Labs - YARA & Sigma for SOC Analysts

Preliminary knowledge

- XDR: Known as Extended detection and response, it's like EDR with AI and automation.

  XDR definition

  Extended detection and response, often abbreviated as XDR, is a unified security incident platform that uses AI and automation. It provides organizations with a holistic, efficient way to protect against and respond to advanced cyberattacks.

- Packer: A program that compresses a file using an algorithm, often times obfuscating the code

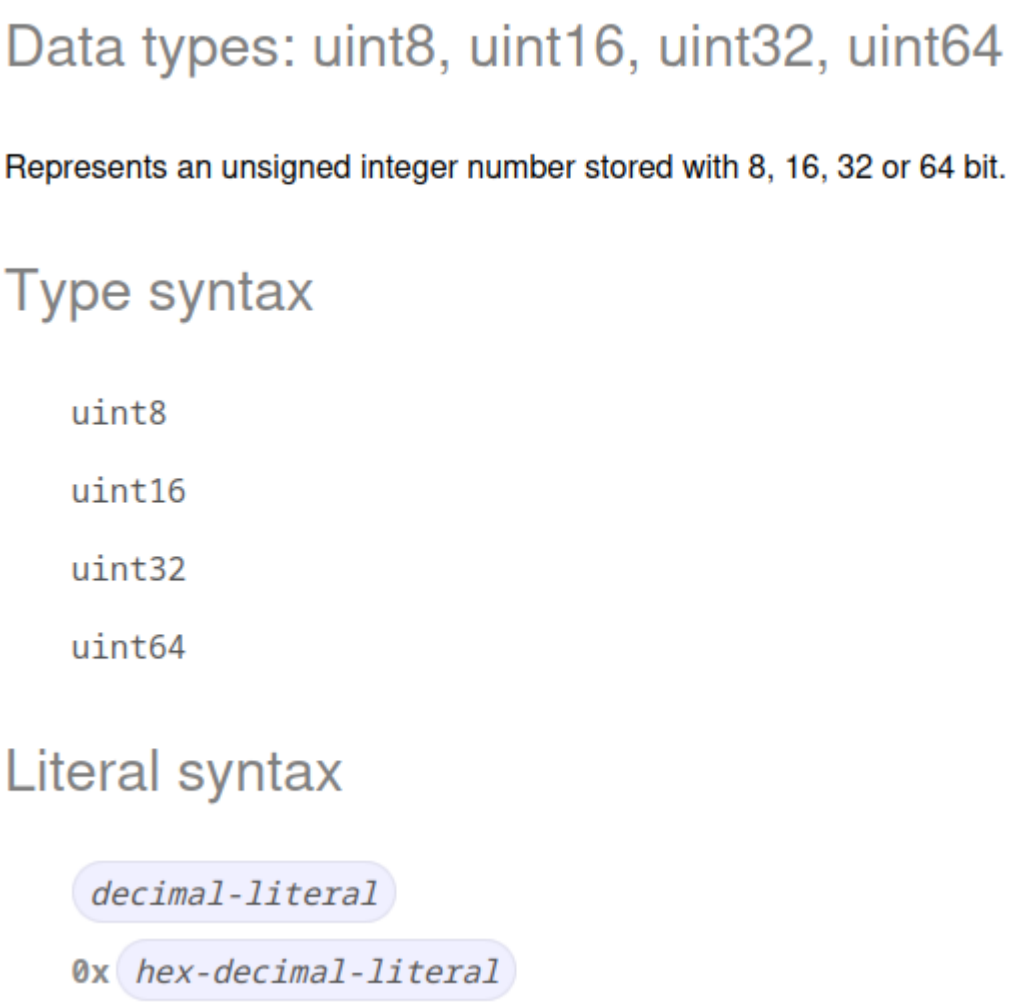
  What is a packer?

  In general, a packer is a program that compresses a file using an algorithm. For example, ZIP, RAR. Some packers, such as UPX, are specifically designed to compress an executable, obfuscating the code in the process. In the context of malware, this can be used as an evasion method.

  There are many types of packers. Both legitimate utilities (VMprotect, ASpack, Enigma Protector), and custom code written by hackers. But these are either less popular in attacks, so we'll leave them out, or too complex for the scope of this article.

  -> UPX is an type of executable file compressor. Web page given at: https://upx.github.io/

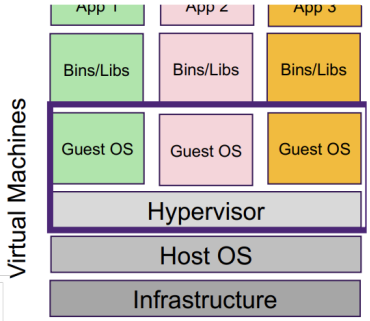
- uint8, uint16, uint32, uint64: Unsigned integers (no negative values) of 8, 16, 32 or 64 bits.

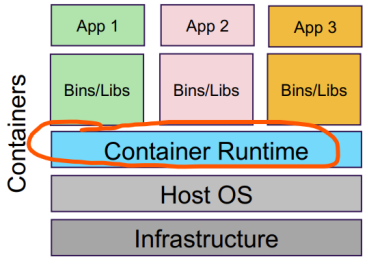


- yara: A tool aimed at idnetifying and classifying malware samples, documentation links:
  https://yara.readthedocs.io/en/latest/
- Strings: A Command that looks at the printable characters in files.
- yarGen: A tool that automatically YARA from strings found in malware files by removing all strings that also appear in legitimate file.
  https://github.com/Neo23x0/yarGen
- Runtime environment: An middleman between the application and the running OS or an Level n-1 abstraction beneath an level n abstraction, because OS can come in different and various types, so an runtime helps resolve that issue of communicating with the various OS for code execution and allows the programmers to focus just on the code.
  - This idea also applies to container runtime.
  - This is also probably why having docker group gives you root because it hs the permission to interact with the host os and requires high privileges

Containers

- **Containers** offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run.
- Completely isolated environments: services, network interfaces, mounts
- Multiple containers running atop the **OS kernel** directly – far more lightweight:
  - share the OS kernel
  - start much faster (in seconds)
  - use a fraction of the memory (compared to booting an entire OS)
  - one physical machine/server can run much more containers than VMs
- Running Linux Containers on Windows (not easy but possible)
  - Windows Subsystem for Linux (WSL)
  - Docker Desktop (helps you build, share, and run containers on Mac and Windows as you do on Linux)





ubuntu CentOS

https://cloud.google.com/containers/
https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/linux-containers

- We also come across this post:

biffbobfred • 2y ago • Edited 2y ago

It's because of what had to be done at the time. Docker should be thought of as "instigator of kernel Magick to get things isolated". At one point those things needed to be done as root.

This is no longer the case. cgroups v2 is a higher level of Kernel Magick isolation, and makes rootless Docker closer to a reality.

https://rootlesscontaine.rs/

2 Reply Award Share ...

Qronosa OP • 2y ago

thank you! that looks like a very useful resource.

2 Reply Award Share ...

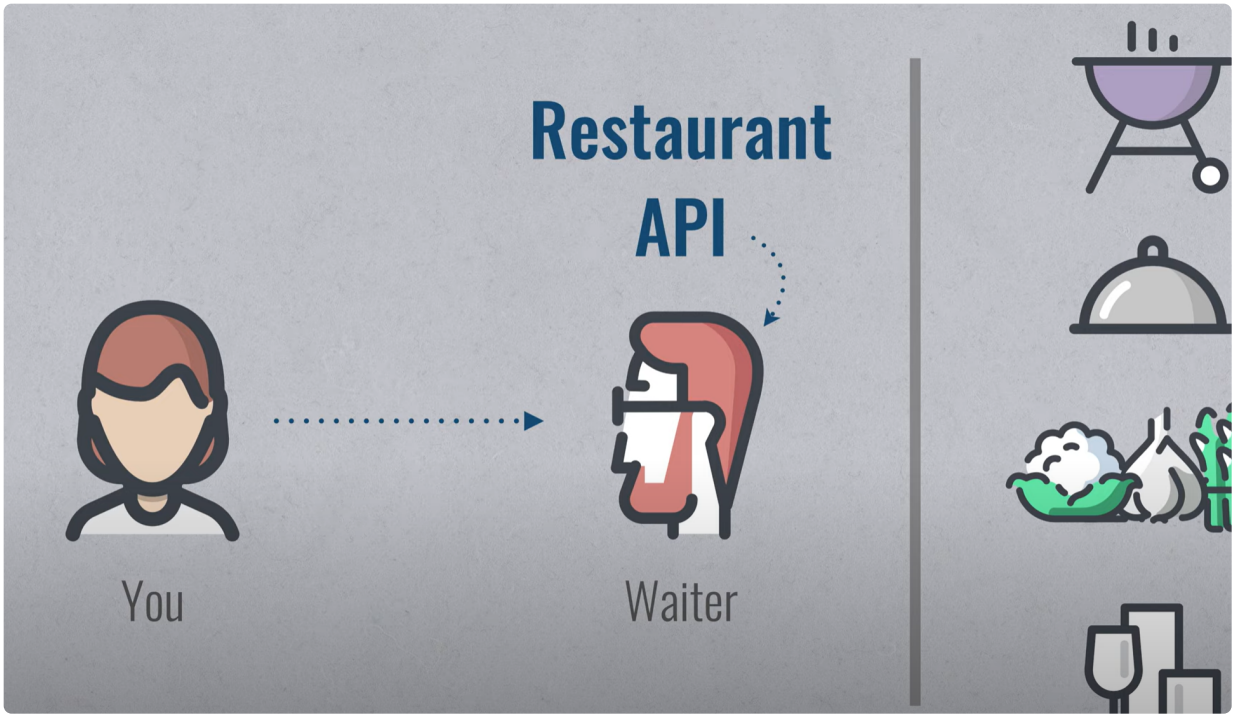
(https://www.reddit.com/r/docker/comments/yk4qrj/dockerd_runs_with_root_user_why_is_this_acceptable/)")

- Referenced resouores

https://stackoverflow.com/questions/3900549/what-is-runtime

https://www.underthehoodlearning.com/what-is-runtime-environment/

- API: Known as the application programming interface, is an interface for the code that user can do some certain request for it to do something (shielding complexity of what the code is doing behind the scene). E.g. restaurant non IT example (waiter is an API for delivering meal, payment, getting drinks for you)



-> RPC vs API's: They are somewhat similar, but the main difference is API's are framework for communication while RPC is an request from the client to the server as if it was local (e.g. on AD environment) with network communication details hidden from the user but API is an framework for communication.

Summary of differences: RPC vs. REST

	RPC	REST
What is it?	A system allows a remote client to call a procedure on a server as if it were local.	A set of rules that defines structured data exchange between a client and a server.
Used for	Performing actions on a remote server.	Create, read, update, and delete (CRUD) operations on remote objects.
Best fit	When requiring complex calculations or triggering a remote process on the server.	When server data and data structures need to be exposed uniformly.
Statefulness	Stateless or stateful.	Stateless.
Data passing format	In a consistent structure defined by the server and enforced on the client.	In a structure determined independently by the server. Multiple different formats can be passed within the same API.

- Example illustration: Emailing to family members to get pizza baked is like saying locally to them bake pizza for me (RPC), while to go to pizza hut you must go pizzeria,

place an order and choose from the menus (framework for communication).

An Example of RPCs and APIs

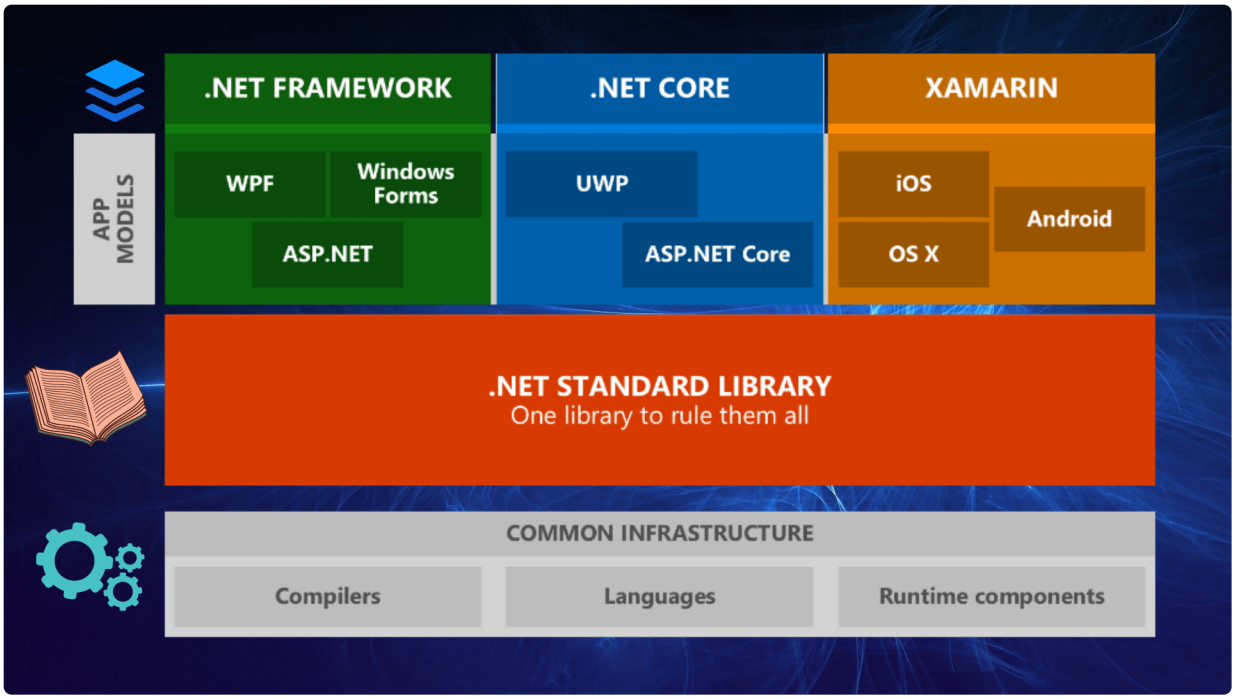
An example can help to demonstrate the subtle difference between RPCs and APIs. RPCs connect to a network and APIs interact with other computers or programs on that network. Essentially, RPCs are local while APIs are communicative.

Let's take a look at another case. Suppose you sent your mom an email to ask her to make some cookies for you. This can be thought of as an RPC. You (the customer) sent a request for cookies to your mom (the server). Your mom does the necessary baking, and then sends you the cookies to fulfill your request.

If you choose to get a pizza and your mom doesn't have the ingredients, you must go to a pizzeria. Just like when you go to the pizzeria, you don't have access to the kitchen ingredients, but you can place an order. The menu provides the pizzas available for ordering along with the toppings and other options you can choose from. This is an example of an application programming interface (API).

An RPC is a specific request for information on the blockchain, whereas an API is like a framework for complex communication.

- Virtualisation: Translates IT hardware resources into software/virtual resources (e.g. For VM, it's through technology like hypervisor).
- Framework: An software model where it helps developer from reinventing the wheels through having useful reusable piece of code.
- .NET Framework: It is an software development framework and runtime that is used to ease application development across various platforms.
    - It is an virtual machine for compiling and executing programs written in different languages such as C#, VB. NET, etc.
- E.g. Main components of .NET Framework:



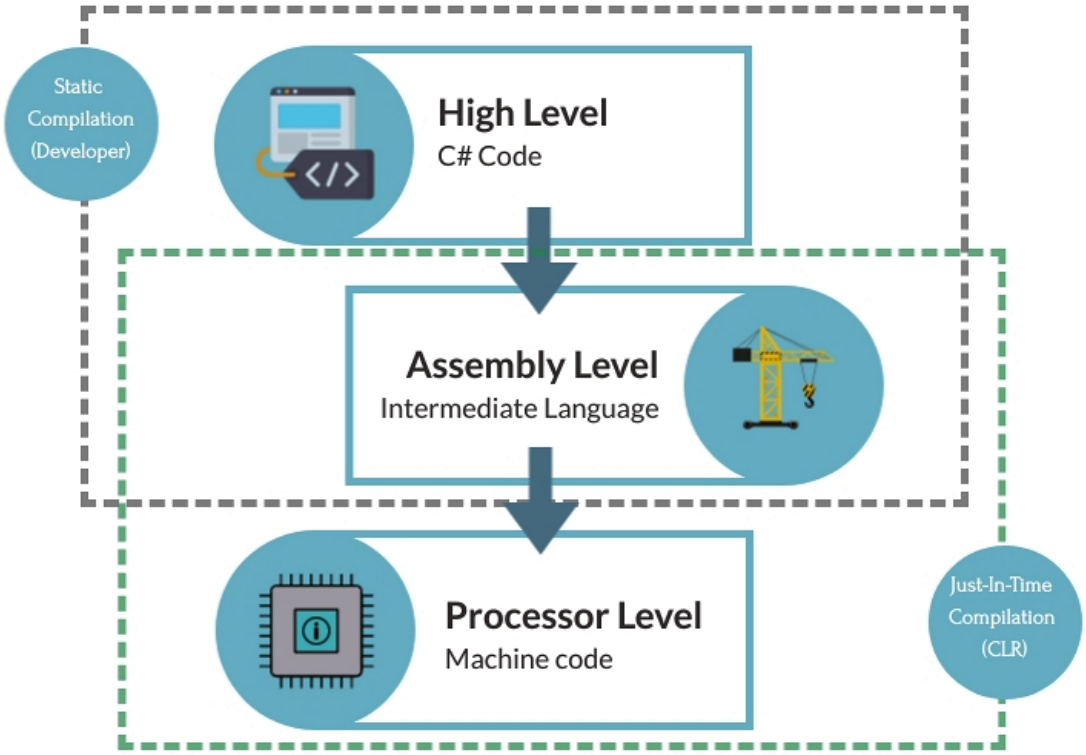
-> .Net Framework, .Net Core and XAMARIN and different implementation of .Net platform (here we denote our original meaning of Net Framework as .Net platform as there we see .Net Framework actually means one implementation of App models shown above).
-> .Net framework for windows specific, .Net Core for Windows, Mac and Linux platform and Xamarin for mobile related platform (IOS, Android, Windows phone).

-> Common infrastructure:
-> The base layer of components that allows code to execute in practice, from compilers to languages to runtime components.
-> Code written in any .Net language is compiled into intermediate byte code called Common Intermediate Language (CIL) that be ported across operating system and platforms.
-> The CIL is then compiled by the so called Common Language Runtime (CLR) which compile CIL code into machine-readable code that can be executed on the specific platform.

-> .Net Standard Library: It is the set of fundamental API's that all .Net implementations needs to implement. The reason for this is for greater unity in theh .NET ecosystem (without it, we can have like different forms of HTML and code sharing will be hard)- we can think of HTML as .NET Standard while browsers as .NET implementations.
-> Brings "structure" and "unity" to the .NET platforms:
- We don't use it directly but it acts like abstract class that a set of fundamental API's (string processes, database connections, I/O connections ...) known as Base Class Library (BCL) that all .NET implementation must implement.

-> App Model: It is the framework components that are specific to a certain type of apps. E.g. The Windows Forms app model include all the code that we need to create a Desktop Windows application (form, button, label).

- Compilation process for C#:

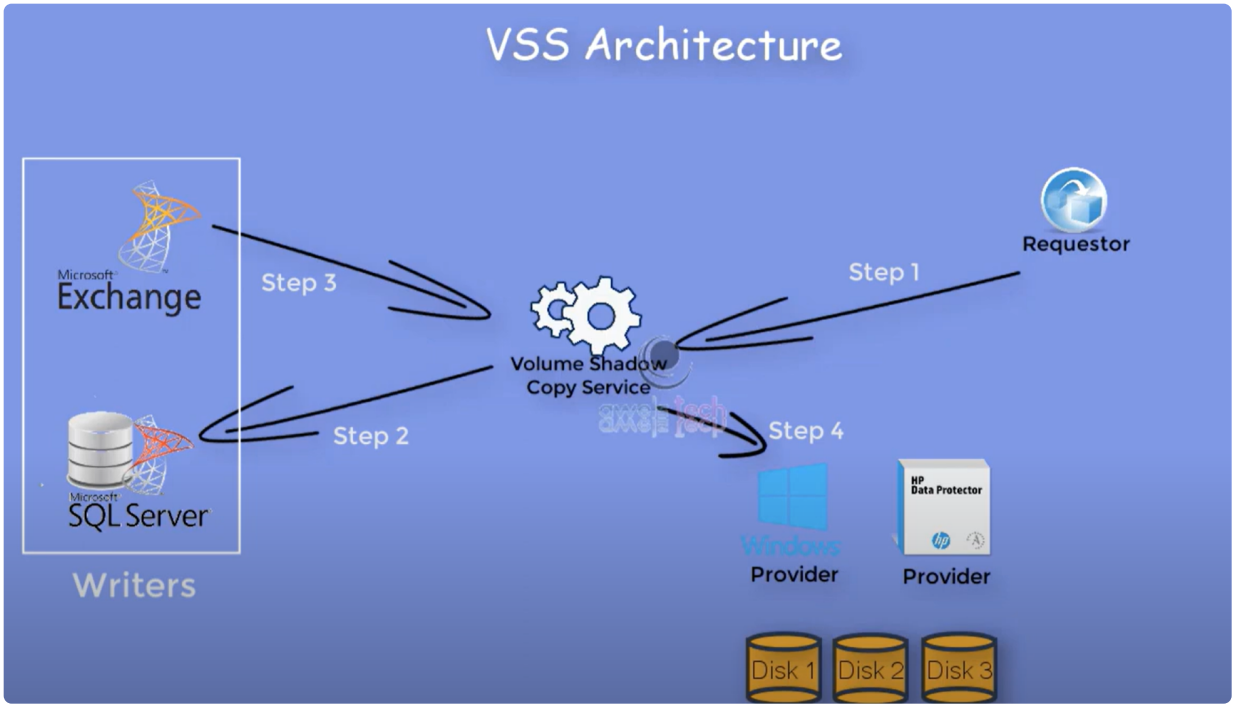


  1. Static complilation by devleopers so code become an Portable Executable files such as .exe or .dll
  -> Files are portable because they can be run on various platforms!
  -> .exe file can be run by itself because it has an entry point while dll's cannot, they are functions that are linked to the appllication at run time dynamically.
  2. When the .Net program is launched from the previous step, the CLR compiles the CIL (from previous step) into machine code suitable for the specific OS.
- References: https://www.underthehoodlearning.com/what-is-runtime-environment/
- https://www.underthehoodlearning.com/what-really-is-the-net-framework/
- monodis: Tool for disassembling CIL code: https://www.mono-project.com/docs/tools+libraries/tools/monodis/
- dnSpy: An reverse engineering solution for .NET and Unity (game engine devloped by C++) assembly.
- Pointers: Stores addresses instead of values.
- Wiper attacks: Malware-based attacks designed to permanently delete or corrupt data on targeted system.
  https://www.crowdstrike.com/cybersecurity-101/attack-types/wiper-attack/
- Entropy: In paritcular Shannon Entropy, it measure randomness within a set of data, with values ranging from 0-8 (8 being very random).
  - In relation to malware, compression reduces sizes of certain parts by replacing duplicated parts with references to single instance of that part, which causes less duplicated contents and less predictability in data.

https://cocomelonc.github.io/malware/2022/11/05/malware-analysis-6.html
https://practicalsecurityanalytics.com/file-entropy/

- UTF-8, 16, 32, ASCII: Fixed length encoding of 8, 16 or 32 bits, the first 128 in ASCII characters are also the same in UTF-8.
  https://stackoverflow.com/questions/19212306/whats-the-difference-between-ascii-and-unicode
- Sections: Sections in PE are the actual content of the executable code, which can be data, resources utilized by program or the actual code (as examples).
- Authenticode: Also known as code signing, is like digital signature for software, it allows the validation of authenticity of the software shipped.
  https://reversea.me/index.php/authenticode-i-understanding-windows-authenticode/
- Resource: The resources that the executable utilises in PE. There are various types of resources that an program can have (e.g. icon, dialog...)
- yaraGen: General tool for developing good yara rule through automated tools.
- Opcode: A number interpreted by the machine which represents the basic operation to be performed by the instruction given to the computer processor.
- Instruction: An instruction is an order given to a computer processor by an computer program.
  https://www.geeksforgeeks.org/representing-instructions-in-computer/
  https://stackoverflow.com/questions/17638888/difference-between-opcode-byte-code-mnemonics-machine-code-and-assembly
- DOS: Known as the Disk operating system, an different operating system than windows that is free and some differences being single tasking (only allows one task execution at once).
  - User in windows can choose to run program in "DOS" mode if they choose to.
- MS-DOS stub: An application that runs under MS-DOS, with the default message being "This program cannot run in DOS mode".
  https://www.geeksforgeeks.org/difference-between-dos-and-windows/
  https://learn.microsoft.com/en-us/windows/win32/debug/pe-format
- Executable Image: A program that is in a certain state (from being compiled and linked) that can be executed.
  https://discuss.codecademy.com/t/what-is-an-executable-image-and-what-is-the-difference-between-out-and-o/755415
- In memory and On disk: In memory is where memory resides on the RAM and not persistent while On disk is where memory is on the disk and is generally persistent.
  https://stackoverflow.com/questions/25802521/difference-between-in-memory-databases-and-disk-memory-database
  https://www.heavy.ai/technical-glossary/in-memory-database
- Hex editor: Editor where you can edit the hex representation of a file.

- Device file: A special file that appears as normal file on the file system (but they are not actual files).
  - They allow the interaction with device drivers
  - E.g. Device file of /dev/null discards everything written to it, it is like an black hole.
- Device driver: A special program that controls some specific hardware attached to the computer (e.g. the mouse and keyboard)
  https://www.techtarget.com/searchenterprisedesktop/definition/device-driver
- Shellcode: Hex representation of a binary's executable machine code.
- & in powershell ForEach-Object: It allows you to execute commands in string format.
  - Scripting in powershell ForEach-Object requires spaces between cmdlets and arguments.
- RWX in process memory: It means the memory in that region can be written, read and executed by some thread.
- Buffer: Region of memory used to store data temporarily while it is being moved from one place to another.
- Checksum: At an abstract level, tests for whether the packet/message transferred is valid through checking the appropriate value that the packet should have.
- Memory capture: A capture of memory of the current system, which can potentially allow for tools like YARA to be used for scanning it for suspicious things.
- Memory dump: The process of taking all information content in RAM and writing it to a storage file (usually an .dmp file)
- Memory foresencis: Process of capturing and analyzing an computer's memory to uncover important information.
  https://www.techtarget.com/whatis/definition/memory-dump
  https://intezer.com/blog/incident-response/memory-analysis-forensic-tools/#h-what-s-memory-forensics
- Volume Shadow Copy Service: Backing up and restoring data without taking the application down.
  - Diagram of high level flow chart:



https://www.youtube.com/watch?v=WTKRH7csV8U

- COM: Known as component object model, is an mechanism/standard that allows binary software component to interact, regardless of the language, structure or location of the through a defined COM interface.
  https://learn.microsoft.com/en-us/windows/win32/com/the-component-object-model
  https://stackoverflow.com/questions/455687/what-is-com-component-object-model-in-a-nutshell
- UUID: Known as Universally Unique Identifier, it's an 128 bit identifer that is deemed unique for most practical purpose.
  https://www.howtogeek.com/devops/what-are-uuids-and-why-are-they-useful/
- SSHD: sshd is openssh (open source implementation of ssh) server process.
  https://www.ssh.com/academy/ssh/sshd
- Virtual memory: Memory used by processes which are mapped to physical memory.
- Handle: In Windows it's an abstract reference value to a resource, often memory or an openfile, or a pipe.
  https://stackoverflow.com/questions/902967/what-is-a-windows-handle
- Process Security and Access Rights: An model that windows implement to control access to process object.
- Process_vm_read (0x0010): Reads virtual memory (hence memory) of a process.
- Process_Query_limited_information (0x1000): Retrieves certain information about a process and automatically gives the handle of Process_query_information access right.
- Process_Query_information(0x0400): Retrieves certain information about a process.
  https://learn.microsoft.com/en-us/windows/win32/procthread/process-security-and-access-rights?redirectedfrom=MSDN

- Mitre Attack: A globally-accessible knowledge base of adversary tactics and techniques based on real-world observations.
  https://attack.mitre.org/
- Lsass: Known as the Local Security Authority Subsystem Service, it is a process in Windows that is responsible for enforcing Security Policy of the System.
  https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service
- Event 4776: Logs NTLM authentication attempt performed on "authorative" (DC for domain account and local computer for local attempts) computer.
  https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4776
- Useful sigma documentation to conult: https://github.com/SigmaHQ/sigma-specification/blob/version_2/Sigma_specification.md
- .sys file: A system file used by DOS and Windows OS that are used to store settings and store device driver or other windows functions.
  https://fileinfo.com/extension/sys
- Endpoint: An endpoint is an device that can be connected to the network.
- EDR: Known as end point detection and response, is an endpoint security solution that continuously monitor for cyber-threats like ransomware and malware on the end-user devices.
  https://www.crowdstrike.com/cybersecurity-101/endpoint-security/what-is-an-endpoint/
  https://www.crowdstrike.com/cybersecurity-101/endpoint-security/endpoint-detection-and-response-edr/
- Sysmon: Sysmon is a windows system service and a device driver that allows to monitor and log system activity to the windows activity log.
  https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon
- Artifacts: A piece of data that may or may not be relevant to the investigation / response.
  https://security.stackexchange.com/questions/138118/what-is-the-difference-between-artifact-and-evidence
- Mapping file: For chainsaw purposes, it seems to be a mapping applied to sigma rules and specifies which field in the event logs to be used to sigma rule matching.
- E.g. Mapping file containing NewProcessName matching
  - Translate NewProcess from Sigma to Event.EventData.NewProcessName, which can be used for rule matching for e.g. on Event ID 4688

```
- from: NewProcessName
  to: Event.EventData.NewProcessName
  visible: false
```

```
- <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
- <System>
 <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54849625-5478-4994-A5BA-3E3B0328(
 <EventID>4688</EventID>
 <Version>2</Version>
 <Level>0</Level>
 <Task>13312</Task>
 <Opcode>0</Opcode>
 <Keywords>0x8020000000000000</Keywords>
 <TimeCreated SystemTime="2015-11-12T02:24:52.377352500Z" />
 <EventRecordID>2814</EventRecordID>
 <Correlation />
 <Execution ProcessID="4" ThreadID="400" />
 <Channel>Security</Channel>
 <Computer>WIN-GG82ULGC9GO.contoso.local</Computer>
 <Security />
 </System>
- <EventData>
 <Data Name="SubjectUserSid">S-1-5-18</Data>
 <Data Name="SubjectUserName">WIN-GG82ULGC9GO$</Data>
 <Data Name="SubjectDomainName">CONTOSO</Data>
 <Data Name="SubjectLogonId">0x3e7</Data>
 <Data Name="NewProcessId">0x2bc</Data>
 <Data Name="NewProcessName">C:\\Windows\\System32\\rundll32.exe</Data>
 <Data Name="TokenElevationType">%%1938</Data>
 <Data Name="ProcessId">0xe74</Data>
 <Data Name="CommandLine" />
 <Data Name="TargetUserSid">S-1-5-21-1377283216-344919071-3415362939-1104</Data>
 <Data Name="TargetUserName">dadmin</Data>
 <Data Name="TargetDomainName">CONTOSO</Data>
 <Data Name="TargetLogonId">0x4a5af0</Data>
 <Data Name="ParentProcessName">C:\\Windows\\explorer.exe</Data>
 <Data Name="MandatoryLabel">S-1-16-8192</Data>
 </EventData>
</Event>
```

https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4688

- Call Trace: In splunk, this is not explictly mentioned or defined, but it seems to be what is loaded by the source process based on the examples given in the link. https://www.splunk.com/en_us/blog/security/you-bet-your-lsass-hunting-lsass-access.html
- Comsvc.dll: A natively found dll that can be used to dump lsass through its export function.
- -c in python sigma: A cofiguration/mapping file applied to some fields from the sigma file for rule matching in the actual SIEM//scripting format.
  E.g. Splunk example (notice how EventID gets converted to EventCode for splunk SIEM)

proc_access_win_lsass_dump_comsvcs_dll.yml - Notepad

File Edit Format View Help

```
title: Lsass Memory Dump via Comsvcs DLL
id: a49fa4d5-11db-418c-8473-1e014a8dd462
status: test
description: Detects adversaries leveraging the MiniDump export function from comsvcs.dll via rundll32 to perform a memory du
references:
    - https://twitter.com/shantanukhande/status/1229348874298388484
    - https://modexp.wordpress.com/2019/08/30/minidumpwritedump-via-com-services-dll/
author: Roberto Rodriguez (Cyb3rWard0g), OTR (Open Threat Research)
date: 2020/10/20
modified: 2022/10/09
tags:
    - attack.credential_access
    - attack.t1003.001
logsource:
    category: process_access
    product: windows
detection:
    selection:
        TargetImage|endswith: '\lsass.exe'
        SourceImage: 'C:\Windows\System32\rundll32.exe'
        CallTrace|contains: 'comsvcs.dll'
    condition: selection
falsepositives:
    - Unknown
level: critical
```

splunk-windows.yml - Notepad

File Edit Format View Help

```
  windows-printservice-admin:
    product: windows
    service: printservice-admin
    conditions:
      source: 'WinEventLog:Microsoft-Windows-PrintService/Admin'
  windows-printservice-operational:
    product: windows
    service: printservice-operational
    conditions:
      source: 'WinEventLog:Microsoft-Windows-PrintService/Operational'
  windows-codeintegrity-operational:
    product: windows
    service: codeintegrity-operational
    conditions:
      source: 'WinEventLog:Microsoft-Windows-CodeIntegrity/Operational'
  windows-smbclient-security:
    product: windows
    service: smbclient-security
    conditions:
      source: 'WinEventLog:Microsoft-Windows-SmbClient/Security'
  windows-rpc-firewall:
    product: rpc_firewall
    category: application
    conditions:
      source: 'WinEventLog:RPCFW'
  windows-firewall-advanced-security:
    product: windows
    service: firewall-as
    conditions:
      source: 'WinEventLog:Microsoft-Windows-Windows Firewall With Advanced Security/Firewall'
  windows-bits-client:
    product: windows
    service: bits-client
    conditions:
      source: 'WinEventLog:Microsoft-Windows-Bits-Client/Operational'
fieldmappings:
  EventID: EventCode
```

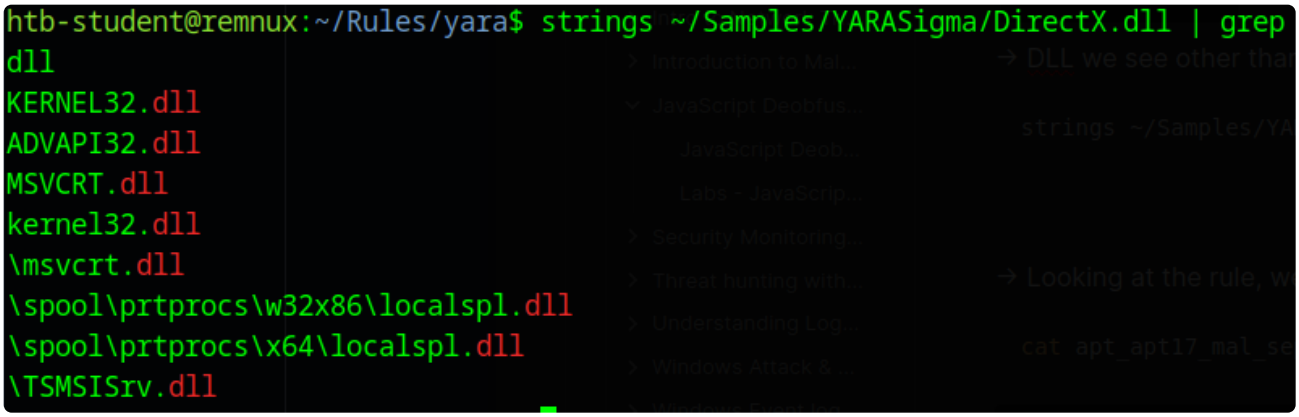
- Script block: An particular logging in powershell that occurs which logs the script being executed. Each Script block can be identified by an unique id. It also has an event id of 4104.
  https://www.techtarget.com/searchwindowsserver/tutorial/Set-up-PowerShell-script-block-logging-for-added-security

Developing YARA Rules

Question

- Perform string analysis on the "DirectX.dll" sample that resides in the "/home/htb-student/Samples/YARASigma" directory of this section's target. Then, study the "apt_apt17_mal_sep17_1.yar" YARA rule that resides in the "/home/htb-student/Rules/yara" directory and replace "X.dll" with the correct DLL name to ensure the rule will identify "DirectX.dll". Enter the correct DLL name as your answer. Answer format: _.dll
  -> We perform string analysis on DirectX.dll and grep for dll's:

```
strings ~/Samples/YARASigma/DirectX.dll | grep dll
```



-> Looking at the rule, we see:

```
cat apt_apt17_mal_sep17_1.yar
```

```
rule APT17_Malware_Oct17_1 {
   meta:
      description = "Detects APT17 malware"
      license = "Detection Rule License 1.1 https://github.com/Neo23x0/signature
-base/blob/master/LICENSE"
      author = "Florian Roth (Nextron Systems)"
      reference = "https://goo.gl/puVc9q"
      date = "2017-10-03"
      hash1 = "dc9b5e8aa6ec86db8af0a7aa897ca61db3e5f3d2e0942e319074db1aaccfdc83"
   strings:
      $s1 = "\\spool\\prtprocs\\w32x86\\localspl.dll" ascii
      $s2 = "\\spool\\prtprocs\\x64\\localspl.dll" ascii
      $s3 = "\\msvcrt.dll" ascii
      $s4 = "\\TSMSISrv.dll" ascii
   condition:
      ( uint16(0) == 0x5a4d and filesize < 500KB and all of them )
}
```

-> So TSMSISrv.dll is likely to be the DLL we want.
-> We modify the rule and test is accordingly:

```
yara apt_apt17_mal_sep17_1.yar ~/Samples/YARASigma/
```

```
htb-student@remnux:~/Rules/yara$ yara apt_apt17_mal_sep17_1.yar ~/Samples/YARASi
gma/
APT17_Malware_Oct17_1 /home/htb-student/Samples/YARASigma//DirectX.dll
```

-> And this confirms TSMSISrv.dll is the dll we want.

Hunting Evil with YARA (Windows Edition)

Question

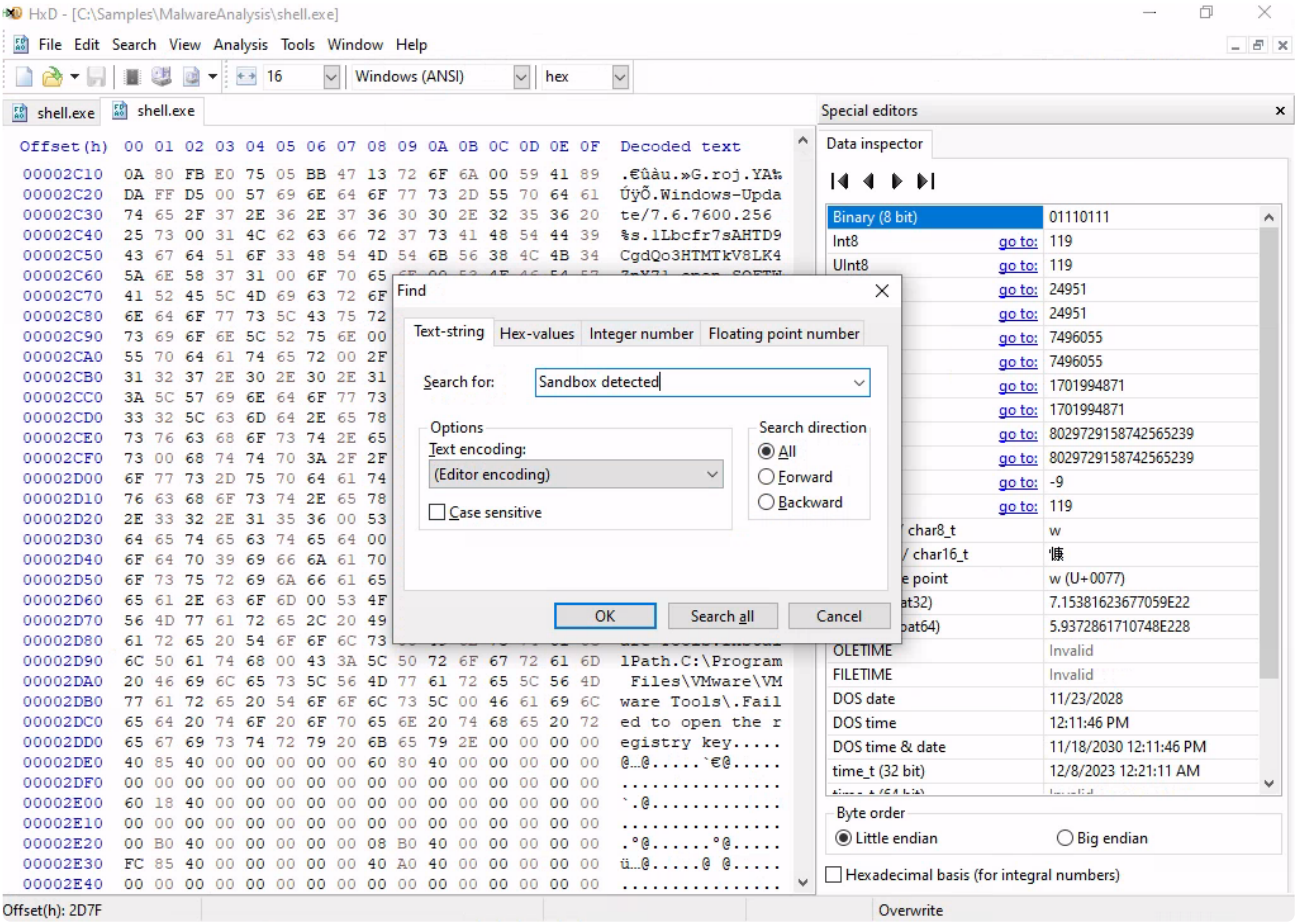
- Study the "C:\Rules\yara\shell_detector.yar" YARA rule that aims to detect "C:\Samples\MalwareAnalysis\shell.exe" inside process memory. Then, specify the appropriate hex values inside the "$sandbox" variable to ensure that the "Sandbox detected" message will also be detected. Enter the correct hex values as your answer. Answer format: Remove any spaces
  -> We first study the YARA rule file as follows:

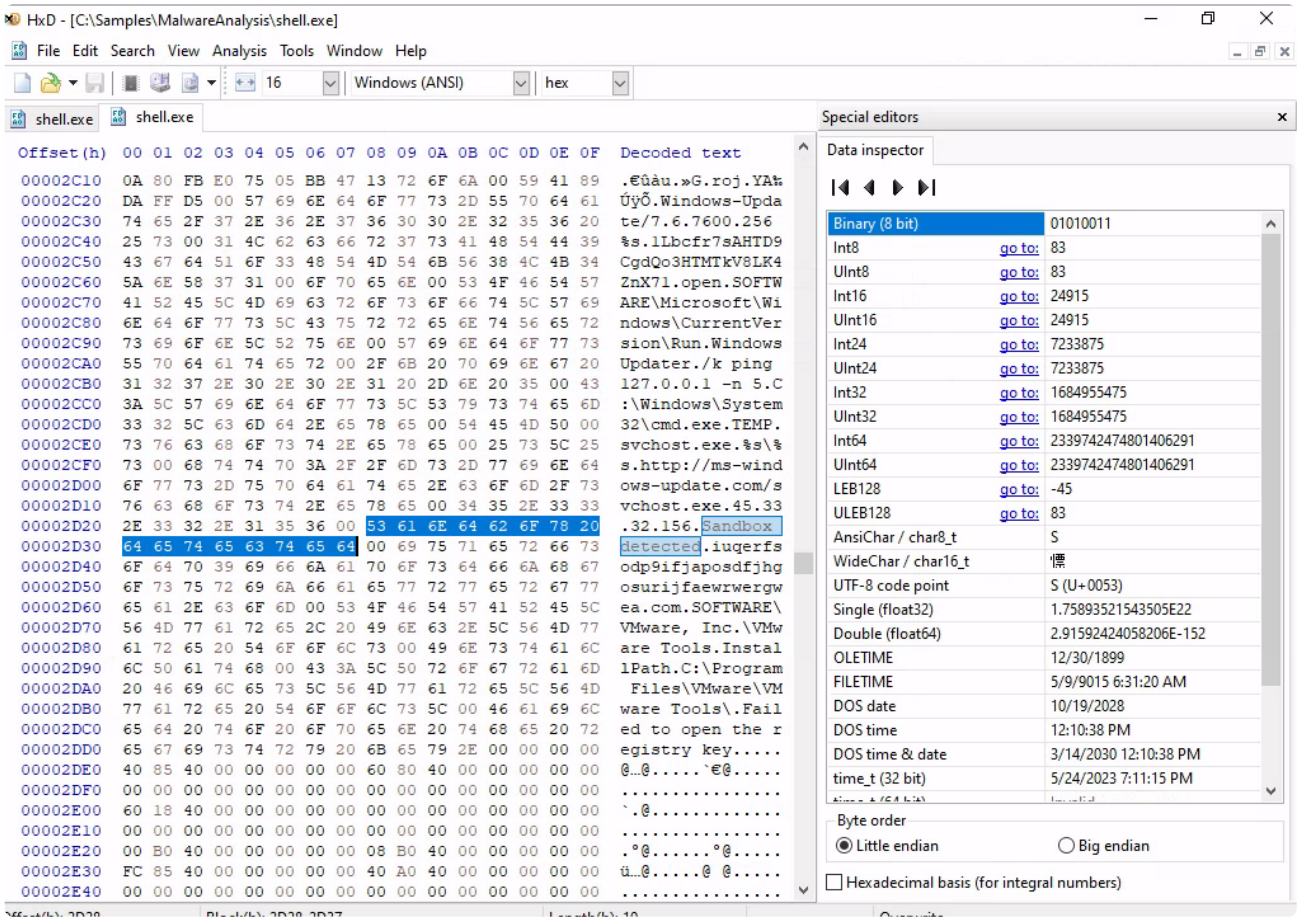
```
rule shell_detected
{
        meta:
```

```
                description             = "Detect Domain & Sandbox
Message In Process Memory"
                author                  = "Dimitrios Bougioukas"

        strings:
                $domain         = { 69 75 71 65 72 66 73 6f 64 70 39 69
66 6a 61 70 6f 73 64 66 6a 68 67 6f 73 75 72 69 6a 66 61 65 77 72 77 65
72 67 77 65 61 2e 63 6f 6d }
                $sandbox        = {  }
        condition:
                $domain and $sandbox
}
```

-> We then open the shell.exe file using an hex editor and searched for the string "Sandbox detected"





-> We copy the hex as follows

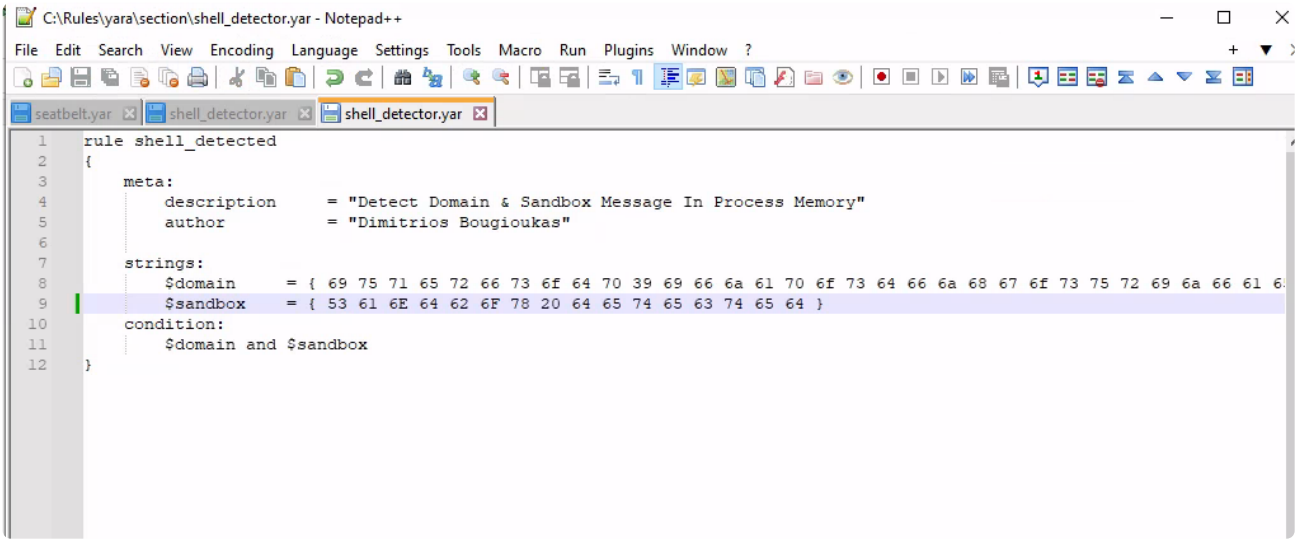
```
53 61 6E 64 62 6F 78 20 64 65 74 65 63 74 65 64
```

-> Hence, the appropriate hex value for sandbox would be as follows and put it in the file:

```
rule shell_detected
{
        meta:
                description             = "Detect Domain & Sandbox
Message In Process Memory"
                author                  = "Dimitrios Bougioukas"

        strings:
                $domain         = { 69 75 71 65 72 66 73 6f 64 70 39 69
66 6a 61 70 6f 73 64 66 6a 68 67 6f 73 75 72 69 6a 66 61 65 77 72 77 65
72 67 77 65 61 2e 63 6f 6d }
                $sandbox        = { 53 61 6E 64 62 6F 78 20 64 65 74 65
63 74 65 64 }
        condition:
```

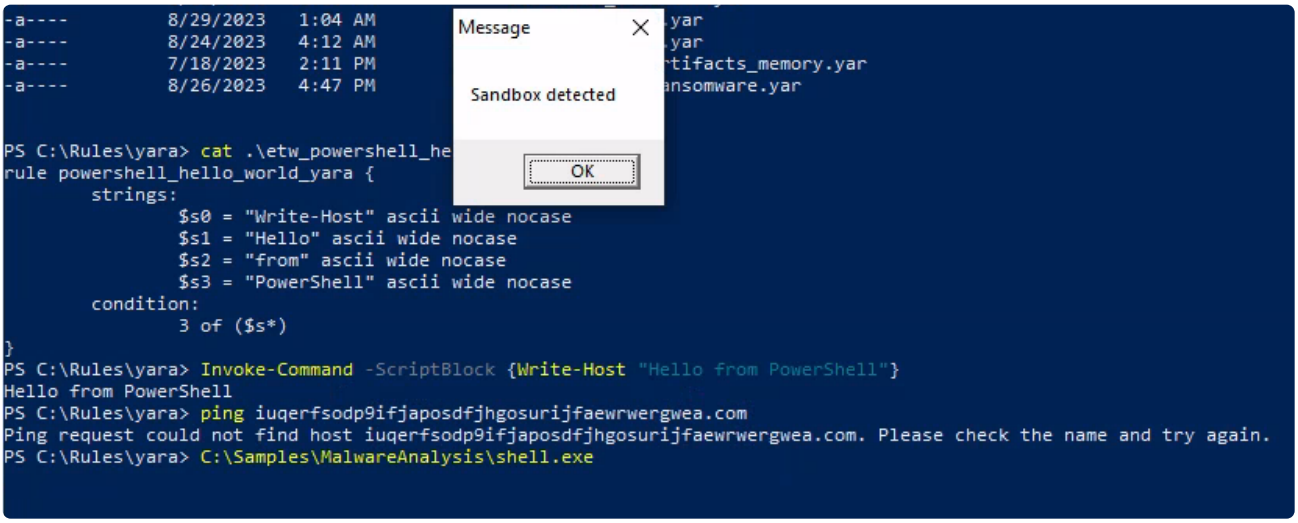
```
                $domain and $sandbox
}
```



-> `53616E64626F78206465746563746564`

-> Now we run the shell and analyse it using powershell script

```
C:\Samples\MalwareAnalysis\shell.exe
```




```
Get-Process | ForEach-Object {"Scanning for any shell in memory for
pid:" + $_.id; & "yara64.exe" "C:\Rules\yara\section\shell_detector.yar"
$_.id}
```

```
PS C:\tools\SilkETW\v8\SilkETW> Get-Process | ForEach-Object {"Scanning for any shell in memory for pid:" + $_.id; & "ya
ra64.exe" "C:\Rules\yara\section\shell_detector.yar" $_.id}
Scanning for any shell in memory for pid:7840
Scanning for any shell in memory for pid:7072
Scanning for any shell in memory for pid:2396
Scanning for any shell in memory for pid:5172
Scanning for any shell in memory for pid:7540
Scanning for any shell in memory for pid:424
error scanning 424: can not attach to process (try running as root)
Scanning for any shell in memory for pid:496
error scanning 496: can not attach to process (try running as root)
Scanning for any shell in memory for pid:4844
error scanning 4844: can not attach to process (try running as root)
Scanning for any shell in memory for pid:5236
Scanning for any shell in memory for pid:3872
Scanning for any shell in memory for pid:5092
Scanning for any shell in memory for pid:776
Scanning for any shell in memory for pid:988
Scanning for any shell in memory for pid:5412
Scanning for any shell in memory for pid:792
Scanning for any shell in memory for pid:796
Scanning for any shell in memory for pid:4832
Scanning for any shell in memory for pid:5788
shell_detected 5788
Scanning for any shell in memory for pid:0
error scanning 0: can not attach to process (try running as root)
Scanning for any shell in memory for pid:4076
Scanning for any shell in memory for pid:660
Scanning for any shell in memory for pid:1728
error scanning 1728: can not attach to process (try running as root)
Scanning for any shell in memory for pid:1904
```

-> We got an shell detected with an pid of 5788!

Hunting Evil with YARA (Linux Edition)

Question

- Study the following resource https://blogs.vmware.com/security/2022/09/threat-report-illuminating-volume-shadow-deletion.html to learn how WannaCry performs shadow volume deletion. Then, use yarascan when analyzing "/home/htb-student/MemoryDumps/compromised_system.raw" to identify the process responsible for deleting shadows. Enter the name of the process as your answer.
  -> We study the related links and see the following related to WannaCry:

  WMIC SHADOWCOPY Delete Used in Wana Decrypt0r 2.0

  The confusingly named Wana Decrypt0r 2.0 also uses a very conservative approach; as shown, the command calls not only vssadmin to delete shadows, but also wmic shadowcopy delete. Then, just to hamper recovery further, bcdedit is called to prevent automatic recovery from the VSCs that were presumably already deleted.

  Wana Decrypt0r 2.0 is an ancestor of WannaCry, which wormed its way around the world starting in 2017 using the EternalBlue SMB exploit. The large number of variants associated with this Wana Decrypt0r 2.0 can be attributed in part to a ransomware generator. This program, Aron WanaCryptor 2.0 Generator v1.0, allows enterprising criminals to customize the WannaCry lock screen text, images, and colors. For this reason, there are many variants of this ransomware that are functionally identical, but given new names based on the text from the ransom window.

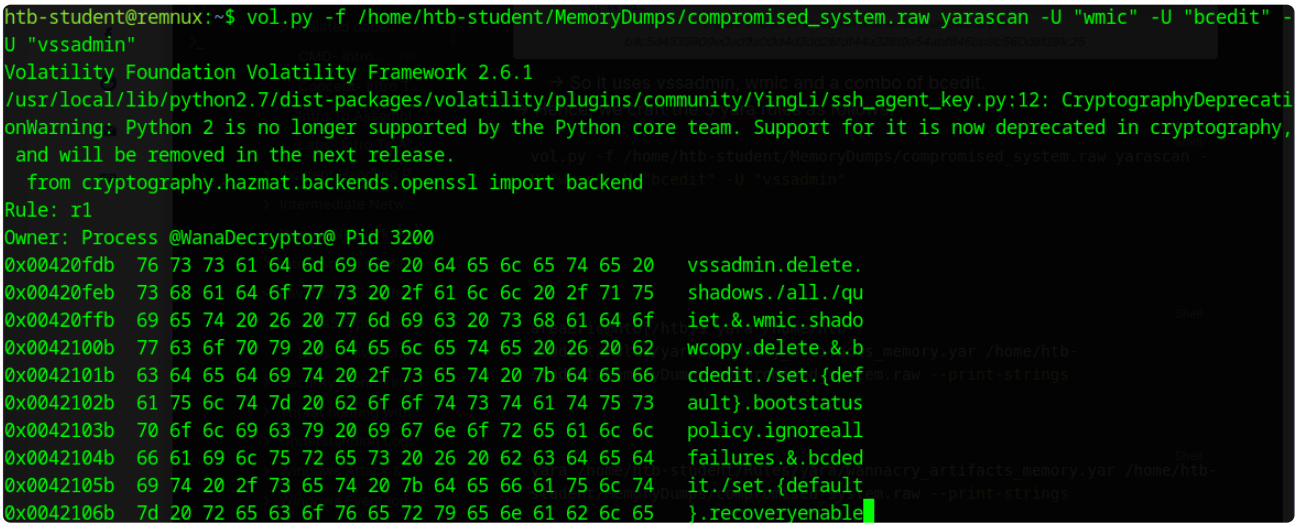
  ```
                    s_/c_vssadmin_delete_shadows_/all_/_00420fd8  XREF[2]:   004066fe(*), 0040670c(R)
  00420fd8 2f 63 20      ds        "/c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /se
           76 73 73
           61 64 6d...
  ```

  Figure 7: Use of vssadmin and wmic shadowcopy in b9c5d4339809e0ad9a00d4d3dd26fdf44a32819a54abf846bb9b560d81391c25

-> So it uses vssadmin, wmic and a combo of bcedit.

-> Hence, we craft the 3 yara rules as follows:

```
vol.py -f /home/htb-student/MemoryDumps/compromised_system.raw yarascan
-U "wmic" -U "bcedit" -U "vssadmin"
```



-> And we identified the process that is responsible for vs deletion is @WanaDecryptor@ with pid of 3200.

Developing Sigma Rules

Question

- Using sigmac translate the "C:\Tools\chainsaw\sigma\rules\windows\builtin\windefend\win*defender_threat.yml" Sigma rule into the equivalent PowerShell command. Then, execute the PowerShell command against "C:\Events\YARASigma\lab_events_4.evtx" and enter the malicious driver as your answer. Answer format:* .sys

-> Take a look at the rule:

```
cat
C:\Tools\chainsaw\sigma\rules\windows\builtin\windefend\win_defender_thr
eat.yml
```

```
PS C:\Windows\system32> cat C:\Tools\chainsaw\sigma\rules\windows\builtin\windefend\win_defender_threat.yml
title: Windows Defender Threat Detected
id: 57b649ef-ff42-4fb0-8bf6-62da243a1708
status: stable
description: Detects all actions taken by Windows Defender malware detection engines
references:
    - https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-antivirus/troubleshoot-window
s-defender-antivirus
author: JÃ¡n TrenÄanskÃ½
date: 2020/07/28
tags:
    - attack.execution
    - attack.t1059
logsource:
    product: windows
    service: windefend
detection:
    selection:
        EventID:
            - 1006 # The antimalware engine found malware or other potentially unwanted software.
            - 1116 # The antimalware platform detected malware or other potentially unwanted software.
            - 1015 # The antimalware platform detected suspicious behavior.
            - 1117 # he antimalware platform performed an action to protect your system from malware or other potentiall
y unwanted software.
    condition: selection
falsepositives:
    - Unlikely
level: high
```

-> We see a yara rule defined for windows security log on malware find.

-> Converting to powershell equivalent command

```
python sigmac -t powershell
'C:\Tools\chainsaw\sigma\rules\windows\builtin\windefend\win_defender_th
reat.yml'
```

```
PS C:\Tools\sigma-0.21\tools> python sigmac -t powershell 'C:\Tools\chainsaw\sigma\rules\windows\builtin\windefend\win_d
efender_threat.yml'
Get-WinEvent | where {($_.ID -eq "1006" -or $_.ID -eq "1116" -or $_.ID -eq "1015" -or $_.ID -eq "1117") } | select TimeC
reated,Id,RecordId,ProcessId,MachineName,Message
```

-> Outputted powershell script

```
Get-WinEvent | where {($_.ID -eq "1006" -or $_.ID -eq "1116" -or $_.ID -
eq "1015" -or $_.ID -eq "1117") } | select
TimeCreated,Id,RecordId,ProcessId,MachineName,Message
```

-> Now we add the path we want to look for the event log

```
Get-WinEvent -path 'C:\Events\YARASigma\lab_events_4.evtx' | where
{($_.ID -eq "1006" -or $_.ID -eq "1116" -or $_.ID -eq "1015" -or $_.ID -
eq "1117") } | select
TimeCreated,Id,RecordId,ProcessId,MachineName,Message
```

```
Get-WinEvent | where {($_.ID -eq "1006" -or $_.ID -eq "1116" -or $_.ID -eq "1015" -or $_.ID -eq "1117") } | select TimeC
reated,Id,RecordId,ProcessId,MachineName,Message
PS C:\Tools\sigma-0.21\tools> Get-WinEvent -path 'C:\Events\YARASigma\lab_events_4.evtx' | where {($_.ID -eq "1006" -or
$_.ID -eq "1116" -or $_.ID -eq "1015" -or $_.ID -eq "1117") } | select TimeCreated,Id,RecordId,ProcessId,MachineName,Mes
sage


TimeCreated : 12/11/2020 4:28:44 AM
Id          : 1116
RecordId    : 177
ProcessId   : 4172
MachineName : WIN10-client01.offsec.lan
Message     : Microsoft Defender Antivirus has detected malware or other potentially unwanted software.
               For more information please see the following:
              1
                Name: High
                ID: 4
                Severity: Tool
                Category: 1
                Path: Suspended
                Detection Origin: Concrete
                Detection Type: 0x00000000
                Detection Source: file:_C:\Users\admmig\Documents\mimikatz.exe
                User: Local machine
                Process Name: 1
                Security intelligence Version: %41
                Engine Version: %42

TimeCreated : 12/11/2020 4:28:44 AM
Id          : 1117
RecordId    : 176
ProcessId   : 4172
MachineName : WIN10-client01.offsec.lan
Message     : Microsoft Defender Antivirus has taken action to protect this machine from malware or other potentially
```

-> We found some matches using the powershell script.

-> The sys file we are interested in is mimidrv.sys, which seems to be the malicious driver.

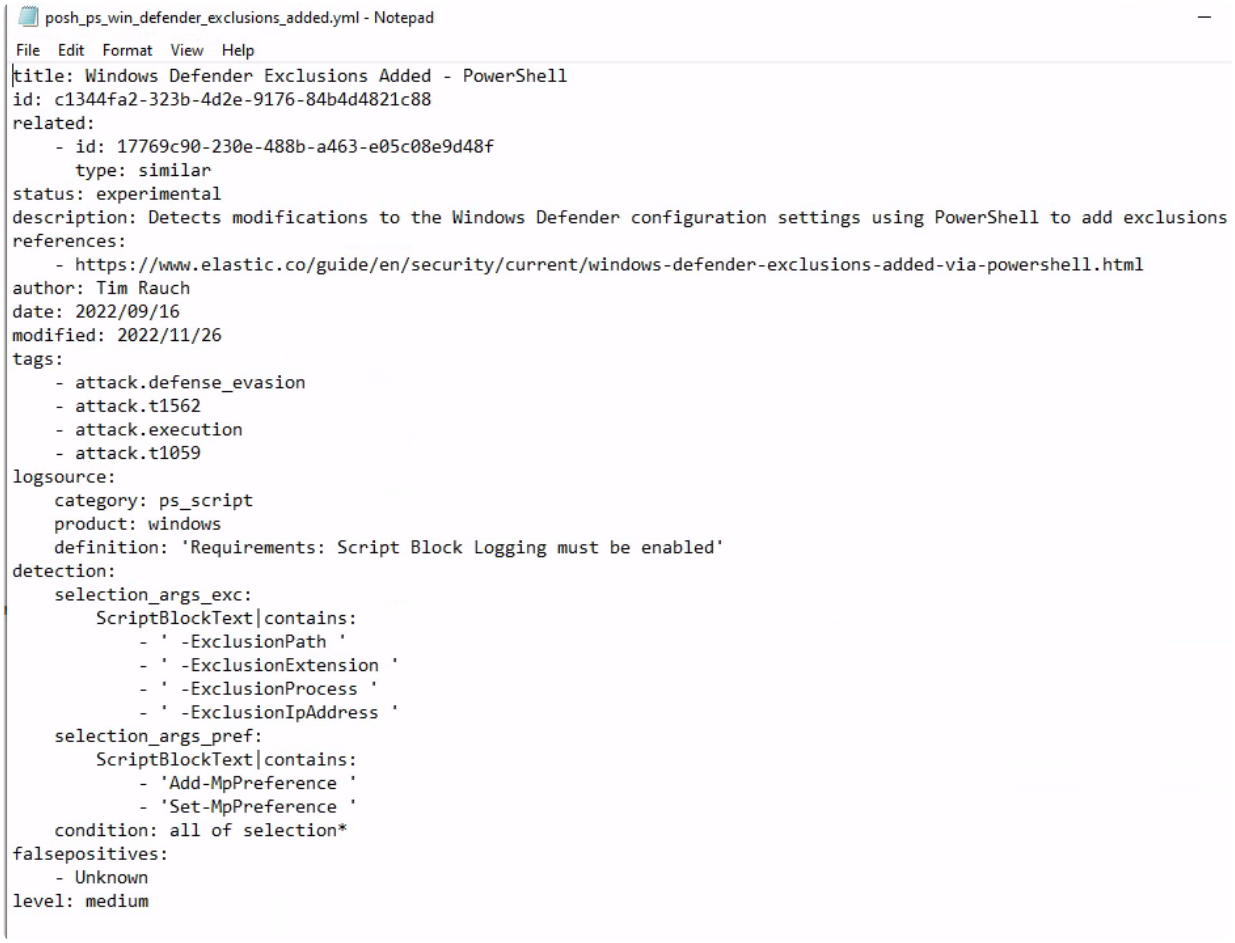
```
TimeCreated : 12/11/2020 4:28:01 AM
Id          : 1116
RecordId    : 173
ProcessId   : 4172
MachineName : WIN10-client01.offsec.lan
Message     : Microsoft Defender Antivirus has detected malware or other potentially unwanted software.
               For more information please see the following:
              1
                Name: High
                ID: 4
                Severity: Tool
                Category: 1
                Path: Suspended
                Detection Origin: Concrete
                Detection Type: 0x00000000
                Detection Source: file:_C:\Users\admmig\Documents\mimidrv.sys;
              file:_C:\Users\admmig\Documents\mimilib.dll
                User: Local machine
                Process Name: 1
                Security intelligence Version: %41
                Engine Version: %42
```

Hunting Evil with Sigma (Chainsaw Edition)

Question

- Use Chainsaw with the "C:\Tools\chainsaw\sigma\rules\windows\powershell\powershell_script\posh_ps_win_defender_exclusions_added.yml" Sigma rule to hunt for suspicious Defender exclusions inside "C:\Events\YARASigma\lab_events_5.evtx". Enter the excluded directory as your answer.

-> Observe the sigma file:

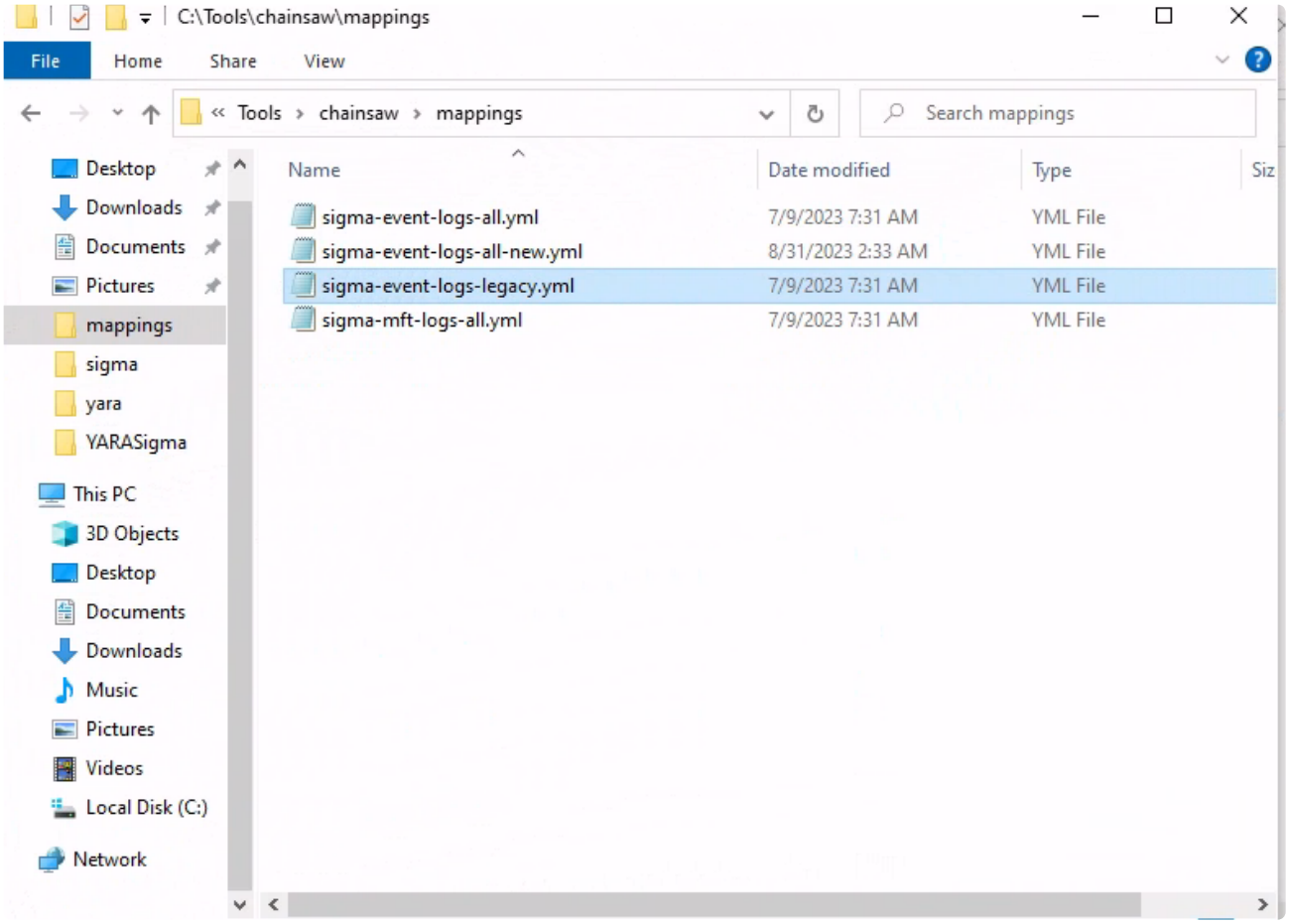


posh_ps_win_defender_exclusions_added.yml - Notepad

File Edit Format View Help

```
title: Windows Defender Exclusions Added - PowerShell
id: c1344fa2-323b-4d2e-9176-84b4d4821c88
related:
    - id: 17769c90-230e-488b-a463-e05c08e9d48f
      type: similar
status: experimental
description: Detects modifications to the Windows Defender configuration settings using PowerShell to add exclusions
references:
    - https://www.elastic.co/guide/en/security/current/windows-defender-exclusions-added-via-powershell.html
author: Tim Rauch
date: 2022/09/16
modified: 2022/11/26
tags:
    - attack.defense_evasion
    - attack.t1562
    - attack.execution
    - attack.t1059
logsource:
    category: ps_script
    product: windows
    definition: 'Requirements: Script Block Logging must be enabled'
detection:
    selection_args_exc:
        ScriptBlockText|contains:
            - ' -ExclusionPath '
            - ' -ExclusionExtension '
            - ' -ExclusionProcess '
            - ' -ExclusionIpAddress '
    selection_args_pref:
        ScriptBlockText|contains:
            - 'Add-MpPreference '
            - 'Set-MpPreference '
    condition: all of selection*
falsepositives:
    - Unknown
level: medium
```

-> We consult the mapping file directory



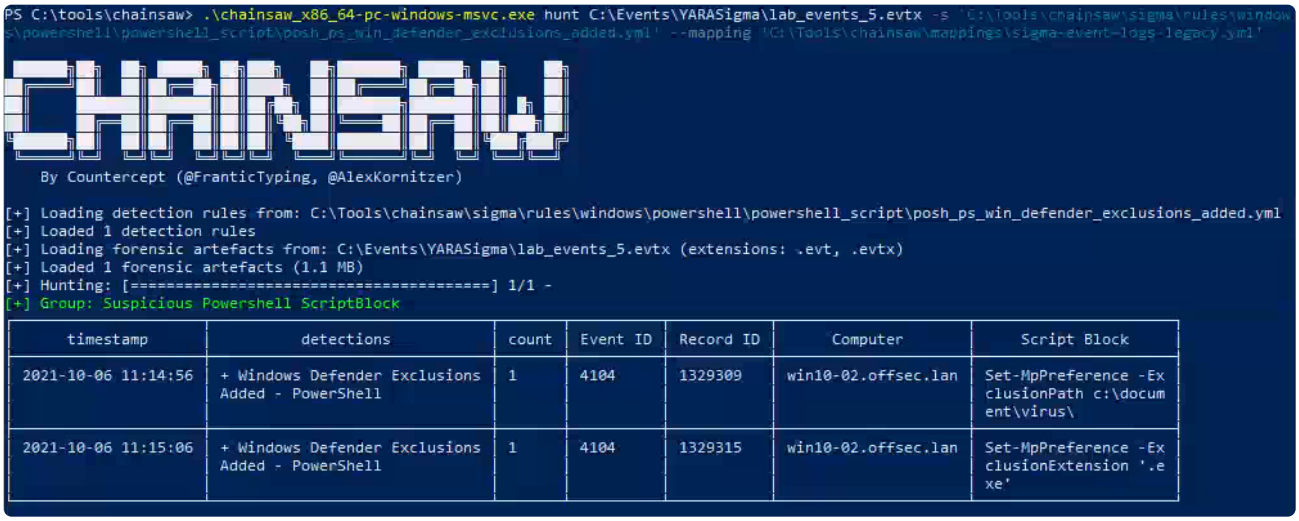
-> And we see that sigma-event-logs-legacy.yml has

```
- name: Suspicious Powershell ScriptBlock
  timestamp: Event.System.TimeCreated
  filter:
    int(EventID): 4104
    Provider: Microsoft-Windows-PowerShell
  fields:
    - from: Provider
      to: Event.System.Provider
      visible: false
    - name: Event ID
      from: EventID
      to: Event.System.EventID
    - name: Record ID
      from: EventRecordID
      to: Event.System.EventRecordID
    - name: Computer
      from: Computer
      to: Event.System.Computer
    - name: Script Block
      from: ScriptBlockText
      to: Event.EventData.ScriptBlockText
```

-> Which seems to be doing the correct conversion (ScriptBlockText -> Event.EvetData.ScriptBlockText) and the provider of Microsoft-Windows-Powershell

belongs to ps_script.
-> Hence, our powershell script would look like the following

```
.\chainsaw_x86_64-pc-windows-msvc.exe hunt
C:\Events\YARASigma\lab_events_5.evtx -s
'C:\Tools\chainsaw\sigma\rules\windows\powershell\powershell_script\posh
_ps_win_defender_exclusions_added.yml' --mapping
'C:\Tools\chainsaw\mappings\sigma-event-logs-legacy.yml'
```



-> So the excluded path is C:\document\virus\

Hunting Evil with Sigma (Splunk Edition)

Question

- Using sigmac translate the "C:\Rules\sigma\file_event_win_app_dropping_archive.yml" Sigma rule into the equivalent Splunk search. Then, navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and submit the Splunk search sigmac provided. Enter the TargetFilename value of the returned event as your answer.
  -> We craft the splunk query using sigma as follows

```
python sigmac -t splunk
C:\Rules\sigma\file_event_win_app_dropping_archive.yml -c
.\config\splunk-windows.yml
```

```
PS C:\tools\sigma-0.21\tools> python sigmac -t splunk C:\Rules\sigma\file_event_win_app_dropping_archive.yml -c .\config
\splunk-windows.yml
((Image="*\\winword.exe" OR Image="*\\excel.exe" OR Image="*\\powerpnt.exe" OR Image="*\\msaccess.exe" OR Image="*\\mspu
b.exe" OR Image="*\\eqnedt32.exe" OR Image="*\\visio.exe" OR Image="*\\wordpad.exe" OR Image="*\\wordview.exe" OR Image=
"*\\certutil.exe" OR Image="*\\certoc.exe" OR Image="*\\CertReq.exe" OR Image="*\\Desktopimgdownldr.exe" OR Image="*\\es
entutl.exe" OR Image="*\\finger.exe" OR Image="*\\notepad.exe" OR Image="*\\AcroRd32.exe" OR Image="*\\RdrCEF.exe" OR Im
age="*\\mshta.exe" OR Image="*\\hh.exe" OR Image="*\\sharphound.exe") (TargetFilename="*.zip" OR TargetFilename="*.rar"
OR TargetFilename="*.7z" OR TargetFilename="*.diagcab" OR TargetFilename="*.appx"))
```

-> We get the following splunk query

```
((Image="*\\winword.exe" OR Image="*\\excel.exe" OR
Image="*\\powerpnt.exe" OR Image="*\\msaccess.exe" OR
Image="*\\mspub.exe" OR Image="*\\eqnedt32.exe" OR Image="*\\visio.exe"
OR Image="*\\wordpad.exe" OR Image="*\\wordview.exe" OR
Image="*\\certutil.exe" OR Image="*\\certoc.exe" OR
Image="*\\CertReq.exe" OR Image="*\\Desktopimgdownldr.exe" OR
Image="*\\esentutl.exe" OR Image="*\\finger.exe" OR
Image="*\\notepad.exe" OR Image="*\\AcroRd32.exe" OR
Image="*\\RdrCEF.exe" OR Image="*\\mshta.exe" OR Image="*\\hh.exe" OR
Image="*\\sharphound.exe") (TargetFilename="*.zip" OR
TargetFilename="*.rar" OR TargetFilename="*.7z" OR
TargetFilename="*.diagcab" OR TargetFilename="*.appx"))
```

-> Searching it we get

i	Time	Event
>	11/8/22 11:27:19.000 AM	11/08/2022 11:27:19 AM LogName=Microsoft-Windows-Sysmon/Operational EventCode=11 EventType=4 ComputerName=DESKTOP-EGSS5IS.uniwaldo.local User=NOT_TRANSLATED Sid=S-1-5-18 SidType=0 SourceName=Microsoft-Windows-Sysmon Type=Information RecordNumber=51177 Keywords=None TaskCategory=File created (rule: FileCreate) OpCode=Info Message=File created: RuleName: - UtcTime: 2022-11-08 19:27:19.859 ProcessGuid: {96192a2a-ad68-636a-2905-000000000d00} ProcessId: 4352 Image: C:\Users\waldo\Downloads\SharpHound.exe TargetFilename: C:\Users\waldo\Downloads\20221108112718_BloodHound.zip CreationUtcTime: 2022-11-08 19:27:19.859 User: NT AUTHORITY\SYSTEM Collapse host = DESKTOP-EGSS5IS | source = WinEventLogSysmon_DESKTOP-EGSS5IS.txt | sourcetype = WinEventLog:Sysmon

-> Typical! An sharphound running to create a bloodhound zip file for attack path analysis.

Skills Assessment

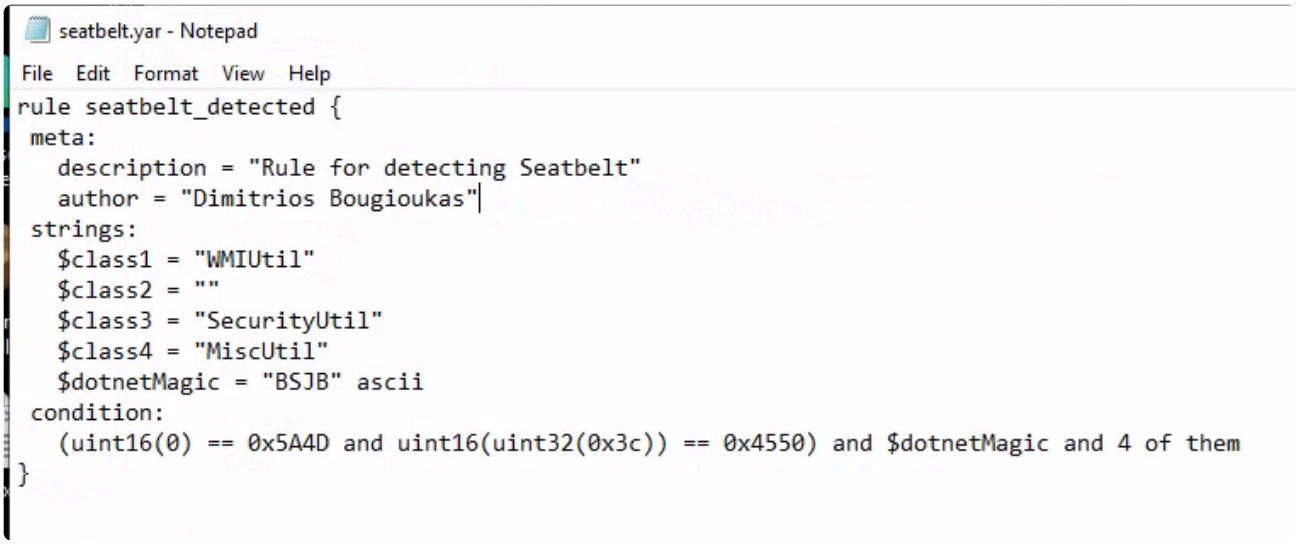
Scenario

- In this skills assessment section, we'll practice YARA rule development and using Sigma rules to hunt for threats within event logs.
- For the initial question, you'll be tasked with developing a YARA rule aimed at identifying the malicious `Seatbelt.exe` file, commonly used by attackers for maintaining operational security.
- In the subsequent question, you'll be using a Sigma rule to identify instances of shadow volume deletion - a technique often utilized by ransomware groups.

Questions

- The "C:\Rules\yara\seatbelt.yar" YARA rule aims to detect instances of the "Seatbelt.exe" .NET assembly on disk. Analyze both "C:\Rules\yara\seatbelt.yar" and "C:\Samples\YARASigma\Seatbelt.exe" and specify the appropriate string inside the "$class2" variable so that the rule successfully identifies "C:\Samples\YARASigma\Seatbelt.exe". Answer format: L_____r

-> We observe the yara file and see that it seems to be related to the util file.



-> We research the seatbelt.exe executable and see that it made from C#, hence it is an .Net Executable which uses the .Net framework.

README License

Seatbelt is a C# project that performs a number of security oriented host-survey "safety checks" relevant from both offensive and defensive security perspectives.

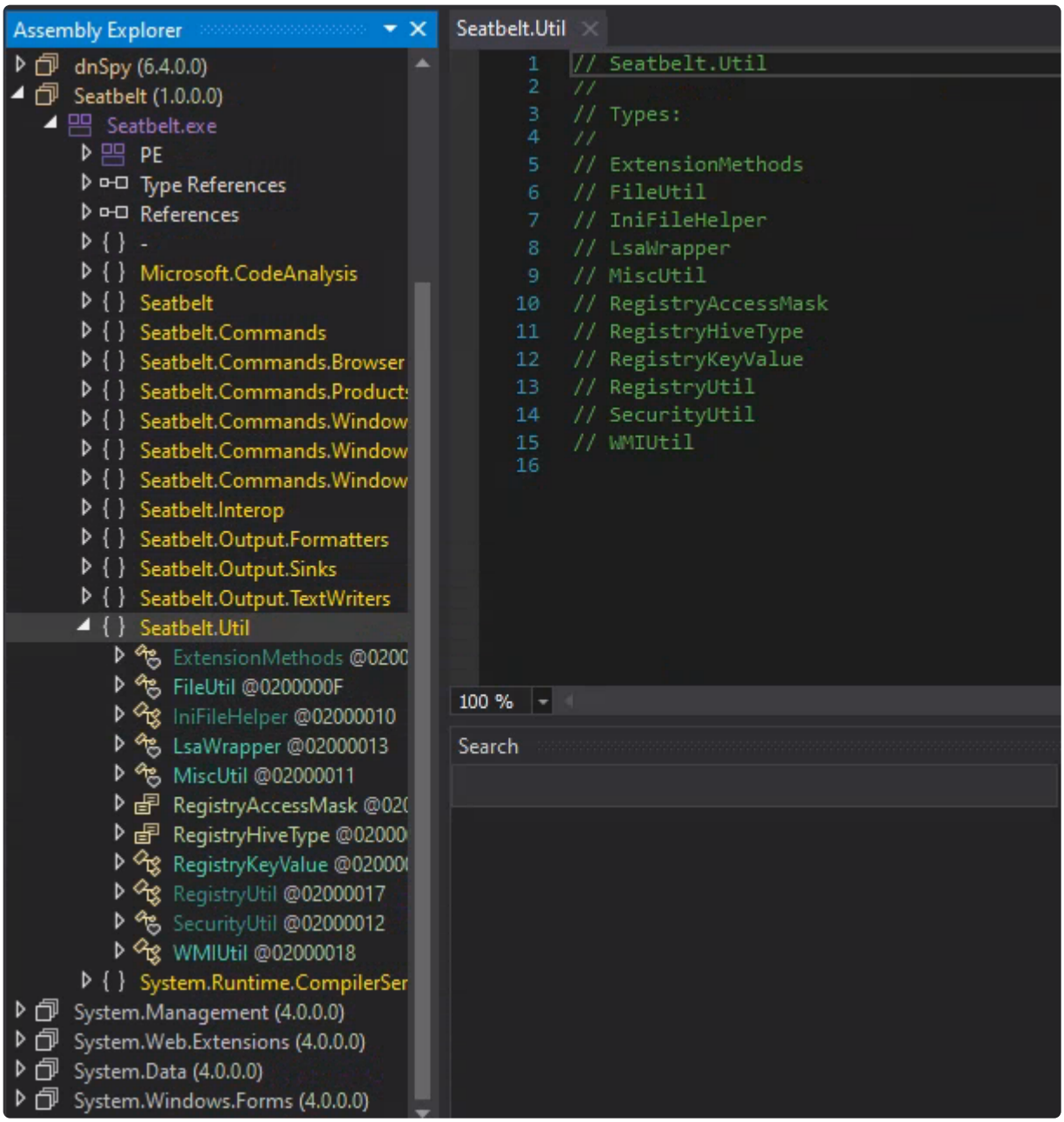
@andrewchiles' HostEnum.ps1 script and @tifkin_'s Get-HostProfile.ps1 provided inspiration for many of the artifacts to collect.

@harmj0y and @tifkin_ are the primary authors of this implementation.

Seatbelt is licensed under the BSD 3-Clause license.

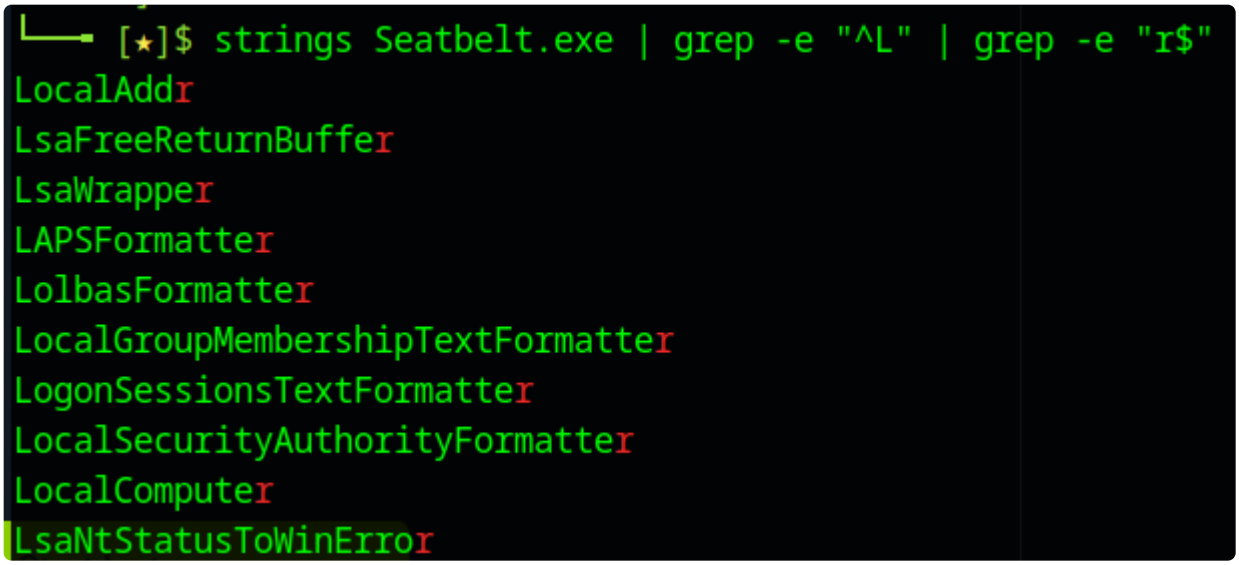
https://github.com/GhostPack/Seatbelt

-> We reverse the .Net executable (since its a program made from C#) using dnSpy.



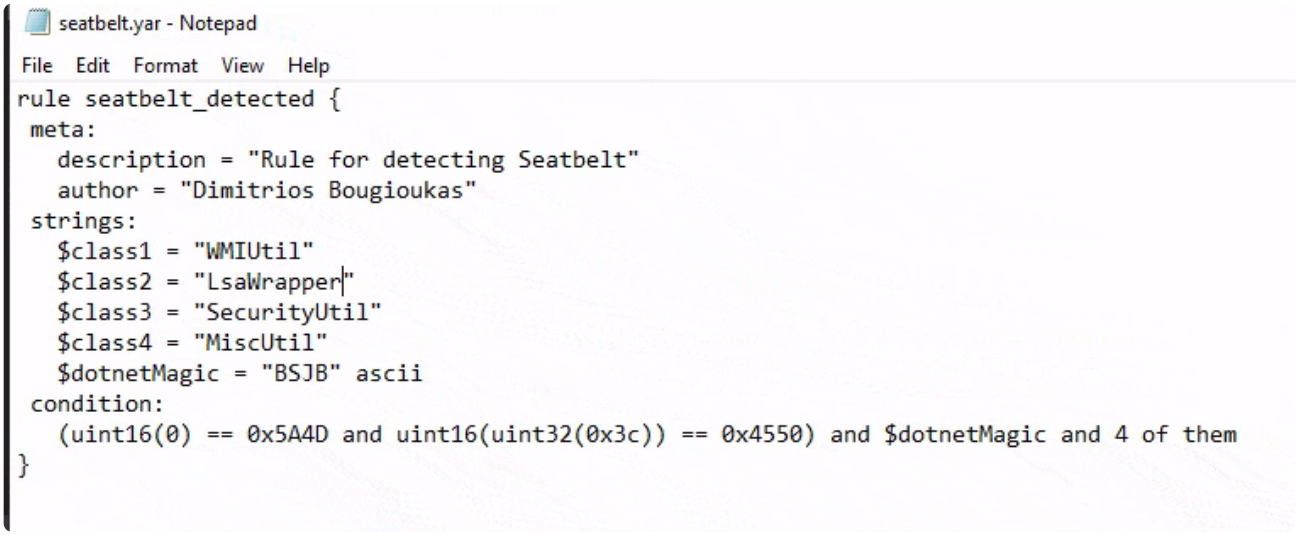
- We see that the .Seatbelt.exe has an utility class which contained the 3 Util classes in the yara rule file.
- So it seems like it's LsaWrapper and verify it as follows:

```
strings Seatbelt.exe | grep -e "^L" | grep -e "r$"
```



-> Note: We delete Seatbelt.exe from the machine after the command above.

-> Indeed, we can confirm that it is LsaWrapper and let's put it in in the Yara file.



-> Lastly, we execute the yara file and confirm it

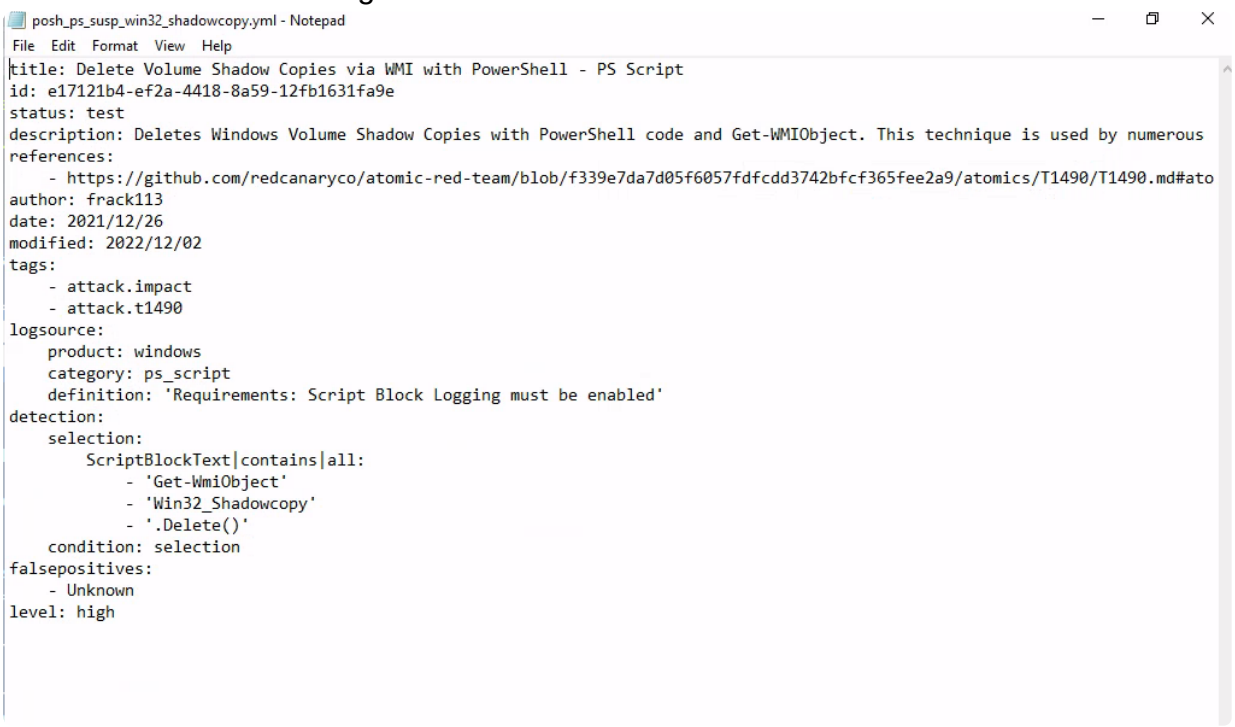
```
yara64.exe -s C:\Rules\yara\seatbelt.yar
C:\Samples\YARASigma\Seatbelt.exe

yara64.exe C:\Rules\yara\seatbelt.yar C:\Samples\YARASigma\
```

```
PS C:\Samples\YARASigma> yara64.exe -s C:\Rules\yara\seatbelt.yar C:\Samples\YARASigma\Seatbelt.exe
seatbelt_detected C:\Samples\YARASigma\Seatbelt.exe
0x68299:$class1: WMIUtil
0x69f18:$class2: LsaWrapper
0x682c0:$class3: SecurityUtil
0x682a1:$class4: MiscUtil
0x2b068:$dotnetMagic: BSJB
```

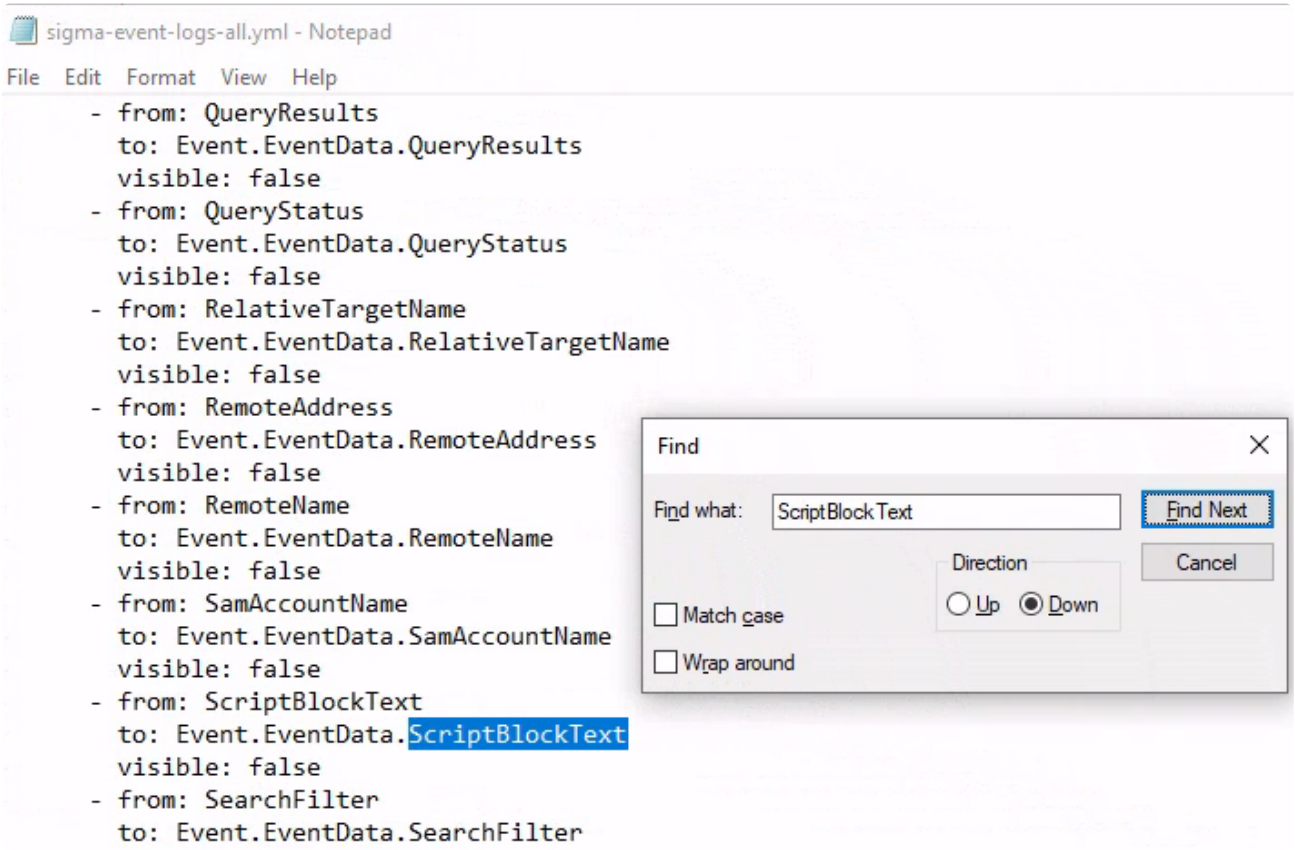
```
PS C:\Samples\YARASigma> yara64.exe C:\Rules\yara\seatbelt.yar C:\Samples\YARASigma\
seatbelt_detected C:\Samples\YARASigma\\Seatbelt.exe
PS C:\Samples\YARASigma>
```

- Use Chainsaw with the "C:\Tools\chainsaw\sigma\rules\windows\powershell\powershell_script\posh_ps_susp_win32_shadowcopy.yml" Sigma rule to hunt for shadow volume deletion inside "C:\Events\YARASigma\lab_events_6.evtx". Enter the identified ScriptBlock ID as your answer.
  -> We first observe the sigma file



  -> So we see it's Script Block Text must contain all of Get-WmiObject, Win32_Shadowcopy and .Delete()

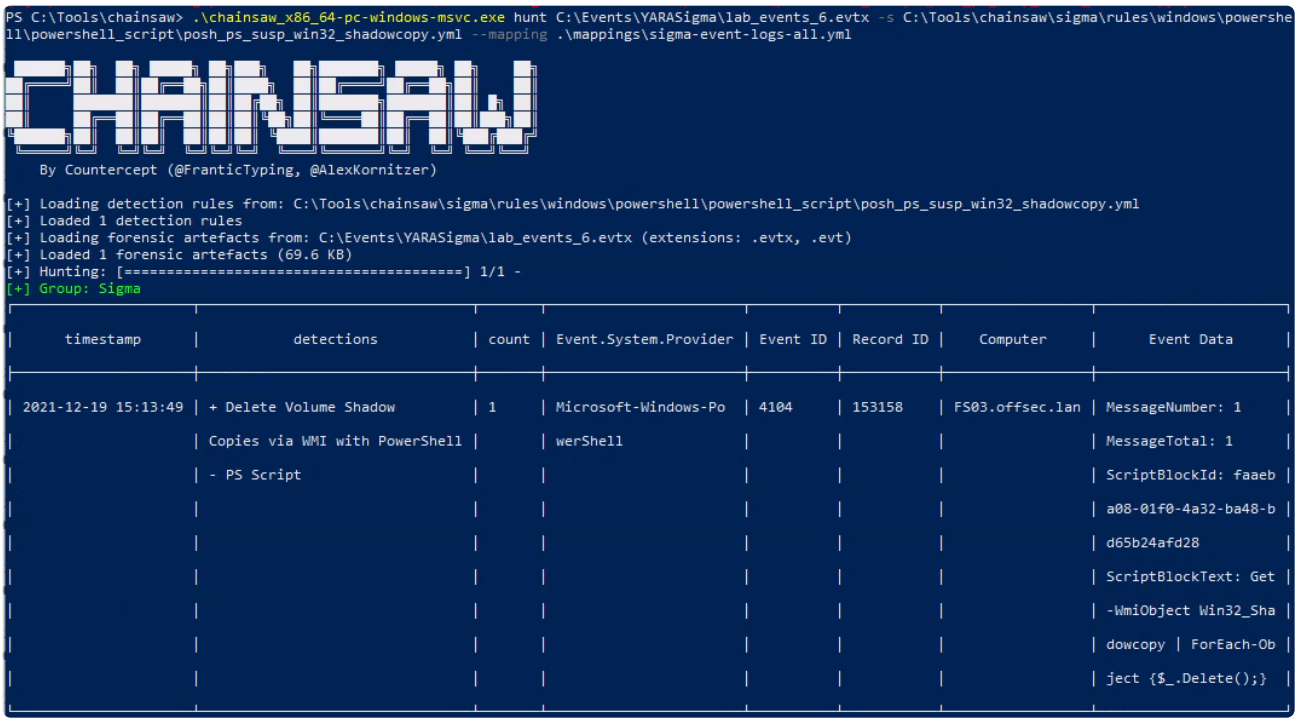
-> We now check the mapping file for chainsaw:



-> Hence, we'll use this as the mapping file.

-> We now perform the hunt using chainsaw

```
.\chainsaw_x86_64-pc-windows-msvc.exe hunt
C:\Events\YARASigma\lab_events_6.evtx -s
C:\Tools\chainsaw\sigma\rules\windows\powershell\powershell_script\posh_
ps_susp_win32_shadowcopy.yml --mapping .\mappings\sigma-event-logs-
all.yml
```



-> Hence, the script block id is

```
faaeba08-01f0-4a32-ba48-bd65b24afd28
```

-> And we can see all the shadowcopy object in WMI service is getting deleted.