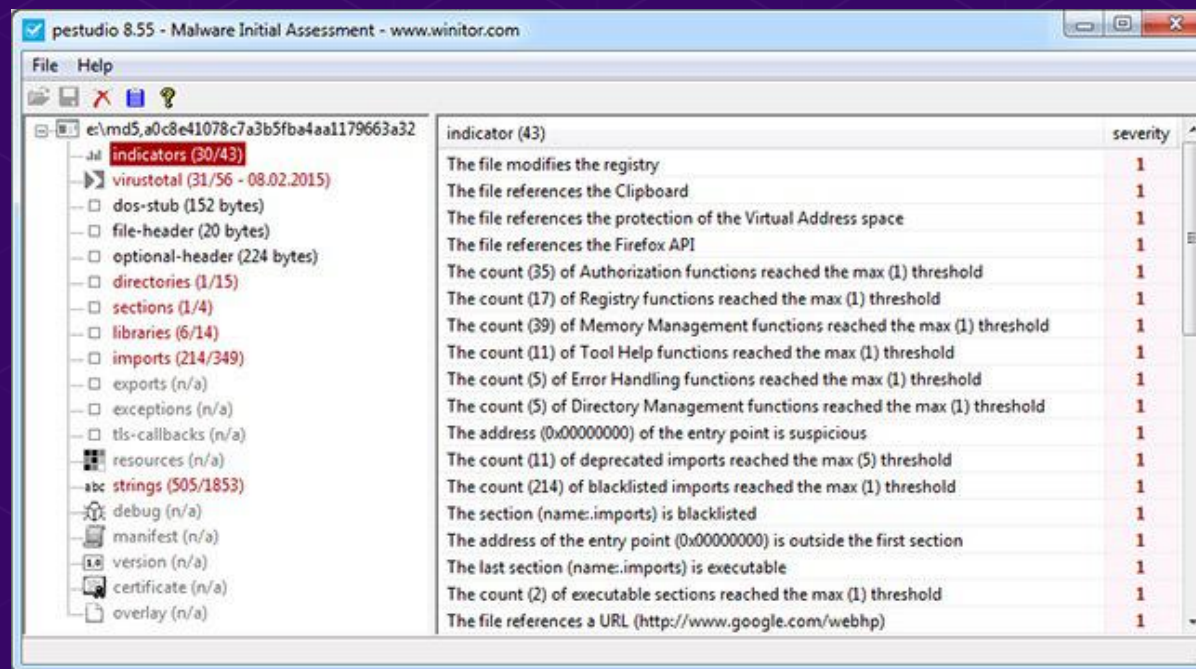
Static Properties Analysis

- Also know as metadata analysis
- Initial triage of a artifact
- Analyze strings, header and overall structure of the artifact
- Does not involve executing the malware
- Capabilities include extracting indicators such as:
 - Hashes, IP addresses and Domains
 - Imports and Exports (DLLs)
 - File Headers
 - Checking if malware is obfuscated or packed
 - API Calls – Provide functions
 - Entropy
 - ImpHash

```
1. !This program cannot be run in DOS mode. 23. o/o/
2. Rich 24. advapi32
3. .text 25. ntdll
4. .data 26. user32
5. ExitProcess 27. 1+KY
6. kernel32.dll 28. #%li
7. ws2_32 29. ]>*K
8. cks=u 30. QQVP
9. ttp= 31. advpack
10. cks= 32. StubPath
11. CONNECT %s:%i HTTP/1.0 33. SOFTWARE\Classes\http\shell\open\commandV
12. QSRW 34. Software\Microsoft\Active Setup\Installed Components\
13. ?503 35. test
14. 200 36. www.practicalmalwareanalysis.com
15. thj@h 37. admin
16. VSWRQ 38. VideoDriver
17. YZ_[^ 39. WinVMX32-
18. s f5 40. vmx32to64.exe
19. YZ_[^ 41. SOFTWARE\Microsoft\Windows\CurrentVersion\Run
20. QVIM 42. SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
21. 6!*h<8 43. AppData
22. ^-m-m<|<|<|M 44. V%X_
```

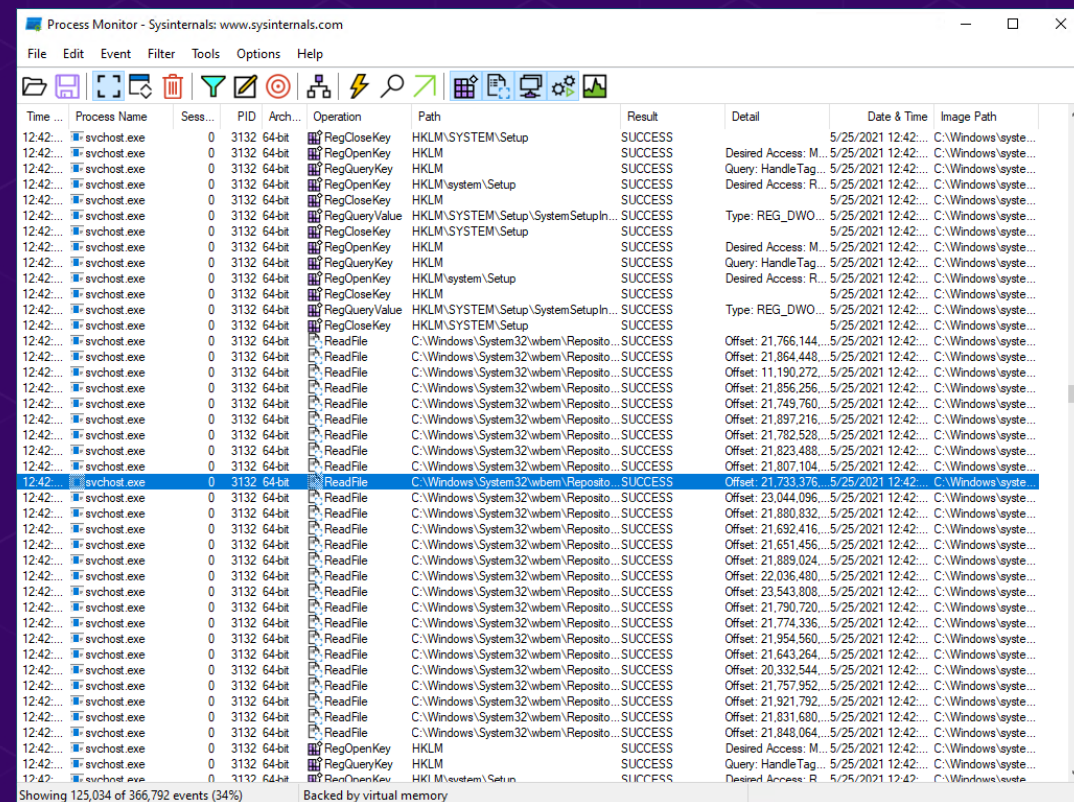
Static Properties Analysis

- Obtain IOCs
 - Compare with OSINT
 - Create Signatures
- Not very useful for packed/obfuscated malware
- Tools
 - PEStudio, PEFrame
 - Strings (Floss, String Sifter, Bintext, Strings)
 - Detect it Easy (DIE), EXEinfo
 - CFF Explorer
 - Capa
 - Bulk Extractor
 - Manalyze
 - Cybershef
 - Hxd



Interactive Behavioral Analysis

- Also known as dynamic analysis
- Execute and analyze the malware as it runs on the system
 - Complete visibility on the malware actions
 - Registry actions
 - Read/write actions on disk
 - Network connections
 - Persistence mechanisms
 - Mutexes
 - Process modification (injects, threads)
- Some malware will fail to run due to defense mechanisms
 - Fail to run on virtualized systems
 - Difference actions if it detects analysis tools or VM
 - Self-Deletion
- Very useful to analyze packed malware
- Can produce extra artifacts for analysis

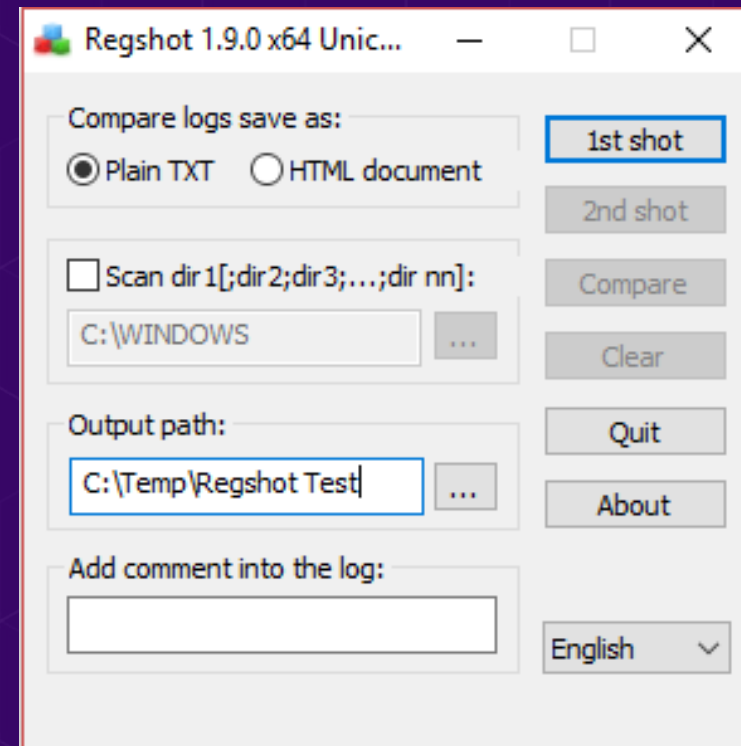


The screenshot displays the Process Monitor application window, showing a list of system events for the process svchost.exe. The table below represents the data shown in the application's main pane.

| Time | Process Name | Session | PID | Arch | Operation | Path | Result | Detail | Date & Time | Image Path |
|-----------|--------------|---------|------|-------|---------------|--------------------------------------|---------|-----------------------|---------------------|----------------------|
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegCloseKey | HKLM\SYSTEM\Setup | SUCCESS | | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegOpenKey | HKLM | SUCCESS | Desired Access: M... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegQueryKey | HKLM | SUCCESS | Query: HandleTag... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegOpenKey | HKLM\system\Setup | SUCCESS | Desired Access: R... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegCloseKey | HKLM | SUCCESS | | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegQueryValue | HKLM\SYSTEM\Setup\SystemSetupIn... | SUCCESS | Type: REG_DWO... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegCloseKey | HKLM\SYSTEM\Setup | SUCCESS | | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegOpenKey | HKLM | SUCCESS | Desired Access: M... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegQueryKey | HKLM | SUCCESS | Query: HandleTag... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegOpenKey | HKLM\system\Setup | SUCCESS | Desired Access: R... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegCloseKey | HKLM | SUCCESS | | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegQueryValue | HKLM\SYSTEM\Setup\SystemSetupIn... | SUCCESS | Type: REG_DWO... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegCloseKey | HKLM\SYSTEM\Setup | SUCCESS | | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,766,144... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,864,448... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 11,190,272... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,856,256... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,749,760... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,897,216... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,782,528... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,823,488... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,807,104... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,733,376... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 23,044,096... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,880,832... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,692,416... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,651,456... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,889,024... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 22,036,480... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 23,543,808... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,790,720... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,774,336... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,954,560... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,643,264... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 20,332,544... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,757,952... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,921,792... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,831,680... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | ReadFile | C:\Windows\System32\wbem\Reposito... | SUCCESS | Offset: 21,848,064... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegOpenKey | HKLM | SUCCESS | Desired Access: M... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegQueryKey | HKLM | SUCCESS | Query: HandleTag... | 5/25/2021 12:42:... | C:\Windows\sysste... |
| 12:42:... | svchost.exe | 0 | 3132 | 64bit | RegOpenKey | HKLM\system\Setup | SUCCESS | Desired Access: R... | 5/25/2021 12:42:... | C:\Windows\sysste... |

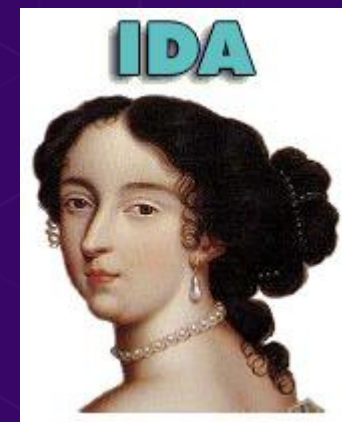
Interactive Behavioral Analysis

- Can adapt to malware needs open inner stages of execution
- Not perfect
- Tools
 - Process Hacker or Process Explorer
 - ProcMon
 - Regshot
 - Procdot
 - Wireshark
 - Fakenet
 - FakeDNS
 - Fiddler
 - Cmd watcher
 - CaptureBat



Manual Code Reversing

- Also known as Code Analysis
- Hardest way to analyze malware
- Binary Patching
- Nothing can hide
- Debuggers
 - Allow for static and dynamic code analysis
 - Step by step execution (process stepping)
 - Tools
 - WinDBG, x64dbg, Radare2, X64dbg
- Disassemblers
 - Translates machine level instructions to assembly code and sometimes to high level code
 - Tools
 - Ghidra
 - Ida



X64dbg

The screenshot displays the X64dbg debugger interface for a process named 267.exe. The main window shows assembly code with the following instructions:

```
0040E022 <267.EntryPoint> E8 599C0000 call <267.sub_417C80>
0040E027 E9 16FEFFFF jmp 267.40E042
0040E02C CC int3
0040E02D CC int3
0040E02E CC int3
0040E02F CC int3
0040E030 <267.sub_40E030> 8B4C24 04 mov ecx,dword ptr ss:[esp+4]
0040E034 F7C1 03000000 test ecx,3
0040E03A 74 24 je 267.40E060
0040E03C 8A01 mov al,byte ptr ds:[ecx]
0040E03E 83C1 01 add ecx,1
0040E041 84C0 test al,al
0040E043 74 4E je 267.40E093
0040E045 F7C1 03000000 test ecx,3
0040E04B 75 EF jne 267.40E03C
0040E04D 05 00000000 add eax,0
0040E052 8DA424 00000000 lea esp,dword ptr ss:[esp]
0040E059 8DA424 00000000 lea esp,dword ptr ss:[esp]
0040E060 8B01 mov eax,dword ptr ds:[ecx]
0040E067 8A FFEFE7E add edx,edx
0040E069 83F0 FF xor eax,FFFFFFFF
0040E06C 33C2 xor eax,edx
0040E06E 83C1 04 add ecx,4
0040E071 A9 00010181 test eax,81010100
0040E076 74 E8 je 267.40E060
0040E078 8B41 FC mov eax,dword ptr ds:[ecx-4]
0040E07B 84C0 test al,al
0040E07D 74 32 je 267.40E0B1
0040E07F 84E4 test ah,ah
0040E081 74 24 je 267.40E0A7
0040E083 7A 13 test eax,FF0000
0040E088 74 13 je 267.40E076
```

The right-hand pane shows the register window with the following values:

```
EAX 75B13428 <kernel32.BaseThreadInitThunk>
EBX 7EFDE000
ECX 00000000
EDX 0040E022 <267.EntryPoint>
EBP 0018FF94
ESP 0018FF8C
ESI 00000000
EDI 00000000
EIP 0040E022 <267.EntryPoint>
```

The bottom pane shows a memory dump at address 75B13430:

```
0018FF8C 75B13430 return to kernel32.75B13430 from ???
0018FF90 7EFDE000
0018FF94 0018FFD4
0018FF98 77019802 return to ntdll.77019802 from ???
0018FF9C 7EFDE000
0018FFA0 775B1805
0018FFA4 00000000
0018FFA8 00000000
0018FFAC 7EFDE000
0018FFB0 00000000
0018FFB4 76E57C7F return to 76E57C7F from 76E57920
0018FFB8 00000000
0018FFBC 0018FFA0
0018FFC0 00000000
0018FFC4 FFFFFFFF End of SEH Chain
```

The status bar at the bottom indicates: Paused INT3 breakpoint "entry breakpoint" at <267.EntryPoint> (0040E022) Time Wasted Debugging: 0:00:03:58

Ghidra

The screenshot displays the Ghidra CodeBrowser interface for the file `Emotet/5vdk9ge0f_7.exe`. The main window shows assembly code for the `entry` function, with the instruction `PUSH 0x60` at address `0040de42` highlighted. The assembly listing includes:

```
0040de33 39 88 e8    CMP     dword ptr [EAX + 0x4000e8]=>IMAGE_NT_HEAD
              00 40 00
0040de39 0f 95 c1    SETNZ  CL
0040de3c 8b c1      MOV     EAX, ECX
0040de3e c3        RET
              LAB_0040de3f                                XREF[4]:
0040de3f 33 c0     XOR     EAX, EAX
0040de41 c3        RET
              LAB_0040de42                                XREF[1]:
0040de42 6a 60     PUSH   0x60
0040de44 68 68 a6   PUSH   DAT_0045a668
              45 00
0040de49 e8 36 7d   CALL   __SEH_prolog4
              00 00
0040de4e 83 65 fc   AND    dword ptr [BBP + Stack[-0x8]], 0x0
              00
0040de52 8d 45 90   LEA   EAX=>Stack[-0x74], [EBP + -0x70]
0040de55 50       PUSH  EAX
0040de56 ff 15 f0   CALL  dword ptr [->KERNEL32.DLL::GetStartupInfo
```

The decompiled code on the right shows the `entry` function signature and its body:

```
1 /* WARNING: Function: __SEH_prolog4 repla
2 /* WARNING: Function: __SEH_epilog4 repla
3 /* WARNING: Globals starting with '_' ove
4
5
6 WPARAM entry(void)
7
8 {
9     HANDLE hHeap;
10    BOOL BVar1;
11    int iVar2;
12    uint uVar3;
13    DWORD dwFlags;
14    SIZE_T dwBytes;
15    LPSOFTWAREVERSION lpVersionInformation;
16    _STARTUPINFOA local_74;
17    DWORD local_2c;
18    DWORD local_28;
19    DWORD local_24;
20    WPARAM local_20;
21    undefined4 uStack12;
22    undefined4 local_8;
23
24    __security_init_cookie();
```

The console at the bottom shows the message: `Successfully compiled: WindowsResourceReference.java`. The status bar at the bottom indicates the current address is `0040de42`, the instruction is `entry`, and the operation is `PUSH 0x60`.

Key Takeaways

- Malware analysis can provide guidance and perspective
- Some malware samples will require tweaking and inputs
- You don't have to do all the stages of malware analysis
- Like programming, practice is the key to success
- **Must do!!** to understand how to perform defense better
- Code analysis is not meant for Incident Responders to accomplish.



Resources

- Malware Repositories
 - Malshare
 - theZoo
 - Malware Traffic Analysis
 - Virusshare
- Courses
 - Malware Unicorn
 - PMA (TCM Academy)
 - FOR610 (SANS)
 - Malware Noob2Ninja
- Books
 - Practical Malware Analysis
 - IDA Pro Book
 - Malware Data Science
 - Practical Binary Analysis
- CTFs
 - Flare CTF
 - Zombie Land CTF
- Github
 - <https://github.com/rshipp/awesome-malware-analysis>



Summary

[Scenario](#)

[What is Malware Analysis?](#)

[Why Malware Analysis?](#)

[Malware Analysis Thought Process](#)

[Malware Analysis Lab Setup](#)

[Stages of Malware Analysis](#)

[Key Takeaways](#)

[Resources](#)



Questions



Thank you!!

Slides - <https://tinyurl.com/peakcybermalware>

<https://www.linkedin.com/in/agustin227/>

<https://twitter.com/agu227>