UNDERSTANDING AND HIDING YOUR OPERATIONS

ATTL4S & ElephantSe4l

ATTL4S

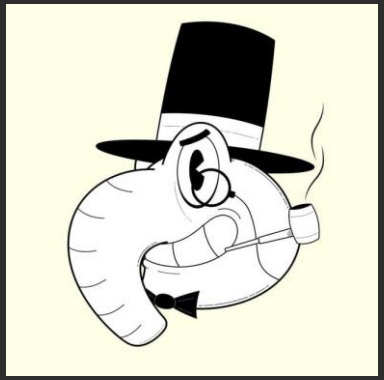
- Daniel López Jiménez (a.k.a. ATTL4S)
  - Twitter: @DaniLJ94
  - GitHub: @ATTL4S
  - Youtube: ATTL4S
- Loves Windows and Active Directory security
  - Senior Security Consultant at NCC Group
  - Associate Teacher at Universidad Castilla-La Mancha (MCSI)

Confs: NavajaNegra, No cON Name, h-c0n, Hack&Beers

Posts: Crummie5, NCC Group's blog, Hackplayers

Certs: CRTO, PACES, OSCP, CRTE

ElephantSe4l



- Godlike Programmer and Elephant Seal
  - Twitter: @ElephantSe4l
  - GitHub: @ElephantSe4l
- Very curious, he enjoys understanding complex and weird things
- Mind behind all the low-level contents of my talks

This has been written by ATTL4S

WWW.CRUMMIE5.CLUB

The goal of this talk is being a resource for comprehending the meaning of OPSEC and creating awareness in your operations, so as you can successfully face – and <u>improve</u> – experienced security teams and their detection capabilities

Agenda

1. Introduction
2. Become your Adversary
3. Facing a Mature Adversary
4. Why you should be using a C2

Adversary Simulation

- You probably heard of **Red Team** assessments

"Emulation of adversarial behaviours and techniques used by real-world threat actors"

- The goal of these assessments is improving the organisation's security team

- One of the most important things a threat actor will take care of is not getting caught!

This is Charles, Senior Red Team Analyst (OSCP/OSCE/OSWP/CEH/BB)

He is doing a Red Team engagement for a client

A successful Phising campaign gave him a juicy Meterpreter's session

```
msf6 exploit(multi/handler) > [*] http://10.11.1.130:4444 handling request from 10.11.3.5; (UUID: w
owooad0) Redirecting stageless connection from /XTZf9IGB5n_4Z_llp6x4JAtei00zpGGU-Ef with UA 'Mozill
a/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko'
[*] http://10.11.1.130:4444 handling request from 10.11.3.5; (UUID: wowooad0) Attaching orphaned/st
ageless session...
[*] Meterpreter session 6 opened (10.11.1.130:4444 -> 10.11.3.5:58146) at 2020-12-05 04:42:43 -0800
msf6 exploit(multi/handler) >
```

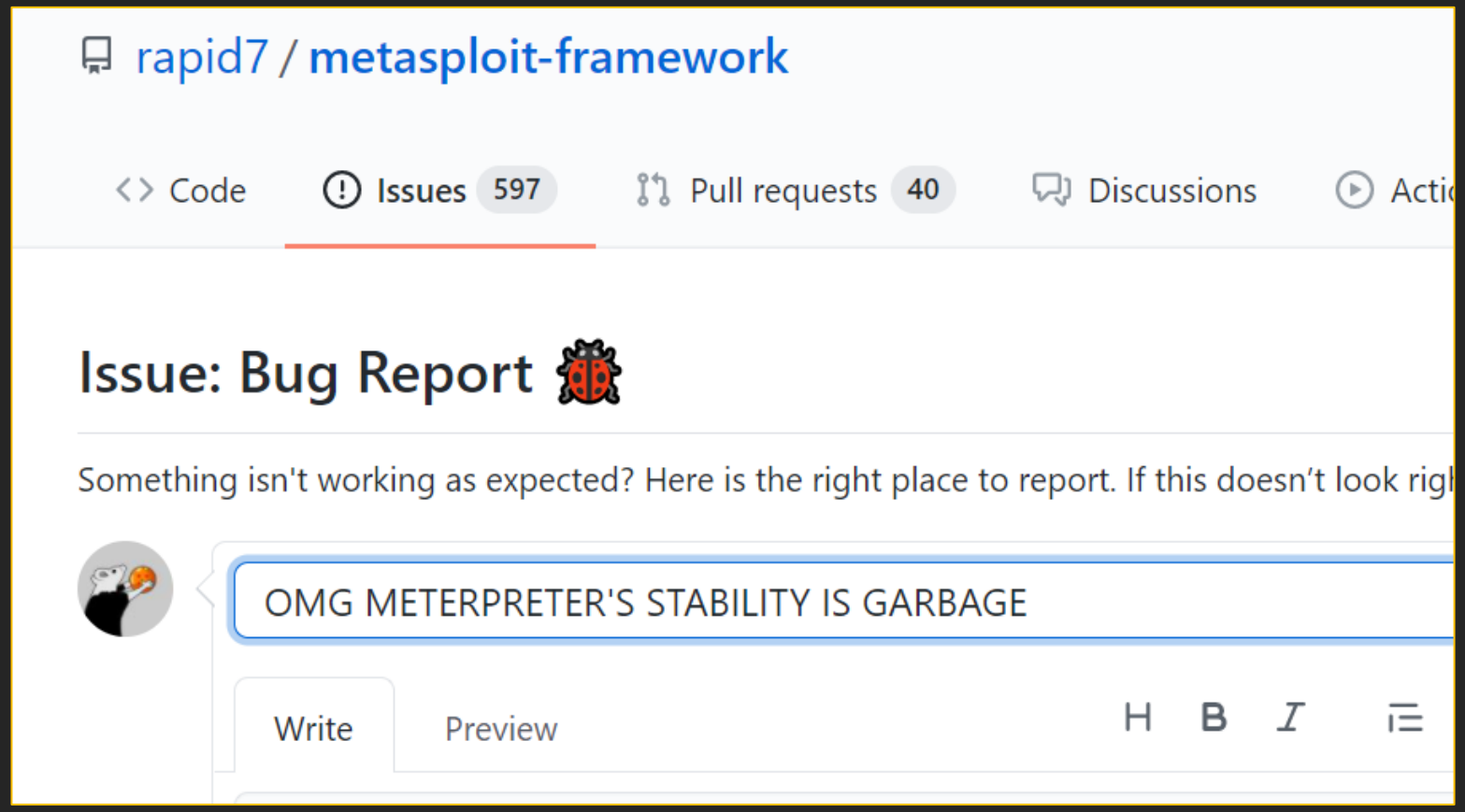
He started doing a local enumeration within the compromised system

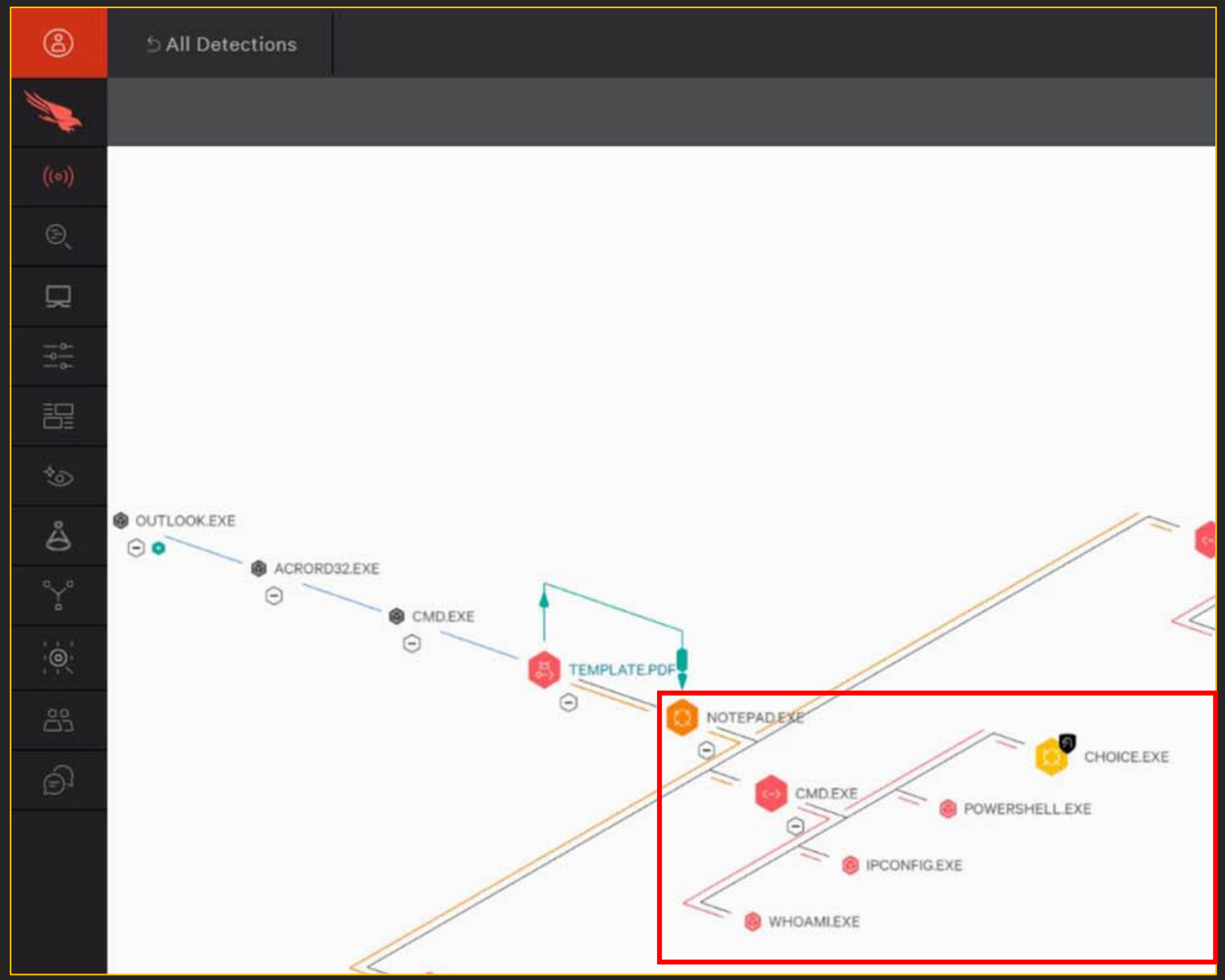
```
meterpreter > shell
Process 5356 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop>whoami
whoami
cap\administrator

C:\Users\Administrator\Desktop>ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter CAP:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b529:79ad:5e98:e5e7%13
   IPv4 Address. . . . . . . . . . . : 10.11.3.5
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.11.3.1

C:\Users\Administrator\Desktop>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

But suddenly...

```
meterpreter >
meterpreter >
meterpreter >
meterpreter > getuid
[-] Error running command getuid: Rex::TimeoutError Operation timed out.
meterpreter > sysinfo
[-] Error running command sysinfo: Rex::TimeoutError Operation timed out.
meterpreter >
meterpreter > Oh no...
```

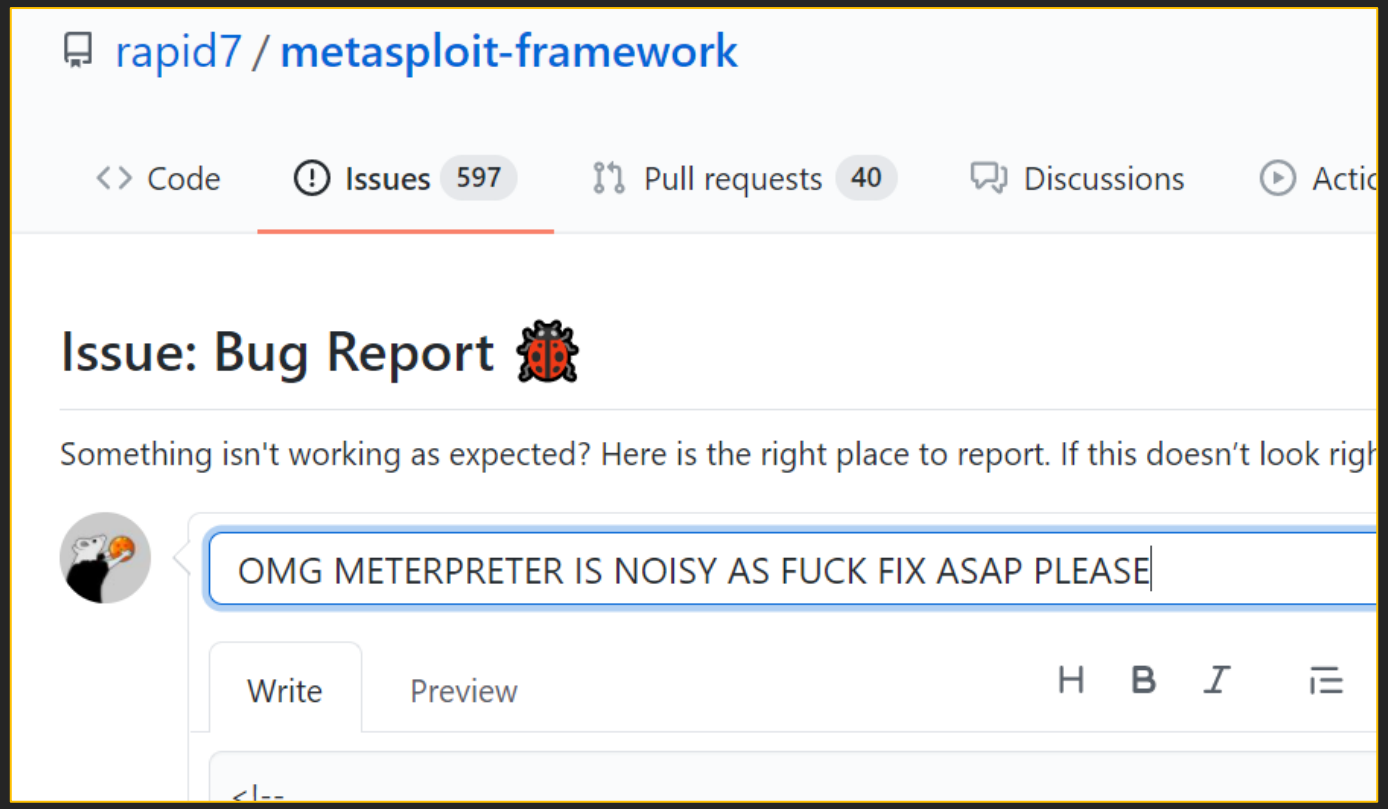
Charles' response to this:





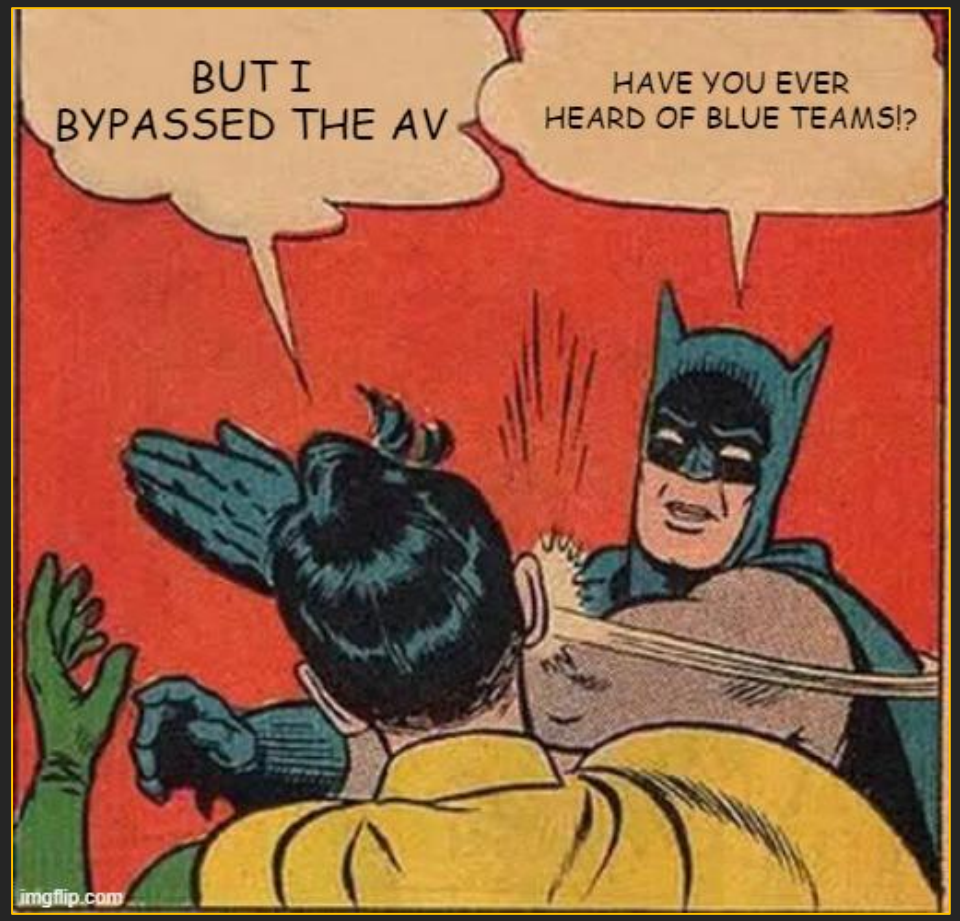
https://www.csoonline.com/article/3404460/review-crowdstrike-falcon-breaks-the-edr-mold.html

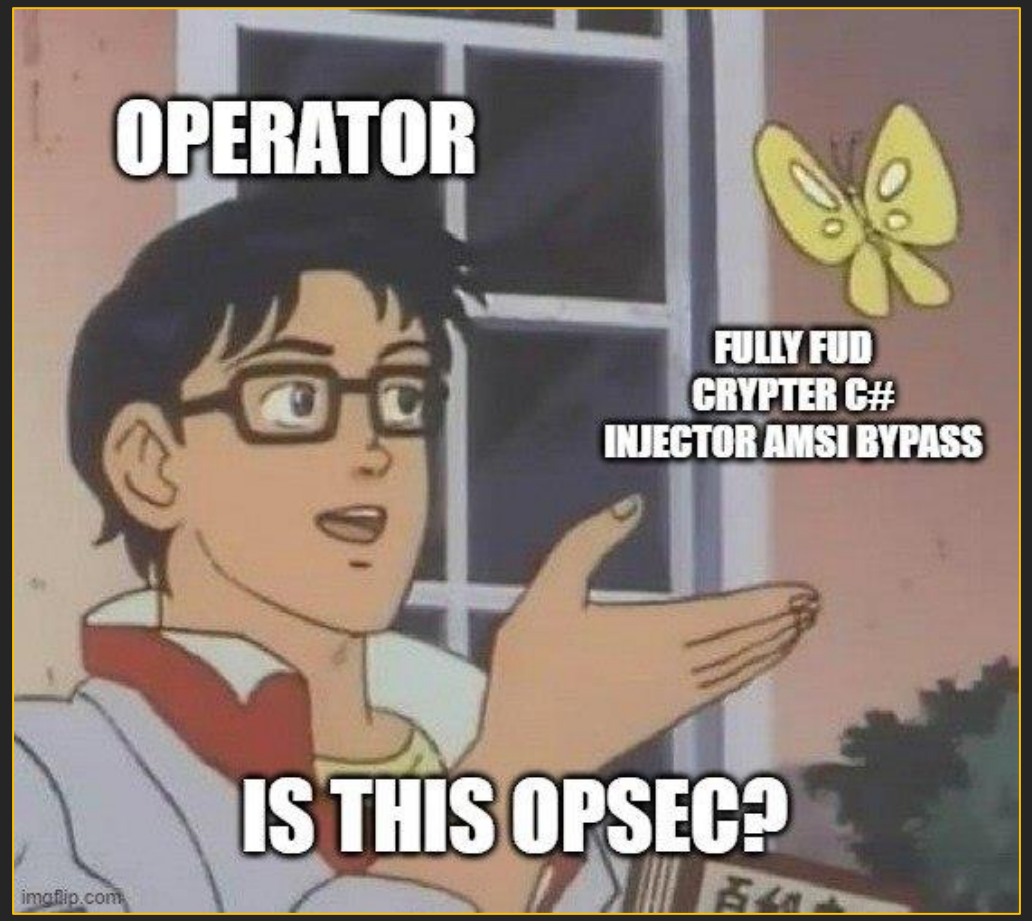
Once Charles got informed about this, he learnt something



...well, or maybe not

OPSEC… What’s This?


BUT I BYPASSED THE AV
HAVE YOU EVER HEARD OF BLUE TEAMS!?
imgflip.com


OPERATOR
FULLY FUD
CRYPTER C#
INJECTOR AMSI BYPASS
IS THIS OPSEC?
imgflip.com

Operational Security (OPSEC)

You may be thinking

- Avoiding too much “noise”
- AV / EDR evasion
- Using legitimate / built-in tools instead of malware

- In fact, when we talk about OPSEC the scope is usually wider

<u>Identification and protection of data that could be useful for an adversary</u>

Robbing a Bank (Easy & Fast)

1. Do not get detected
2. Do not raise suspicions
3. Do not leave forensic traces

Robbing a Bank (Easy & Fast)

Understand your tools

Understand your procedures

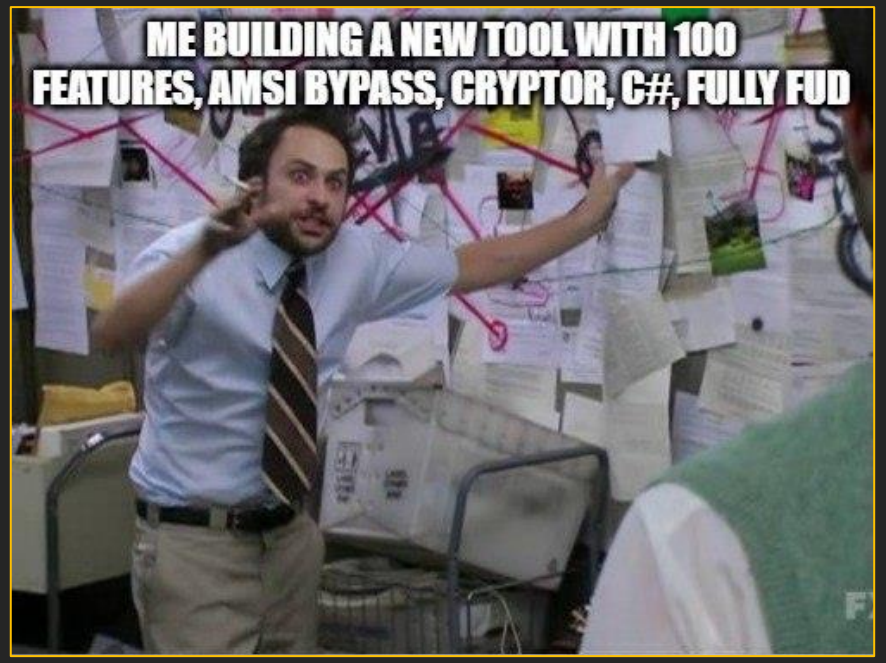
Understand your adversary

1 Do not get detected

2 Do not raise suspicions

3 Do not leave forensic traces

Understand your Tools

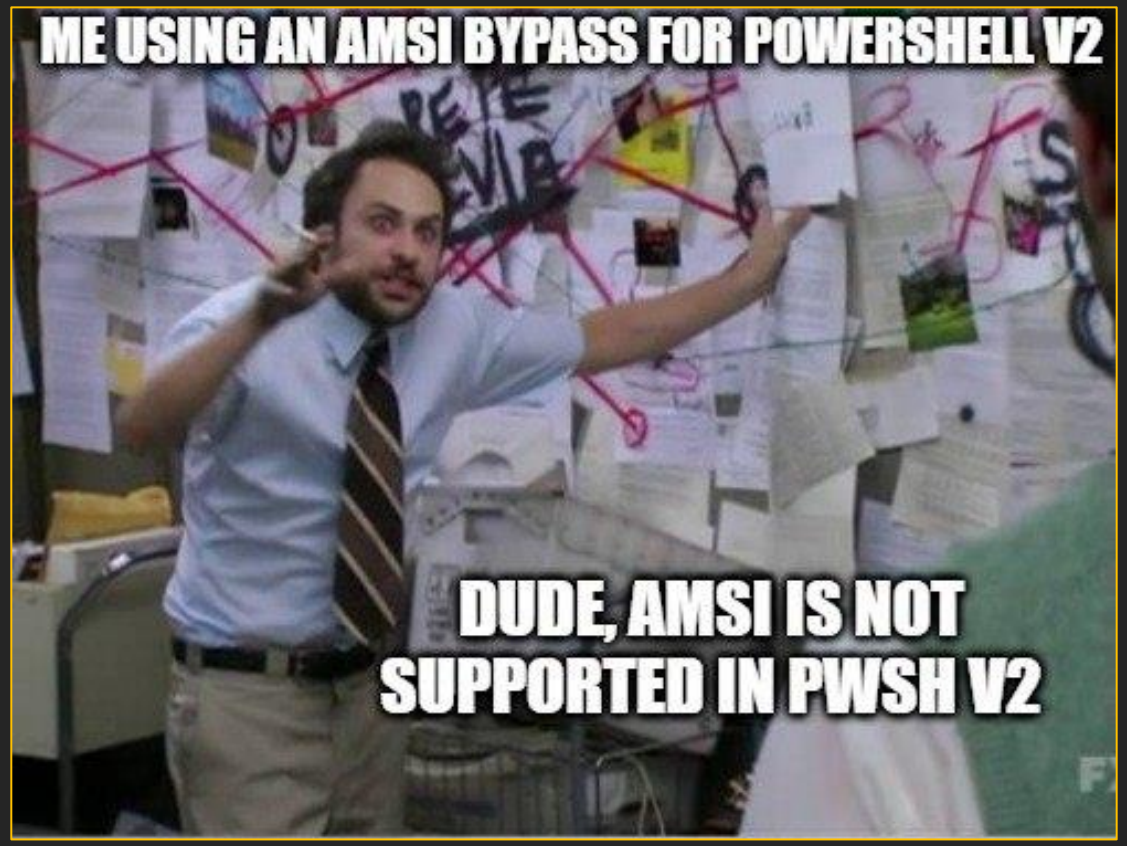


- Does this tool **depend on** executing other binaries? (cmd.exe, wmic.exe, powershell.exe…)
  - Can I implement the same behaviour without executing them?

- Well-known **signatures** or **patterns**?
  - Can I configure or modify them easily?
  - Obfuscation? encryption?

- Do I really need **THIS** tool for **THIS** purpose?
  - Do I really need Mimikatz to dump LSASS?
  - Do I really need SharpHound to check if this user has DCSync rights?
  - Do I really need to create a new service to move laterally to that system?

Understand your Procedures



- There is **no perfect solution** for every-single situation
  - E.g. Sometimes working in memory is the safest place. Other times might be better writing a file to disk
- Obsessing with OPSEC can be a **double-edged sword**
  - There must be a balance between **effort** and **efficiency**
  - Please do not forget we are here to improve Blue Team's detection capabilities!
- How **mature** your adversary is will mark the **minimum security** you'll need in the operation

Become your Adversary

The Adversary

- The organization's security team A.K.A **Blue Team**

- Blue Team **tasks** often include:
  - **Hardening** the environment (patching, logging…)
  - **Monitoring** unusual behaviour (IOCs, suspicious activity…)
  - **Responding** to incidents (system isolation, account lockouts…)
  - **Investigating** the origin of those incidents (forensic traces, artifacts…)

- Understanding these as an attacker will help us in terms of OPSEC

Data is Key

- The **lowest common denominator** of most Blue Team activities is **data**

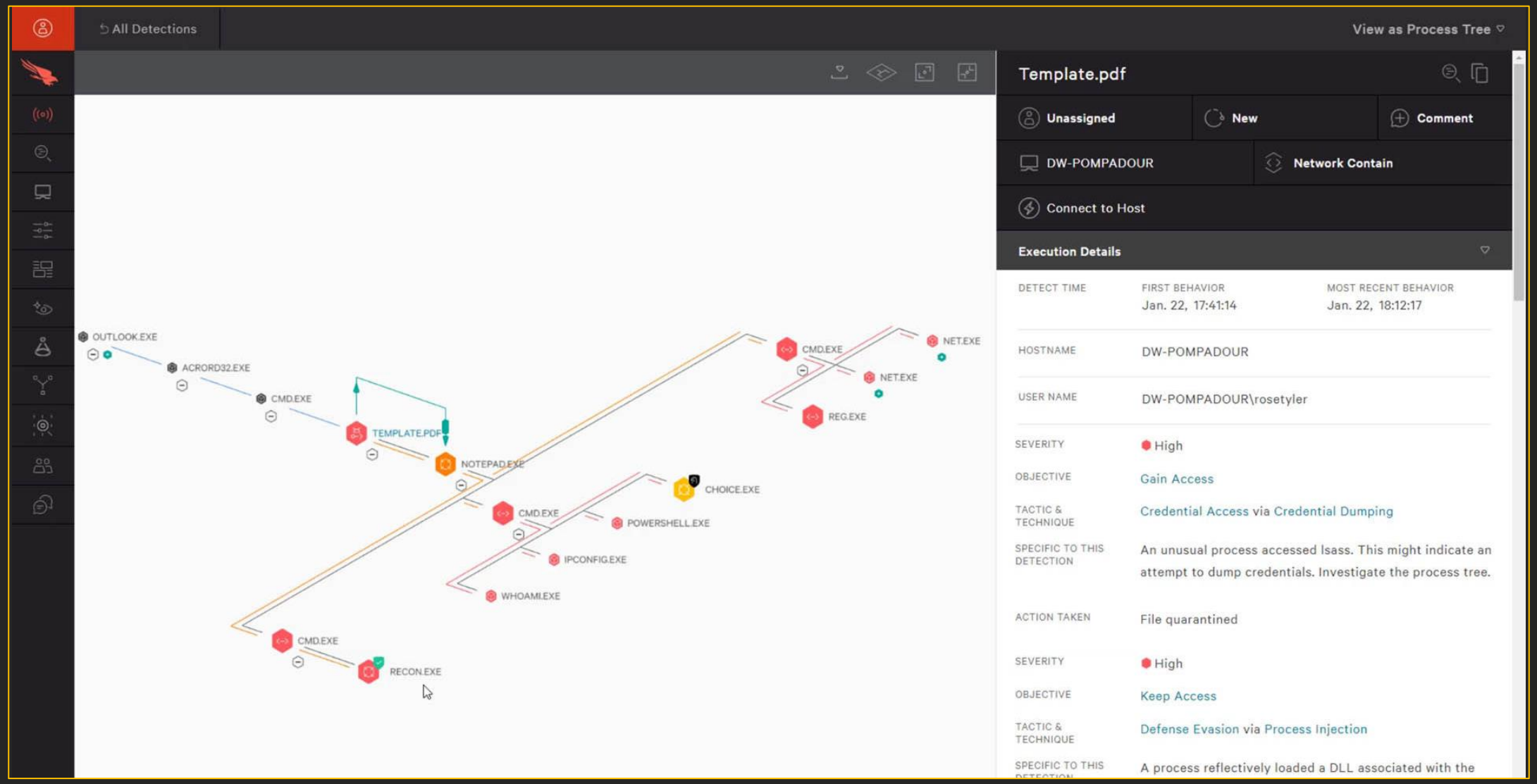
- Defenders **depend on** the data belonging to the assets they want to protect
  - How do you protect something if you don’t know what’s happening there?

- This data **should be relevant for its purposes** (data quality)

  "Nowadays, while most organizations are great at collecting data, they usually do not manage it well to make sense of it" – Roberto Rodriguez (@Cyb3rWard0g)

- It is used to create **defensive capabilities**
  - Creating **timelines** and data **correlations**
  - Configuring alerts for **suspicious** patterns and behaviours
  - Blocking **well-known** patterns and behaviours
  - …

- This applies to how AVs / EDRs work



https://www.csoonline.com/article/3404460/review-crowdstrike-falcon-breaks-the-edr-mold.html

But…
How is this data obtained?

Data Sources

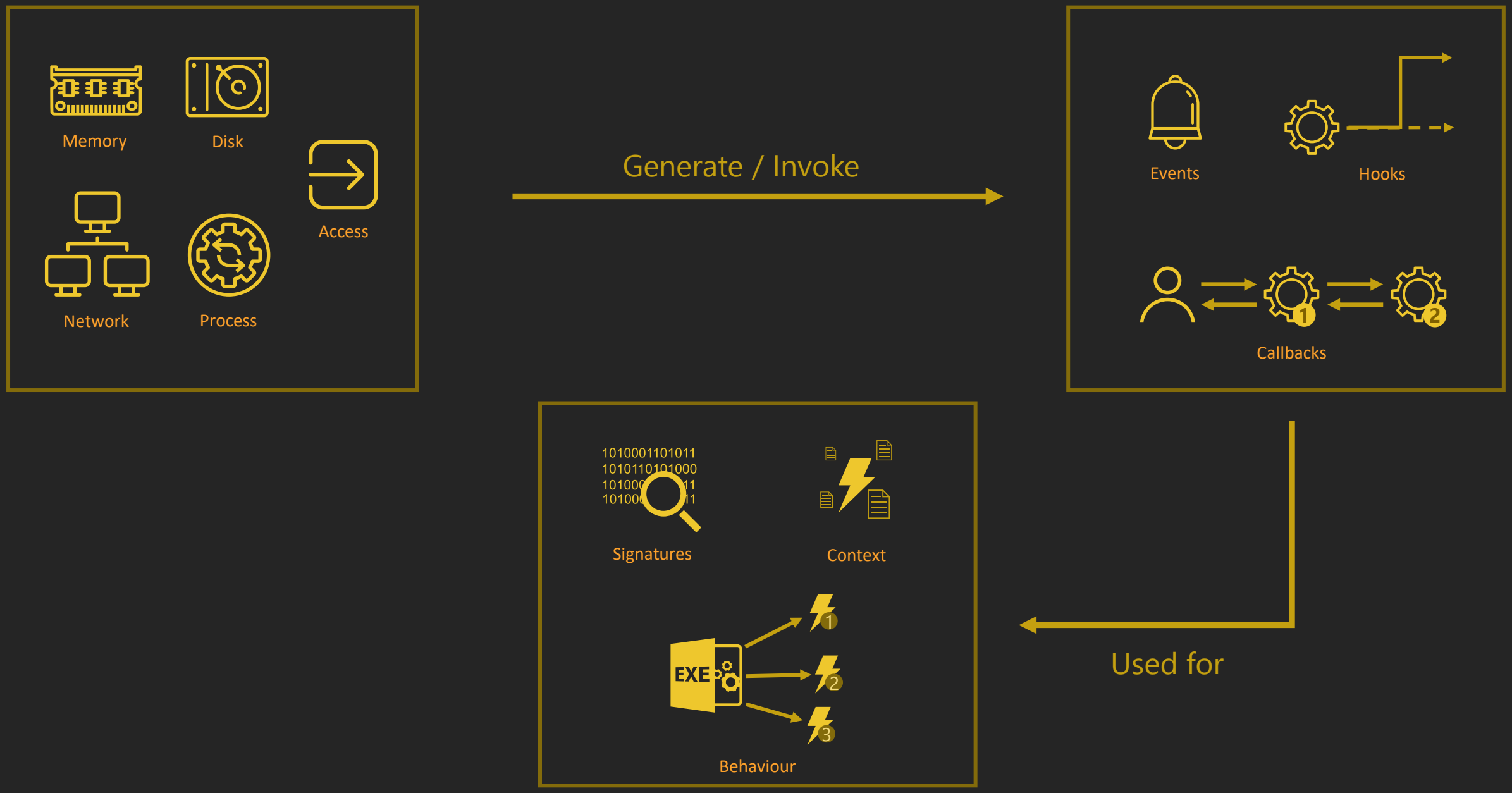
Some Recommended Data Sources

Access Tokens	Detonation chamber	Loaded DLLs	PowerShell logs	VBR
Anti-virus	Digital Certificate Logs	Mail server	Process command-line parameters	Web application firewall logs
API monitoring	DLL monitoring	Malware reverse engineering	Process monitoring	Web logs
Application Logs	DNS records	MBR	Process use of network	Web proxy
Asset Management	EFI	Named Pipes	Sensor health and status	Windows Error Reporting
Authentication logs	Email gateway	Netflow/Enclave netflow	Services	Windows event logs
Binary file metadata	Environment variable	Network device logs	SSL/TLS inspection	Windows Registry
BIOS	File monitoring	Network intrusion detection system	System calls	WMI Objects
Browser extensions	Host network interface	Network protocol analysis	Third-party application logs	
Data loss prevention	Kernel drivers	Packet capture	User interface	

How is this Data Obtained?

- Defenders need mechanisms to **gather the appropriate data** for those systems they want to protect
  - Sysmon
  - EDR agents
  - Logs
  - …

- These mechanisms often leverage features and techniques such as:
  - Event Tracing for Windows (ETW)
  - Callback objects
  - Hooking techniques


Memory
Disk
Access
Network
Process
Generate / Invoke
Events
Hooks
1
2
Callbacks
1010001101011
1010110101000
Signatures
Context
EXE
1
2
3
Behaviour
Used for

ETW, Callbacks n Hooks

ETW, Callbacks n Hooks

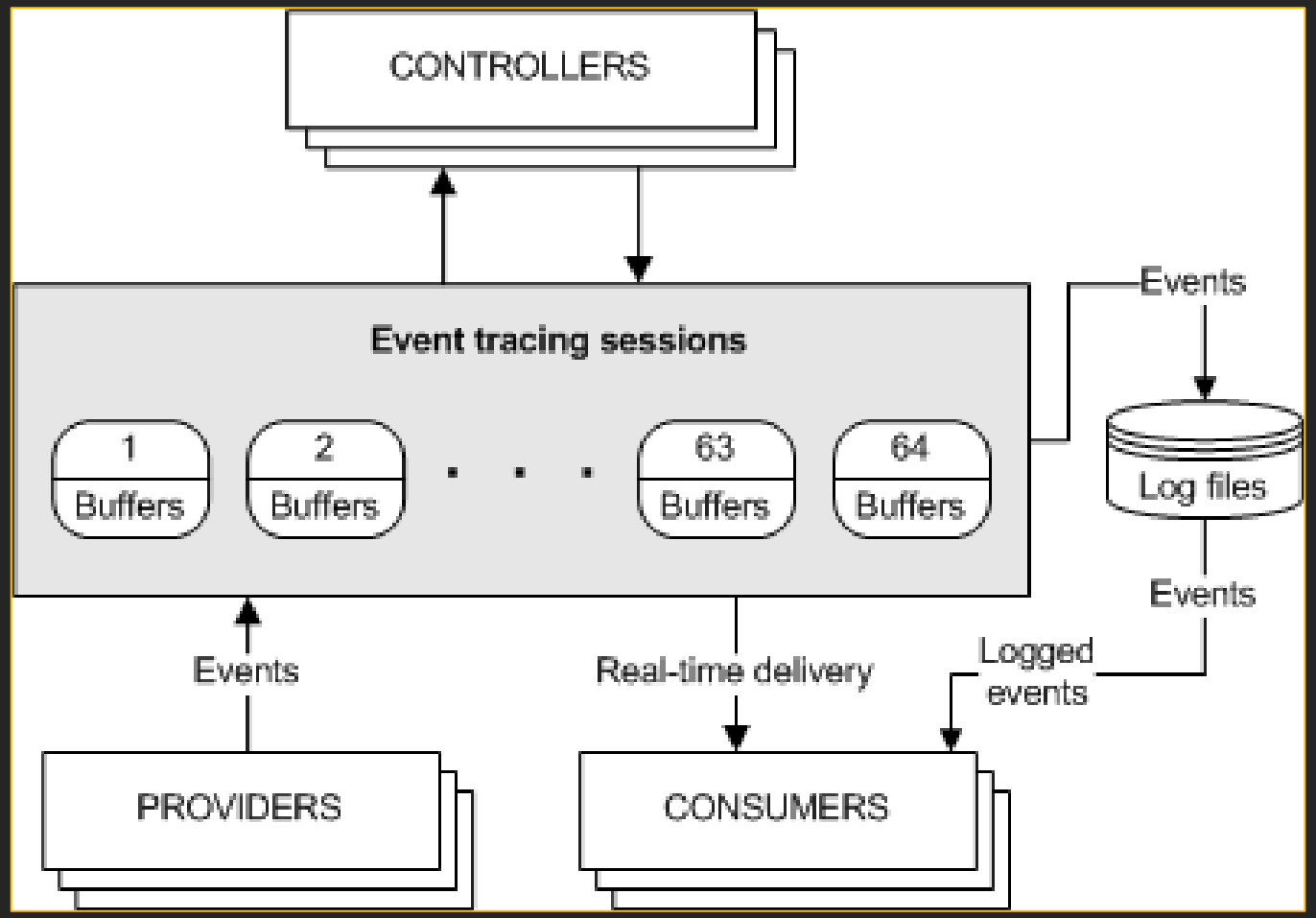
- Not an in-depth explanation. We are going to see a **brief overview** of how these work
- The intend is showing the **amount of information available** to defenders for threat hunting and other activities
- Getting in touch with these will naturally create a **feeling of security awareness** on us

ETW

- Mechanism in Windows to **trace and log system events**
- Think of it as an Event Factory
- ETW is the **data source** for many Blue Team's **alerting and detection strategies**

https://medium.com/palantir/tampering-with-windows-event-tracing-background-offense-and-defense-4be7ac62ac63

- Controllers: start and stop event tracing sessions and enable providers
- Tracing sessions: collect events from providers and serve them to consumers and logs
- Providers: provide events from different components (e.g. PowerShell)
- Consumers: consume events from one or more providers



https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing

- More than **1000 providers** in Windows 10 can offer **huge visibility** for defenders
  - Processes, threads, image loads, network, PowerShell, WMI, WinRM, RDP, Firewall, Defender, .NET…
- For example, Sysmon registers a new ETW provider when you install it
  - Name: Microsoft-Windows-Sysmon
  - Sysmon itself also uses existing ETW providers (e.g. the Windows Kernel Trace) for some of its own events

https://docs.microsoft.com/es-es/archive/blogs/office365security/hidden-treasure-intrusion-detection-with-etw-part-1

Microsoft-Windows-PowerShell provider:

- A command starts/ends
- A Runspace object is constructed
- Runspace connections
- Named pipe usage
- …

Windows PowerShell

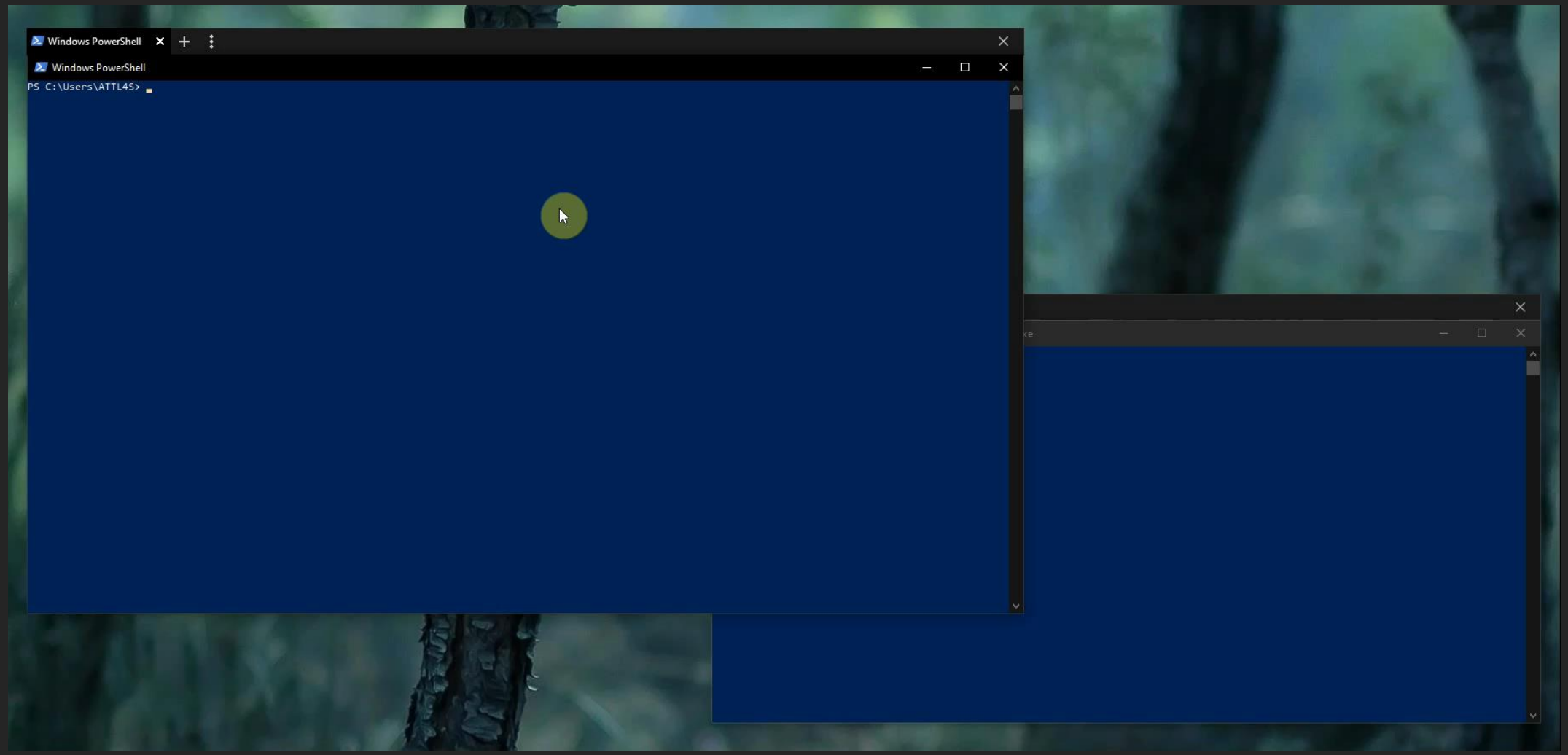
```
PS C:\Users\ATTL4S> (Get-Process -Name powershell).Id
5124
PS C:\Users\ATTL4S>
```

Command Prompt

Command Prompt

```
C:\Users\ATTL4S>logman query providers -pid 5124

Provider                                GUID
-------------------------------------------------------------------------------
.NET Common Language Runtime            {E13C0D23-CCBC-4E12-931B-D9CC2EEE27E4}
Microsoft-Antimalware-Protection        {E4B70372-261F-4C54-8FA6-A5A7914D73DA}
Microsoft-Antimalware-Scan-Interface    {2A576B87-09A7-520E-C21A-4942F0271D67}
Microsoft-IEFRAME                       {5C8BB950-959E-4309-8908-67961A1205D5}
Microsoft-Windows-Application-Experience {EEF54E71-0661-422D-9A98-82FD4940B820}
Microsoft-Windows-AppModel-Runtime      {F1EF270A-0D32-4352-BA52-DBAB41E1D859}
Microsoft-Windows-AsynchronousCausality {19A4C69A-28EB-4D4B-8D94-5F19055A1B5C}
Microsoft-Windows-CAPI2                 {5BBCA4A8-B209-48DC-A8C7-B23D3E5216FB}
Microsoft-Windows-COM-Perf              {B8D6861B-D20F-4EEC-BBAE-87E0DD80602B}
Microsoft-Windows-COM-RundownInstrumentation {2957313D-FCAA-5D4A-2F69-32CE5F0AC44E}
Microsoft-Windows-Crypto-BCrypt         {C7E089AC-BA2A-11E0-9AF7-68384824019B}
Microsoft-Windows-Crypto-RSAEnh         {152FDB2B-6E9D-4B60-B317-815D5F174C4A}
Microsoft-Windows-Deplorch              {B9DA9FE6-AE5F-4F3E-B2FA-8E623C11DC75}
Microsoft-Windows-DeviceManagement-Enterprise-Diagnostics-Provider {3DA494E4-0FE2-415C-B895-FB5265C5C83B}
Microsoft-Windows-DotNETRuntimeRundown  {A669021C-C450-4609-A035-5AF59AF4DF18}
Microsoft-Windows-Eventlog              {FC65DDD8-D6EF-4962-83D5-6E5CFE9CE148}
Microsoft-Windows-Heap-Snapshot         {901D2AFA-4FF6-46D7-8D0E-53645E1A47F5}
Microsoft-Windows-Immersive-Shell       {315A8872-923E-4EA2-9889-33CD4754BF64}
Microsoft-Windows-KnownFolders          {8939299F-2315-4C5C-9B91-ABB86AA0627D}
Microsoft-Windows-Networking-Correlation {83ED54F0-4D48-4E45-B16E-726FFD1FA4AF}
Microsoft-Windows-ntshrui               {676F167F-F72C-446E-A498-EDA43319A5E3}
Microsoft-Windows-OfflineFiles-CscApi   {19EE4CF9-5322-4843-B0D8-BAB81BE4E81E}
Microsoft-Windows-OfflineFiles-CscDclUser {D5418619-C167-44D9-BC36-765BEB5D55F3}
Microsoft-Windows-OfflineFiles-CscFastSync {791CD79C-65B5-48A3-804C-786048994F47}
Microsoft-Windows-OfflineFiles-CscNetApi {361F227C-AA14-4D19-9007-0C8D1A8A541B}
Microsoft-Windows-OfflineFiles-CscService {89D89015-C0DF-414C-BC48-F50E114832BC}
Microsoft-Windows-OfflineFiles-CscUM    {5E23B838-5B71-47E6-B123-6FE02EF573EF}
Microsoft-Windows-PowerShell            {A0C1853B-5C40-4B15-8766-3CF1C58F985A}
Microsoft-Windows-RPC                   {6AD52B32-D609-4BE9-AE07-CE8DAE937E39}
Microsoft-Windows-RPC-Events            {F4AED7C7-A898-4627-B053-44A7CAA12FCD}
```

```
C:\Users\ATTL4S>logman query providers Microsoft-Windows-PowerShell

Provider                                GUID
-------------------------------------------------------------------------------
Microsoft-Windows-PowerShell            {A0C1853B-5C40-4B15-8766-3CF1C58F985A}

Value               Keyword             Description
-------------------------------------------------------------------------------
0x0000000000000001  Runspace            PowerShell Runspace
0x0000000000000002  Pipeline            Pipeline of Commands
0x0000000000000004  Protocol            PowerShell remoting protocol
0x0000000000000008  Transport           PowerShell remoting transport
0x0000000000000010  Host                PowerShell remoting host proxy calls
0x0000000000000020  Cmdlets             All remoting cmdlets
0x0000000000000040  Serializer          The serialization layer
0x0000000000000080  Session             All session layer
0x0000000000000100  Plugin              The managed PowerShell plugin worker
0x0000000000000200  PSWorkflow          PSWorkflow Hosting And Execution Layer
0x0001000000000000  win:ResponseTime    Response Time
0x8000000000000000  Microsoft-Windows-PowerShell/Operational Microsoft-Windows-PowerShell/Operational
0x4000000000000000  Microsoft-Windows-PowerShell/Analytic Microsoft-Windows-PowerShell/Analytic
0x2000000000000000  Microsoft-Windows-PowerShell/Debug Microsoft-Windows-PowerShell/Debug
0x1000000000000000  Microsoft-Windows-PowerShell/Admin Microsoft-Windows-PowerShell/Admin

Value               Level               Description
-------------------------------------------------------------------------------
0x02                win:Error           Error
0x03                win:Warning         Warning
0x04                win:Informational   Information
0x05                win:Verbose         Verbose
0x14                Debug               Debug level defined by PowerShell (which is above Informational defined by system)

PID                 Image
-------------------------------------------------------------------------------
0x00001404          C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

The command completed successfully.

C:\Users\ATTL4S>
```


Windows PowerShell
Windows PowerShell
PS C:\Users\ATTL4S>

Callback Functions

- A callback is a function **invoked within another function**, to complete some kind of routine or action

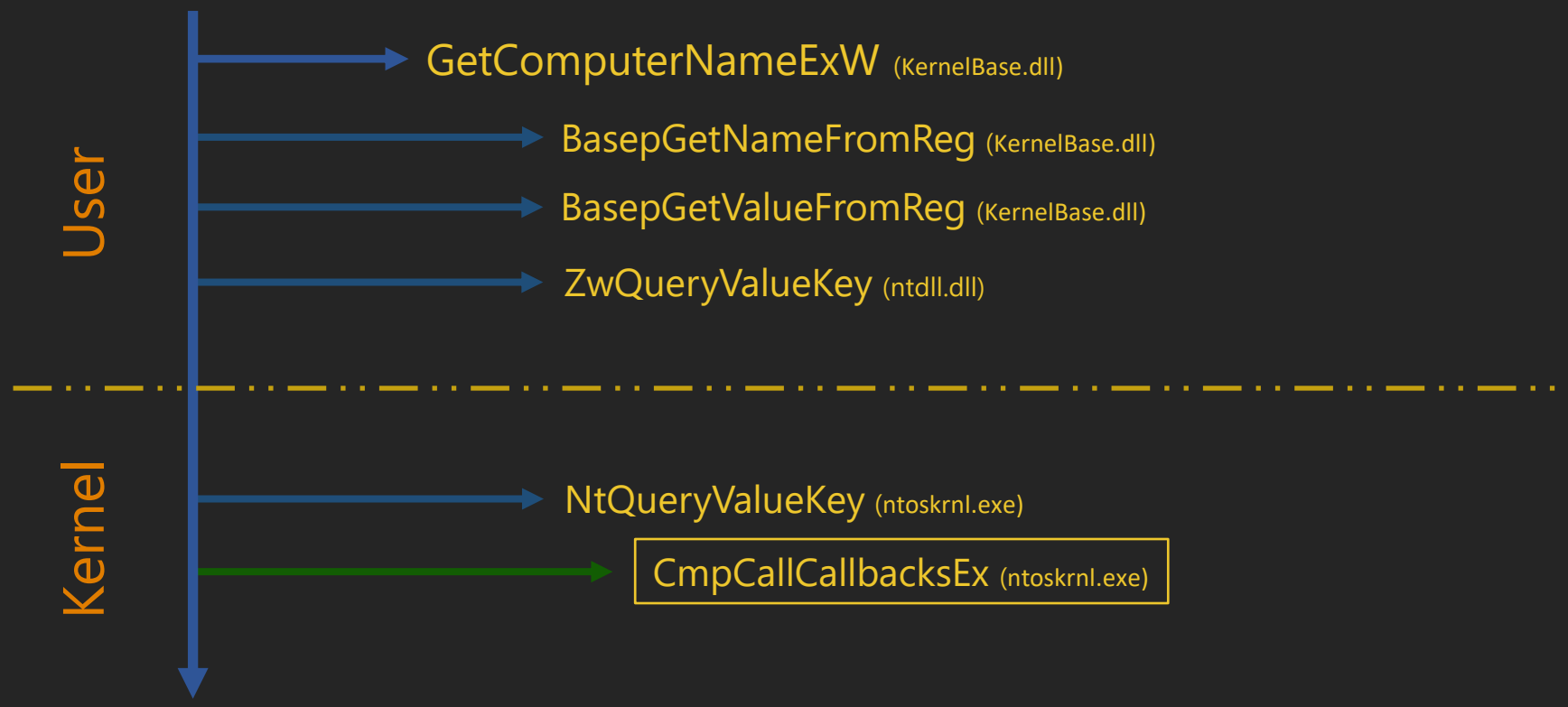
```c
// A simple C program to demonstrate callback
#include<stdio.h>

void A()
{
    printf("I am function A\n");
}

// callback function
void B(void (*ptr)())
{
    (*ptr) (); // callback to A
}

int main()
{
    void (*ptr)() = &A;

    // calling function B and passing
    // address of the function A as argument
    B(ptr);

   return 0;
}
```

Kernel Callbacks

- Windows has a **callback mechanism** in the Kernel
  - One of the Microsoft's responses to prevent Kernel hooking
- This mechanism provides a way for drivers to **receive notifications when certain conditions are satisfied**
  - Drivers can define **callback objects** with a name and a set of attributes
  - Drivers can register **callback routines** for those callback objects
- When conditions are met for a callback, the System calls all the routines registered in it
  - Pre-operation callbacks
  - Post-operation callbacks
- User-mode callbacks not as heavily used as kernel ones

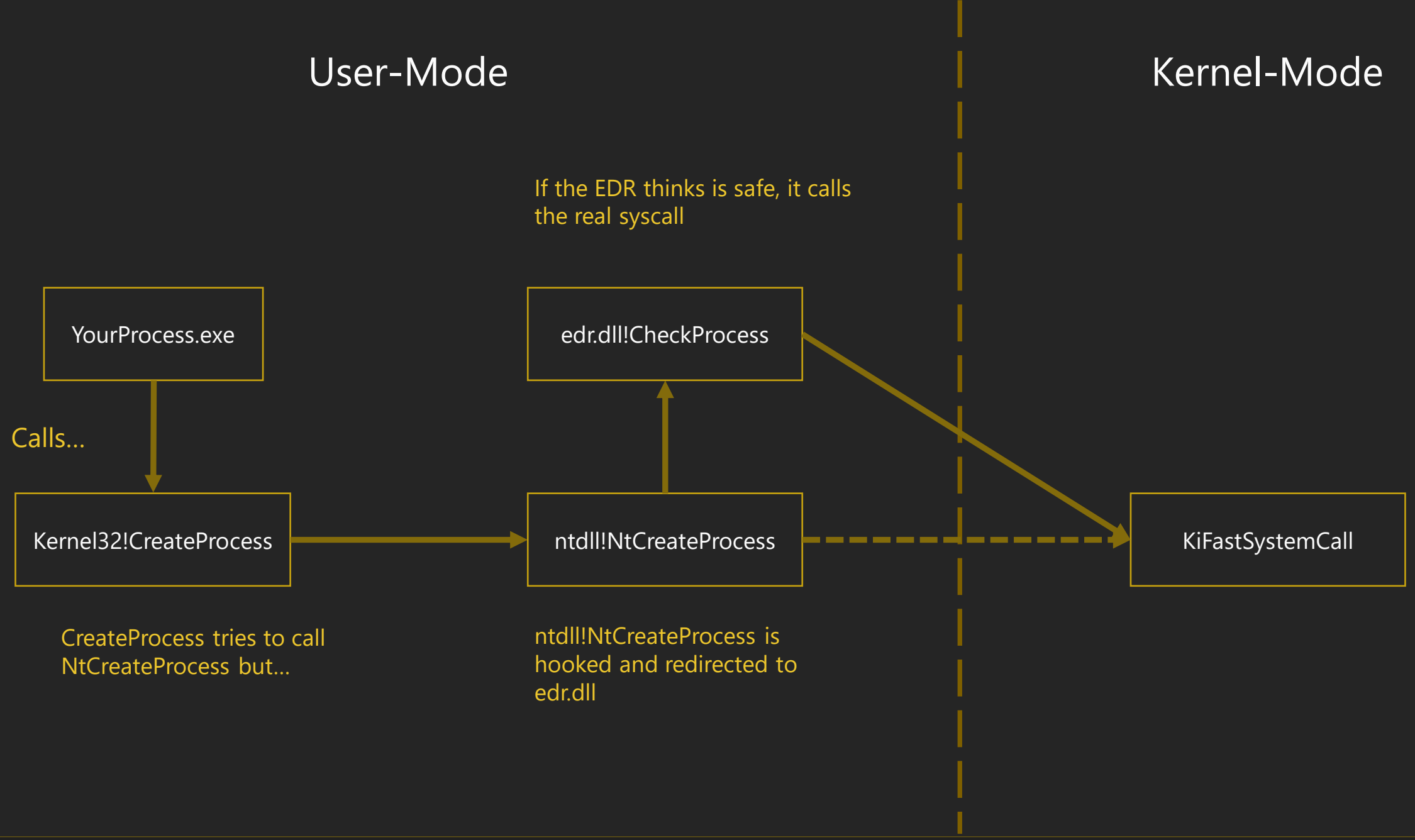
https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/callback-objects

- Callbacks can be used to obtain knowledge or carry out actions when certain conditions are met
  - Object Callbacks: associated to objects such as processes, threads or desktops (e.g. process creations or deletions)
  - Registry Callbacks: associated to the registry (e.g. modifying or creating hives / keys)
  - Filesystem Callbacks (Mini-Filters): associated to interactions with the NTFS filesystem (e.g. creating or deleting a file)
  - …


User
GetComputerNameExW (KernelBase.dll)
BasepGetNameFromReg (KernelBase.dll)
BasepGetValueFromReg (KernelBase.dll)
ZwQueryValueKey (ntdll.dll)
Kernel
NtQueryValueKey (ntoskrnl.exe)
CmpCallCallbacksEx (ntoskrnl.exe)

Hooking

- Technique used to **alter** a process' execution flow and behaviour
- Kernel hooking is **restricted in 64-bit** thanks to **Kernel Patch Protection (KPP)**
  - Anti-Rootkit Measurement
  - Microsoft implemented the Kernel callback mechanism along with ETW modifications as alternatives
- For user-mode hookings, there are a lot of approaches
  - Inline hooks
  - IAT/EAT hooks
  - ...
- EDR/AV agents are widely-known for performing user-mode hookings


User-Mode
Kernel-Mode
If the EDR thinks is safe, it calls the real syscall
YourProcess.exe
edr.dll!CheckProcess
Calls...
Kernel32!CreateProcess
ntdll!NtCreateProcess
KiFastSystemCall
CreateProcess tries to call NtCreateProcess but...
ntdll!NtCreateProcess is hooked and redirected to edr.dll

Sum Up

- Defenders need different **mechanisms** to achieve wider visibility
  - These mechanisms are likely leveraging things like **ETW, kernel callbacks** and **user-land hooks**
  - They can therefore provide **huge visibility** of what is happening in a system

- As an attacker, it is important to **be aware of these mechanisms** so as to adapt our operations properly

- <u>Remember</u>

  "How mature your adversary is will mark the minimum security you'll need in the operation"

Facing a Mature Adversary

Facing a Mature Adversary

- The following slides will show different things a mature Blue Team will be looking nowadays
- The idea is understanding the impact a mature Blue Team can have through different examples
- Areas that will be covered
  - Disk indicators
  - Memory indicators
  - Process indicators
  - Network indicators

Disk Indicators

Defensive mechanisms (and defenders) are often actively looking for **new and modified files**

- Mini-Filters
- ETW (e.g. Microsoft-Windows-Kernel-File provider)

Common indicators they will be looking in those files

- Are there any **public signatures** associated to malware?
- Offensive-related **strings** within the file? (e.g. common patterns or names)
- Offensive-related API functions in the **import table**? (e.g. VirtualAlloc, VirtualProtect…)
- Valid **signature** from a **trusted entity**?
- **Location** where those files were dropped?

Rule Content

```yaml
- title: Detection of SafetyKatz
  id: e074832a-eada-4fd7-94a1-10642b130e16
  status: experimental
  description: Detects possible SafetyKatz Behaviour
  references:
  - https://github.com/GhostPack/SafetyKatz
  tags:
  - attack.credential_access
  - attack.t1003
  author: Markus Neis
  date: 2018/07/24
  logsource:
    product: windows
    service: sysmon
    category: null
  detection:
    selection:
      EventID: 11
      TargetFilename: '*\Temp\debug.bin'
    condition: selection
  falsepositives:
  - Unknown
  level: high
```

https://notebook.community/Cyb3rWard0g/HELK/docker/helk-jupyter/notebooks/sigma/sysmon_ghostpack_safetykatz

Rule Content

```yaml
- title: Suspicious File Characteristics due to Missing Fields
  id: 9637e8a5-7131-4f7f-bdc7-2b05d8670c43
  description: Detects Executables without FileVersion,Description,Product,Company
    likely created with py2exe
  status: experimental
  references:
  - https://securelist.com/muddywater/88059/
  - https://www.virustotal.com/#/file/276a765a10f98cda1a38d3a31e7483585ca3722ecad19d784441293acf1b7beb/detection
  author: Markus Neis
  date: 2018/11/22
  modified: 2019/11/09
  tags:
  - attack.defense_evasion
  - attack.execution
  - attack.t1064
  logsource:
    product: windows
    service: sysmon
    category: null
  detection:
    selection1:
      Description: \?
      FileVersion: \?
    selection2:
      Description: \?
      Product: \?
    selection3:
      Description: \?
      Company: \?
    condition: 1 of them
  fields:
  - CommandLine
  - ParentCommandLine
  falsepositives:
  - Unknown
  level: medium
```

https://github.com/NVISO-BE/sigma-public/blob/master/rules/windows/sysmon/sysmon_susp_file_characteristics.yml

OPSEC

- Your file drops should appear legit!
  - Drop location
  - Filename
  - Import Table
  - Description
  - Company
  - …
- Want to avoid signatures and suspicious strings?
  - Modify your code or use some kind of obfuscation / encryption

Interesting Links

- Hashing vs. Encryption vs. Encoding vs. Obfuscation
  - https://danielmiessler.com/study/encoding-encryption-hashing-obfuscation/
- Engineering antivirus evasion
  - https://blog.scrt.ch/2020/06/19/engineering-antivirus-evasion/
- HELK - SIGMA rules
  - https://github.com/Cyb3rWard0g/HELK/tree/46f3f984466bec09380cc4cb65dbfec8af567a3a/docker/helk-jupyter/notebooks/sigma
- Tracking Malware with Import Hashing
  - https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html
- Using Custom Covenant Listener Profiles & Grunt Templates to Elude AV
  - https://offensivedefence.co.uk/posts/covenant-profiles-templates/
- Cobalt Strike - The Artifact Kit
  - https://www.youtube.com/watch?v=yWVzTooJGTo

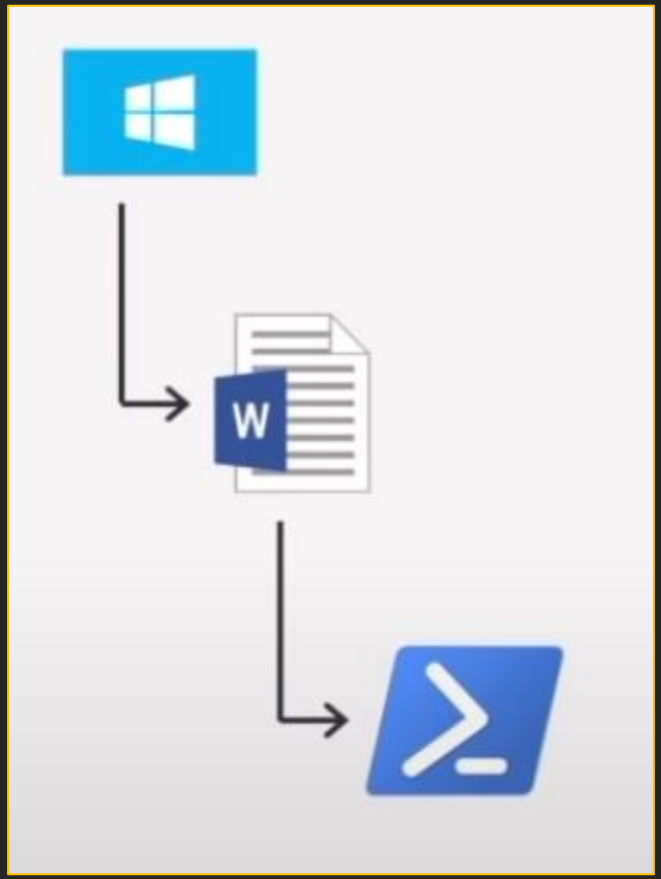
Process Indicators

Defensive mechanisms (and defenders) are often actively looking for **new processes and their behaviour**

- Kernel Callbacks
  - PsSetCreateProcessNotifyRoutine, PsSetCreateThreadNotifyRoutine…
- ETW
  - Microsoft-Windows-Kernel-Process, EventLog-Microsoft-Windows-Sysmon-Operational, Microsoft-Windows-Threat-Intelligence…
- User-land hooks

Common indicators they will be looking in processes

- Parent / child relationships
  - Word -> PowerShell
- Processes never seen in the system
  - Legitimate tools used by malware (LOLBAS)
- Suspicious command-line arguments
  - wmic process call create “powershell.exe -enc …”
- Suspicious API usage
  - Process injection, .NET CLR reflections…
- Processes accessing other key processes
  - LSASS


W

Rule Content

```yaml
- title: Suspicious WMI execution
  id: 526be59f-a573-4eea-b5f7-f0973207634d
  status: experimental
  description: Detects WMI executing suspicious commands
  references:
  - https://digital-forensics.sans.org/blog/2010/06/04/wmic-draft/
  - https://www.hybrid-analysis.com/sample/4be06ecd234e2110bd615649fe4a6fa95403979acf889d7e45a78985eb50acf9?environmentId=1
  - https://blog.malwarebytes.com/threat-analysis/2016/04/rokku-ransomware/
  author: Michael Haag, Florian Roth, juju4
  logsource:
    category: process_creation
    product: windows
    service: null
  detection:
    selection:
      Image:
      - '*\wmic.exe'
      CommandLine:
      - '*/NODE:*process call create *'
      - '* path AntiVirusProduct get *'
      - '* path FirewallProduct get *'
      - '* shadowcopy delete *'
    condition: selection
  fields:
  - CommandLine
  - ParentCommandLine
  tags:
  - attack.execution
  - attack.t1047
  - car.2016-03-002
  falsepositives:
  - Will need to be tuned
  - If using Splunk, I recommend | stats count by Computer,CommandLine following for
    easy hunting by Computer/CommandLine.
  level: medium
```

https://github.com/Neo23x0/sigma/blob/master/rules/windows/process_creation/win_susp_wmi_execution.yml

Administrator: Command Prompt - SilkETW.exe -t user -pn Microsoft-Windows-DotNETRuntime -uk 0x2038 -l verbose -y C:\Users\b...

```
C:\Users\b33f\Tools\SilkETW>SilkETW.exe -t user -pn Microsoft-Windows-DotNETRuntime -uk 0x2038 -l verbose -y C:\Users\b33f\
Desktop\yara -yo matches -ot file -p C:\Users\b33f\Desktop\yara.json

SILKETW

[v0.4 - Ruben Boonen => @FuzzySec]

[+] Collector parameter validation success..
[>] Starting trace collector (Ctrl-c to stop)..
[?] Events captured: 61695
     -> Yara match: Seatbelt_GetTokenInformation
```

```
beacon> execute-assembly C:\Users\b33f\Tools\GhostPack\Seatbelt.exe BasicOSInfo
[*] Tasked beacon to run .NET program: Seatbelt.exe BasicOSInfo
[+] host called home, sent: 265793 bytes
[+] received output:

%&&@@@&&
[DESKTOP-GALB1P7] b33f/16364 (x64)                                  last: 648ms
beacon>
```

Rule Content

```
- title: Mimikatz Detection LSASS Access
  id: 0d894093-71bc-43c3-8c4d-ecfc28dcf5d9
  status: experimental
  description: Detects process access to LSASS which is typical for Mimikatz (0x1000
    PROCESS_QUERY_ LIMITED_INFORMATION, 0x0400 PROCESS_QUERY_ INFORMATION "only old
    versions", 0x0010 PROCESS_VM_READ)
  references:
  - https://onedrive.live.com/view.aspx?resid=D026B4699190F1E6!2843&ithint=file%2cpptx&app=PowerPoint&authkey=!AMvCRTKB_V1J5ow
  - https://cyberwardog.blogspot.com/2017/03/chronicles-of-threat-hunter-hunting-for_22.html
  tags:
  - attack.t1003
  - attack.s0002
  - attack.credential_access
  - car.2019-04-004
  logsource:
    product: windows
    service: sysmon
    category: null
  detection:
    selection:
      EventID: 10
      TargetImage: C:\windows\system32\lsass.exe
      GrantedAccess:
      - '0x1410'
      - '0x1010'
    condition: selection
  falsepositives:
  - unknown
  level: high
```

OPSEC

- Avoid creating new processes as much as possible
  - Can you execute your capability within your process? Local injections might help
  - PPID spoofing might help for parent/child relationships
- Try to blend in – avoid weird behaviours as possible
  - Your process needs Internet? Try working in the context of a process that does this
  - Avoid common offensive patterns
  - Avoid remote code injections as possible
- Command-line arguments?
  - Command-line spoofing might help

Interesting Links

- Will Burgess - Red Teaming in the EDR age
  - https://www.youtube.com/watch?v=l8nkXCOYQC4
- Securi-Tay 2017 - Advanced Attack Detection
  - https://www.youtube.com/watch?v=ihElrBBJQo8
- Raphael Mudge - Session Prepping and Session Passing
  - https://www.youtube.com/watch?v=4xnBn5ZVkKE
- Adam Chester – How to Argue like Cobalt Strike
  - https://blog.xpnsec.com/how-to-argue-like-cobalt-strike/
- ired.team - Parent Process ID (PPID) Spoofing
  - https://www.ired.team/offensive-security/defense-evasion/parent-process-id-ppid-spoofing

Memory Indicators

Defensive mechanisms (and defenders) are often actively looking for **suspicious activity in memory** through memory scans

Common indicators they will be looking in memory:

- PE files in memory not associated with a module on disk
  - MZ header, “This program cannot be run…”
- Module-less threads (A.K.A injected threads)
  - The start address of the thread points to a location with no module associated
- Suspicious memory permissions such as RWX
- Malware-associated strings
- In-memory vs on-disk comparisons
  - Process hollowings, dll hollowings…

dc01.capsule.corp | Strobe | fw01.capsule.corp

Activities | Terminal | Dec 5 02:30

attl4s@ubuntu: ~

```
msf6 exploit(multi/handler) > run -j
[*] Exploit running as background job 4.
[*] Exploit completed, but no session was created.

[*] Started HTTP reverse handler on http://10.11.1.130:4444
msf6 exploit(multi/handler) >
```

OPSEC

- Your thread's start address points to a location with no module associated?
  - Update the address to a more convenient location (e.g. using SetThreadContext)
- Memory injected PE files with no module associated?
  - Module stomping might help
- Avoid stagers!
  - Stagers require several requirements that will reduce your OPSEC. Use stageless payloads
- Avoid RWX memory permissions
- Watchout your memory contents
  - PE headers, sneaky strings…
- Obfuscate memory when it is not in use

Interesting Links

- The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory

- Taking Hunting to the Next Level: Hunting in Memory - SANS Threat Hunting Summit 2017
  - https://www.youtube.com/watch?v=EVBCoV8lpWc

- Adam Chester - Understanding and Evading Get-InjectedThread
  - https://blog.xpnsec.com/undersanding-and-evading-get-injectedthread/

- Raphael Mudge - In-memory Evasion
  - https://www.youtube.com/playlist?list=PL9HO6M_MU2nc5Q31qd2CwpZ8J4KFMhgnK

- Cobalt Strike – Malleable PE, Process Injection, and Post Exploitation
  - https://www.cobaltstrike.com/help-malleable-postex

- Elastic - Hunting In Memory
  - https://www.elastic.co/es/blog/hunting-memory

- Bypassing Memory Scanners with Cobalt Strike and Gargoyle
  - https://labs.f-secure.com/blog/experimenting-bypassing-memory-scanners-with-cobalt-strike-and-gargoyle/

Network Indicators

Defensive mechanisms (and defenders) are often actively looking for **suspicious network activity** within systems

- ETW
  - Microsoft-Windows-Winsock-AFD, Microsoft-Windows-TCPIP…
- Callbacks
  - WskAcceptEvent, WskReceiveEvent…
- IDS/IPS solutions, WAFs, corporate proxies…

- Common indicators they will be looking and doing
  - Traffic inspection
    - SSL/TLS inspection
  - Domains and IPs accessed
    - Domain categorization? Cert information? Weird names?
  - Amount of traffic
  - Processes beaconing
    - Fixed times

Please type in a URL to look up the categorization.

http://nccgroup.com

Check URL

Categorization in URL Filter database version '388990'

	URL	Status	Categorization	Reputation
	http://nccgroup.com	Categorized URL	- Business	Minimal Risk

Rule Content

```yaml
- title: CobaltStrike Malleable Amazon browsing traffic profile
  id: 953b895e-5cc9-454b-b183-7f3db555452e
  status: experimental
  description: Detects Malleable Amazon Profile
  references:
  - https://github.com/rsmudge/Malleable-C2-Profiles/blob/master/normal/amazon.profile
  - https://www.hybrid-analysis.com/sample/ee5eca8648e45e2fea9dac0d920ef1a1792d8690c41ee7f20343de1927cc88b9?environmentId=100
  author: Markus Neis
  tags:
  - attack.t1102
  logsource:
    category: proxy
    product: null
    service: null
  detection:
    selection1:
      c-useragent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like
        Gecko
      cs-method: GET
      c-uri: /s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books
      cs-host: www.amazon.com
      cs-cookie: '*=csm-hit=s-24KU11BB82RZSYGJ3BDK|1419899012996'
    selection2:
      c-useragent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like
        Gecko
      cs-method: POST
      c-uri: /N4215/adj/amzn.us.sr.aps
      cs-host: www.amazon.com
    condition: selection1 or selection2
  falsepositives:
  - Unknown
  level: high
```

https://github.com/Neo23x0/sigma/blob/master/rules/proxy/proxy_cobalt_amazon.yml

OPSEC

- Your traffic should look legit!
  - Try to impersonate common services that are reaching the Internet
  - C2's like Covenant or Cobalt Strike have configurable network profiles
- Cook your domains over low heat
  - Domain categorization, domain age, domain names…
- Route your traffic through high-trust domains
  - Domain fronting
- Take care of your C2 infra
  - Use redirectors between your backend and your targets
- Agent beaconing?
  - Configure proper delay and jitter percentages!

Interesting Links

- SSL/TLS Interception Challenge from the Shadow to the Light
  - https://www.sans.org/reading-room/whitepapers/covert/ssl-tls-interception-challenge-shadow-light-38870

- Being a Good Domain Shepherd
  - https://posts.specterops.io/being-a-good-domain-shepherd-57754edd955f
  - https://posts.specterops.io/being-a-good-domain-shepherd-part-2-5e8597c3fe63

- Tom Steele - Escape and Evasion Egressing Restricted Networks
  - https://www.optiv.com/explore-optiv-insights/blog/escape-and-evasion-egressing-restricted-networks

- Cobalt Strike – Malleable Command and Control
  - https://www.cobaltstrike.com/help-malleable-c2
  - https://posts.specterops.io/a-deep-dive-into-cobalt-strike-malleable-c2-6660e33b0e0b
  - https://github.com/rsmudge/Malleable-C2-Profiles

- Covenant – Listener Profiles
  - https://github.com/cobbr/Covenant/wiki/Listener-Profiles

- Red-Team-Infrastructure-Wiki
  - https://github.com/bluscreenofjeff/Red-Team-Infrastructure-Wiki

Paths of Execution

- Defensive actions are not usually consequence of a single heuristic, as it would lead to a good amount of **false positives**
- Defenders and defensive products often **require a combination of different heuristics** for alerts and actions
- A good detection will try to **cover an attack capability as a “whole”**, instead of just focusing on specific tools or signatures

- Defensive actions are no e heuristic, as it would lead to a good amount o
- Defenders and defensive ination of different **heuristics** for alerts and
- A good detection will try s a **“whole”**, instead of just focusing on specific

Rule Content

```
- title: Detection of SafetyKatz
  id: e074832a-eada-4fd7-94a1-10642b130e16
  status: experimental
  description: Detects possible SafetyKatz Behaviour
  references:
  - https://github.com/GhostPack/SafetyKatz
  tags:
  - attack.credential_access
  - attack.t1003
  author: Markus Neis
  date: 2018/07/24
  logsource:
    product: windows
    service: sysmon
    category: null
  detection:
    selection:
      EventID: 11
      TargetFilename: '*\Temp\debug.bin'
    condition: selection
  falsepositives:
  - Unknown
  level: high
```

Capability Abstraction

Jared Atkinson's Capability Abstraction is a good example of what a good detection approach may look like:

"The idea is that an attacker's tools are merely an abstraction of their attack capabilities, and detection engineers must understand how to evaluate abstraction while building detection logic" – Jared Atkinson

T1208 - Kerberoasting				
Tool	PowerShell Invoke-Kerberoast	Rubeus kerberoast	Mimikatz kerberos::ask	Rubeus asktgs
Managed Code	.NET KerberosRequestorSecurityToken			
Windows API Function	InitializeSecurityContext		LsaCallAuthenticationPackage	
RPC	4f32adc8-6052-4a04-8701-293ccf2096f0 C:\WINDOWS\SYSTEM32\SspiSrv.dll			
Network Protocol	Kerberos TGS-REQ/REP			

https://posts.specterops.io/capability-abstraction-fbeaeeb26384

OK but… why are we talking about this?

Paths of Execution

- When you execute a tool (e.g. Mimikatz), different things are executed **under the hood**

- Capability Abstraction **decomposes** tools into different **layers of execution**
  - Each layer holds key functionality related to the tool and its main purpose (technique)
  - Each layer also represents each context where the execution flow passes through (unmanaged, managed, userland, kernel, RPC, network...)

- You can think of these as **Paths of Execution**
- As an attacker, you should be aware of your Paths of Execution:
  - Is there any step in my path that might not be essential?
  - Would it be interesting to avoid certain steps by starting the execution from a lower level?
  - Lower level of execution means smaller detection surface?

Domain Users

Process 1 (cmd.exe)

```
meterpreter > shell
Process 6600 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop>net users /domain
net users /domain

User accounts for \\DC01

-------------------------------------------------------------------------------
Adam.Wally               Administrator            bulma
bulma_da                 Elijah.Blakley           Fannie.Eames
Guest                    Herminia.Oliva           Ivan.Davie
krbtgt                   Marcy.Hyatt              Merideth.Bolanos
Mutenroshi               Mutenroshi_wa            Nguyet.Catlett
Phyllis.Biondo           Renita.Lintz             Vegeta
Vegeta_sa
The command completed successfully.
```

Process 2 (net.exe)
Process 3 (net1.exe)

Arguments

Meterpreter.exe -> cmd.exe -> net.exe -> net1.exe

Can we potentially improve this?

But… new reflective DLL

0x170000	Private: Commit	228 kB	RWX
0x420000	Private: Commit	152 kB	RWX
0x730000	Private: Commit	424 kB	RWX
0x2ab0000	Private: Commit	172 kB	RWX

```
meterpreter > load extapi
Loading extension extapi...Success.
meterpreter > adsi_
adsi_computer_enum          adsi_dc_enum                adsi_domain_query           adsi_group_enum             adsi_nested_group_user_enum  adsi_user_enum
meterpreter > adsi_user_enum capsule.corp

capsule.corp Objects
====================

samaccountname    name                        distinguishedname                                                                    description
--------------    ----                        -----------------                                                                    -----------
Adam.Wally        Adam Wally                  CN=Adam Wally,OU=Disabled Users,OU=User Accounts,DC=capsule,DC=corp
Administrator     Administrator               CN=Administrator,CN=Users,DC=capsule,DC=corp                                         Built-in account for administering
DC01$             DC01                        CN=DC01,OU=Domain Controllers,DC=capsule,DC=corp
Elijah.Blakley    Elijah Blakley              CN=Elijah Blakley,OU=Disabled Users,OU=User Accounts,DC=capsule,DC=corp
Fannie.Eames      Fannie Eames                CN=Fannie Eames,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
Guest             Guest                       CN=Guest,CN=Users,DC=capsule,DC=corp                                                 Built-in account for guest access
Herminia.Oliva    Herminia Oliva              CN=Herminia Oliva,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
Ivan.Davie        Ivan Davie                  CN=Ivan Davie,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
Marcy.Hyatt       Marcy Hyatt                 CN=Marcy Hyatt,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
Merideth.Bolanos  Merideth Bolanos            CN=Merideth Bolanos,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
Mutenroshi        Mutenroshi                  CN=Mutenroshi,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
Mutenroshi_wa     Mutenroshi WorkstationAdmin CN=Mutenroshi WorkstationAdmin,OU=Accounts,OU=Tier 2,OU=Admin,DC=capsule,DC=corp
Nguyet.Catlett    Nguyet Catlett              CN=Nguyet Catlett,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
PAW01$            PAW01                       CN=PAW01,OU=Devices,OU=Tier 1,OU=Admin,DC=capsule,DC=corp
Phyllis.Biondo    Phyllis Biondo              CN=Phyllis Biondo,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
Renita.Lintz      Renita Lintz                CN=Renita Lintz,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
SQL01$            SQL01                       CN=SQL01,OU=Database,OU=Tier 1 Servers,DC=capsule,DC=corp
Vegeta            Vegeta                      CN=Vegeta,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
Vegeta_sa         Vegeta ServerAdmin          CN=Vegeta ServerAdmin,OU=Accounts,OU=Tier 1,OU=Admin,DC=capsule,DC=corp
WS01$             WS01                        CN=WS01,OU=Desktops,OU=Workstations,DC=capsule,DC=corp
WS02$             WS02                        CN=WS02,CN=Computers,DC=capsule,DC=corp
bulma             Bulma                       CN=Bulma,OU=Enabled Users,OU=User Accounts,DC=capsule,DC=corp
bulma_da          Bulma DA                    CN=Bulma DA,OU=Accounts,OU=Tier 0,OU=Admin,DC=capsule,DC=corp
```

No new processes, no arguments!

- Studying Paths of Execution is something like performing a **threat modelling of your tools** (where the risk is being detected)

- We want to identify all the **obvious weaknesses** associated to our paths
  - Useless process executions
  - Useless usage of arguments
  - Unnecessary calls
  - …

- But as we may be thinking… weaknesses will not always be obvious
  - It might depend on the context or situation we are operating

We will use **syscalls** as an example of something that can be extremely useful in some cases, but not so much in others

System Calls

- Commonly called **syscalls**, they are the **lowest level** of execution available from user-mode
  - They switch the execution from **user-mode** to **kernel-mode**

- They offer considerable advantages to offensive tooling
  - Avoiding userland hooks, avoiding certain signatures and heuristics detections…
  - We are essentially avoiding multiple steps from our Path of Execution

- As with everything, they also entail certain **OPSEC considerations**

OpenProcess (kernel32.dll)
ZwOpenProcess (ntdll.dll)
NtOpenProcess (ntoskrnl.exe)

CreateFile (kernel32.dll)
ZwCreateFile (ntdll.dll)
NtCreateFile (ntoskrnl.exe)

PssCaptureSnapshot (Kernelbase.dll)
ZwDuplicateObject (ntdll.dll)
NtDuplicateObject (ntoskrnl.exe)

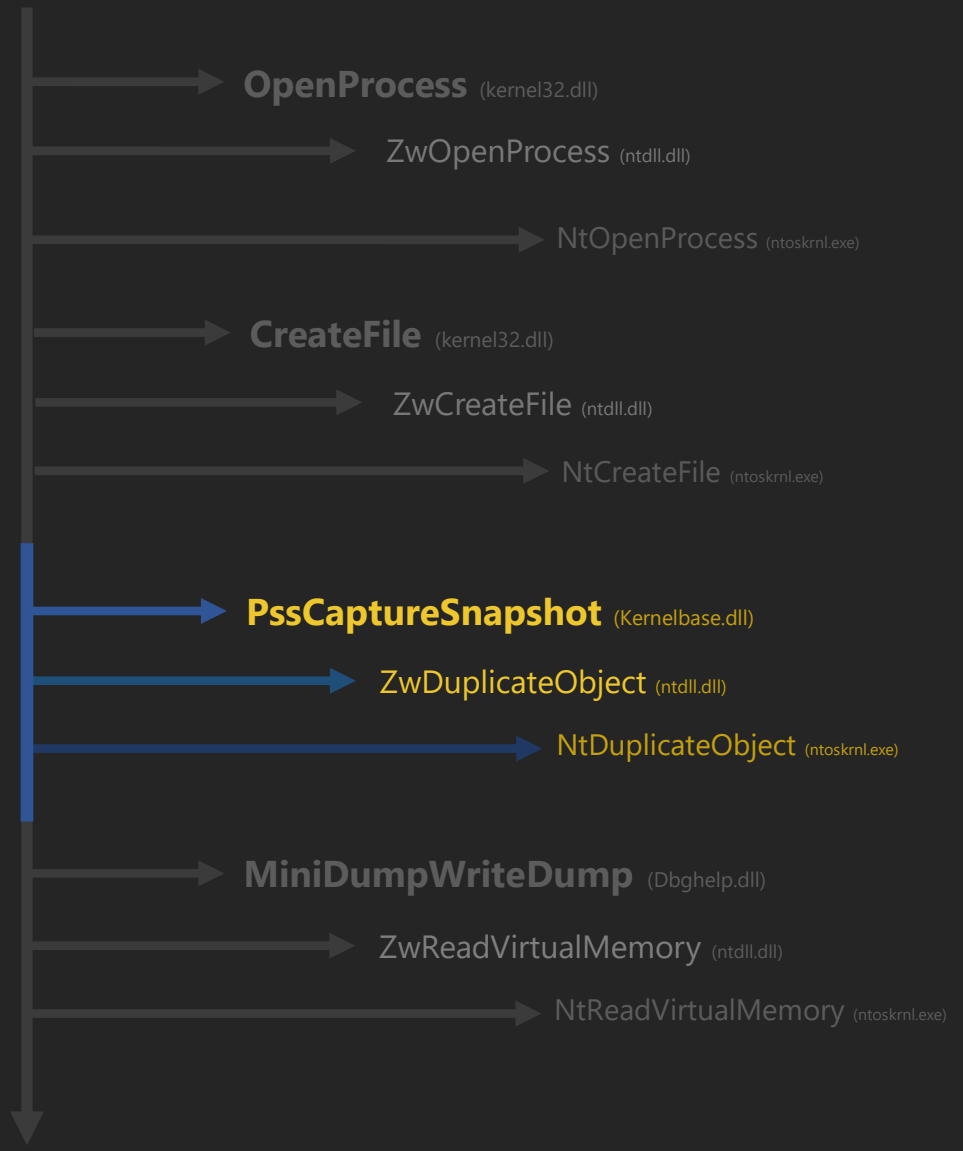
MiniDumpWriteDump (Dbghelp.dll)
ZwReadVirtualMemory (ntdll.dll)
NtReadVirtualMemory (ntoskrnl.exe)

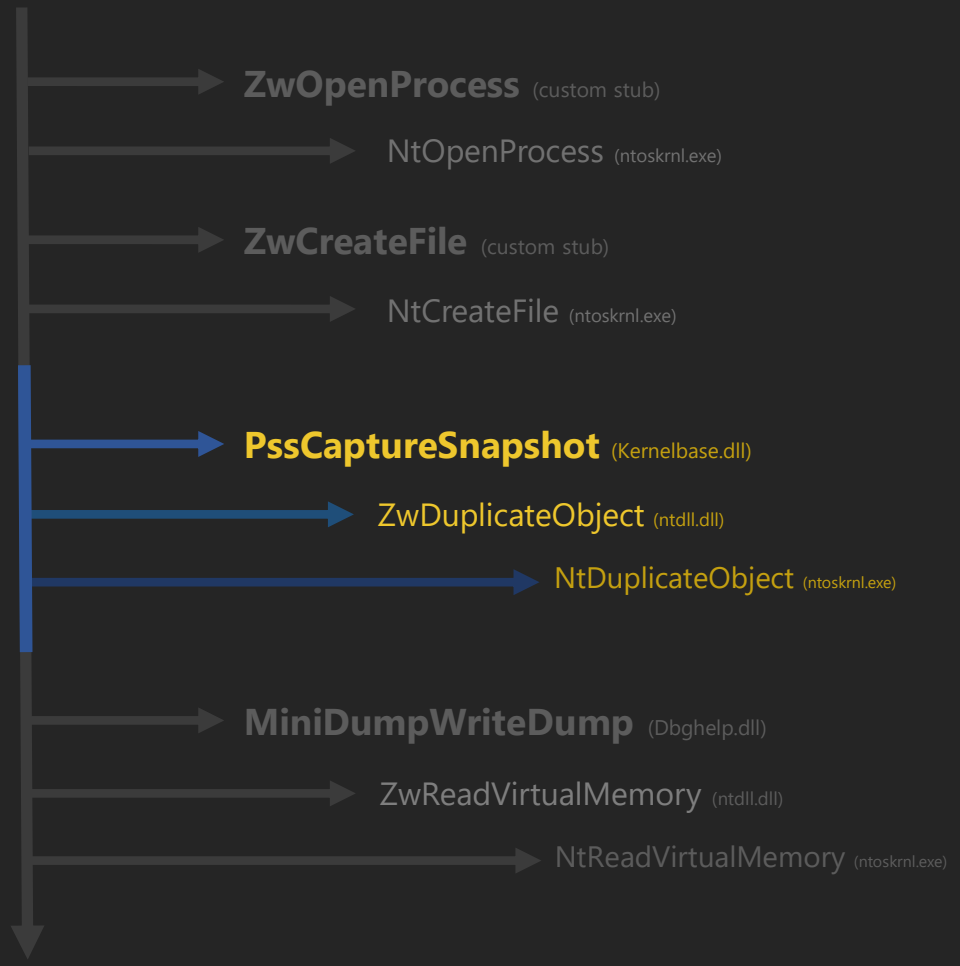
ZwOpenProcess (custom stub)
NtOpenProcess (ntoskrnl.exe)

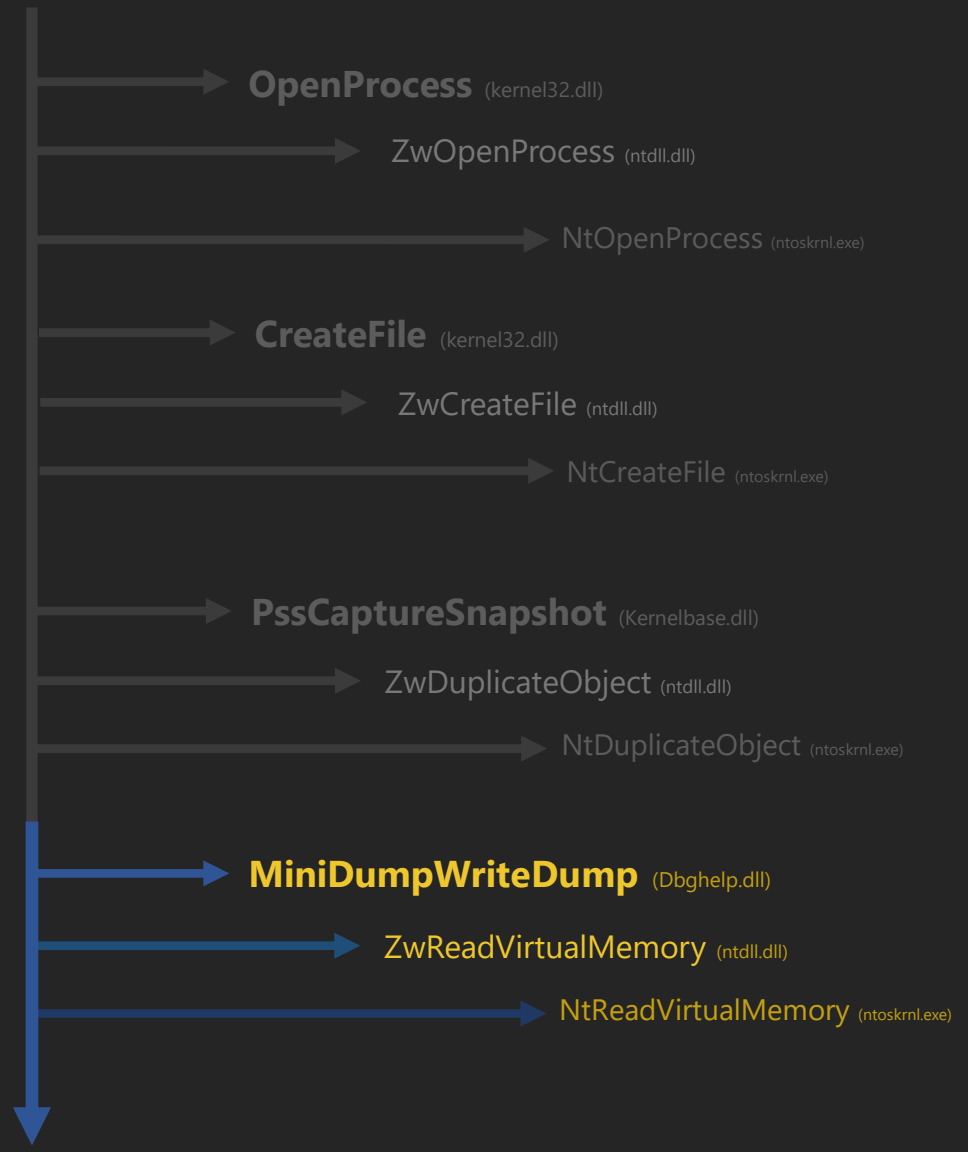
ZwCreateFile (custom stub)
NtCreateFile (ntoskrnl.exe)

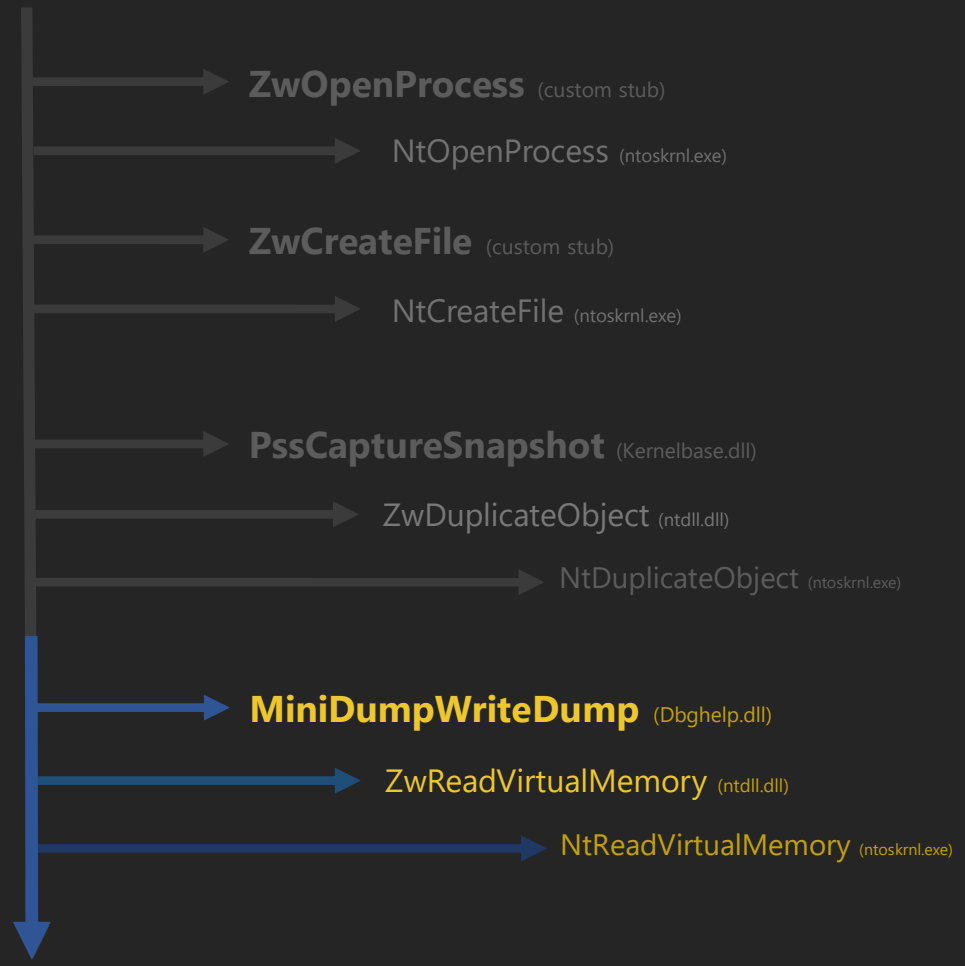
PssCaptureSnapshot (Kernelbase.dll)
ZwDuplicateObject (ntdll.dll)
NtDuplicateObject (ntoskrnl.exe)

MiniDumpWriteDump (Dbghelp.dll)
ZwReadVirtualMemory (ntdll.dll)
NtReadVirtualMemory (ntoskrnl.exe)


OpenProcess (kernel32.dll)
ZwOpenProcess (ntdll.dll)
NtOpenProcess (ntoskrnl.exe)
CreateFile (kernel32.dll)
ZwCreateFile (ntdll.dll)
NtCreateFile (ntoskrnl.exe)
PssCaptureSnapshot (Kernelbase.dll)
ZwDuplicateObject (ntdll.dll)
NtDuplicateObject (ntoskrnl.exe)
MiniDumpWriteDump (Dbghelp.dll)
ZwReadVirtualMemory (ntdll.dll)
NtReadVirtualMemory (ntoskrnl.exe)


ZwOpenProcess (custom stub)
NtOpenProcess (ntoskrnl.exe)
ZwCreateFile (custom stub)
NtCreateFile (ntoskrnl.exe)
PssCaptureSnapshot (Kernelbase.dll)
ZwDuplicateObject (ntdll.dll)
NtDuplicateObject (ntoskrnl.exe)
MiniDumpWriteDump (Dbghelp.dll)
ZwReadVirtualMemory (ntdll.dll)
NtReadVirtualMemory (ntoskrnl.exe)


OpenProcess (kernel32.dll)
ZwOpenProcess (ntdll.dll)
NtOpenProcess (ntoskrnl.exe)
CreateFile (kernel32.dll)
ZwCreateFile (ntdll.dll)
NtCreateFile (ntoskrnl.exe)
PssCaptureSnapshot (Kernelbase.dll)
ZwDuplicateObject (ntdll.dll)
NtDuplicateObject (ntoskrnl.exe)
MiniDumpWriteDump (Dbghelp.dll)
ZwReadVirtualMemory (ntdll.dll)
NtReadVirtualMemory (ntoskrnl.exe)


ZwOpenProcess (custom stub)
NtOpenProcess (ntoskrnl.exe)
ZwCreateFile (custom stub)
NtCreateFile (ntoskrnl.exe)
PssCaptureSnapshot (Kernelbase.dll)
ZwDuplicateObject (ntdll.dll)
NtDuplicateObject (ntoskrnl.exe)
MiniDumpWriteDump (Dbghelp.dll)
ZwReadVirtualMemory (ntdll.dll)
NtReadVirtualMemory (ntoskrnl.exe)

System Calls - OPSEC

- If you are doing a dynamic extraction of service numbers, watchout how you do it!
  - Freshycalls' way of extracting these numbers might be of interest for you
- Manual Syscall executions can be detected using ETW
  - Masked Syscalls might help
- Virtualization can be used to hook Syscalls, be aware of that!

Why Should I Use a C2?

Why Should I Use a C2

- You probably have noticed we've been talking about Cobalt Strike, Metasploit and even Covenant

- Adversary Simulations require:
  - Operators to simulate real-world theat actors
  - Reliable tools that offer the necessary functionality in a customizable way

- That's why we want to use a C2!

- Working with agents allows us to centralize all the functionality and customizations in one single place
  - We don’t want millions of tools, and new processes are expensive!
- Including everything within a custom-made agent or framework requires a lot of time though
- Mature tools such as Metasploit or Cobalt Strike offer reliability and years of work on their shoulders, give some love to them

- Working with agents allows us to centralize all the functionality and customizations in one singl
  - We don't want million r cesses are expensive!
- Including everything withi t or framework requires a lot of time though
- Mature tools such as Metasploit o lt Strike offer reliability and years of work on their shoulders, give some love to them

MANY THANKS!

Any Question?

Is anybody still awake?