Creating Detection Rules

Having now uncovered the Tactics, Techniques, and Procedures (TTPs) employed by this malware, we can proceed to design detection rules, such as `Yara` and `Sigma` rules.

While we will begin to delve into the concepts of Yara and Sigma rule development in this section, we'll only scratch the surface. These are extensive topics with a lot of depth, necessitating a comprehensive study. Hence, we will be dedicating a complete module named 'YARA & Sigma for SOC Analysts' to help you truly master these crucial areas of cyber defense.

Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, let's SSH into the Target IP using the provided credentials. The vast majority of the actions/commands covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

Yara

`YARA (Yet Another Recursive Acronym)`, a widely used open-source pattern matching tool and rule-based malware detection and classification framework lets us create custom rules to spot specific patterns or characteristics in files, processes, or memory. To draft a `YARA` rule for our sample, we'll need to examine the behavior, features, or specific strings/patterns unique to the sample we aim to detect.

Here's a simple example of a `YARA` rule that matches the presence of the string `Sandbox detected` in a process. We remind you that `shell.exe` demonstrated such behavior.

Code: yara

```
rule Shell_Sandbox_Detection {
    strings:
        $sandbox_string = "Sandbox detected"
    condition:
        $sandbox_string
}
```

Now let's add a lot more strings and patterns into the rule to make it better.

We can utilize the `yarGen` tool, which automates the process of generating `YARA` rules, with the prime objective of crafting the best possible rules for manual post-processing. This, however, necessitates a shrewd automatic preselection and a discerning human analyst to generate a robust rule.

First let's create a new directory called `Test` inside the `/home/htb-student/Samples/MalwareAnalysis` directory of this section's target and then let's copy `shell.exe` (residing in the `/home/htb-student/Samples/MalwareAnalysis` directory) to the newly created `Test` directory as follows.

Creating Detection Rules

```
MisaelMacias@htb[/htb]$ mkdir /home/htb-student/Samples/MalwareAnalysis/Test
```

Creating Detection Rules

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

```
MisaelMacias@htb[/htb]$ cp /home/htb-student/Samples/MalwareAnalysis/shell.exe /home/htb-student/Sa
```

To automatically create a Yara rule for `shell.exe` we should execute the following (inside the `/home/htb-student/yarGen-0.23.4` directory).

Creating Detection Rules

```
MisaelMacias@htb[/htb]$ sudo python3 yarGen.py -m /home/htb-student/Samples/MalwareAnalysis/Test/
--------------------------------------------------------------------------
                    _____
    __ _____ _____/ ___/__ ___
   / // / _ `/ __/ (_ / -_) _ \
   \_, /\_,_/_/  \___/\__/_//_/
  /___/  Yara Rule Generator
         Florian Roth, July 2020, Version 0.23.3

  Note: Rules have to be post-processed
  See this post for details: https://medium.com/@cyb3rops/121d29322282
--------------------------------------------------------------------------
[+] Using identifier 'Test'
[+] Using reference 'https://github.com/Neo23x0/yarGen'
[+] Using prefix 'Test'
[+] Processing PEStudio strings ...
[+] Reading goodware strings from database 'good-strings.db' ...
    (This could take some time and uses several Gigabytes of RAM depending on your db size)
[+] Loading ./dbs/good-imphashes-part3.db ...
[+] Total: 4029 / Added 4029 entries
[+] Loading ./dbs/good-strings-part9.db ...
[+] Total: 788 / Added 788 entries
[+] Loading ./dbs/good-strings-part8.db ...
[+] Total: 332082 / Added 331294 entries
[+] Loading ./dbs/good-imphashes-part4.db ...
[+] Total: 6426 / Added 2397 entries
[+] Loading ./dbs/good-strings-part2.db ...
[+] Total: 1703601 / Added 1371519 entries
[+] Loading ./dbs/good-exports-part2.db ...
[+] Total: 90960 / Added 90960 entries
[+] Loading ./dbs/good-strings-part4.db ...
[+] Total: 3860655 / Added 2157054 entries
[+] Loading ./dbs/good-exports-part4.db ...
[+] Total: 172718 / Added 81758 entries
[+] Loading ./dbs/good-exports-part7.db ...
[+] Total: 223584 / Added 50866 entries
[+] Loading ./dbs/good-strings-part6.db ...
[+] Total: 4571266 / Added 710611 entries
[+] Loading ./dbs/good-strings-part7.db ...
[+] Total: 5828908 / Added 1257642 entries
[+] Loading ./dbs/good-exports-part1.db ...
[+] Total: 293752 / Added 70168 entries
[+] Loading ./dbs/good-exports-part3.db ...
[+] Total: 326867 / Added 33115 entries
[+] Loading ./dbs/good-imphashes-part9.db ...
[+] Total: 6426 / Added 0 entries
[+] Loading ./dbs/good-exports-part9.db ...
[+] Total: 326867 / Added 0 entries
[+] Loading ./dbs/good-imphashes-part5.db ...
[+] Total: 13764 / Added 7338 entries
[+] Loading ./dbs/good-imphashes-part8.db ...
[+] Total: 13947 / Added 183 entries
[+] Loading ./dbs/good-imphashes-part6.db ...
[+] Total: 13976 / Added 29 entries
[+] Loading ./dbs/good-strings-part1.db ...
[+] Total: 6893854 / Added 1064946 entries
[+] Loading ./dbs/good-imphashes-part7.db ...
[+] Total: 17382 / Added 3406 entries
[+] Loading ./dbs/good-exports-part6.db ...
[+] Total: 328525 / Added 1658 entries
[+] Loading ./dbs/good-imphashes-part2.db ...
[+] Total: 18208 / Added 826 entries
[+] Loading ./dbs/good-exports-part8.db ...
[+] Total: 332359 / Added 3834 entries
[+] Loading ./dbs/good-strings-part3.db ...
[+] Total: 9152616 / Added 2258762 entries
[+] Loading ./dbs/good-strings-part5.db ...
[+] Total: 12284943 / Added 3132327 entries
[+] Loading ./dbs/good-imphashes-part1.db ...
[+] Total: 19764 / Added 1556 entries
[+] Loading ./dbs/good-exports-part5.db ...
[+] Total: 404321 / Added 71962 entries
```

```
[+] Processing malware files ...
[+] Processing /home/htb-student/Samples/MalwareAnalysis/Test/shell.exe ...
[+] Generating statistical data ...
[+] Generating Super Rules ... (a lot of magic)
[+] Generating Simple Rules ...
[-] Applying intelligent filters to string findings ...
[-] Filtering string set for /home/htb-student/Samples/MalwareAnalysis/Test/shell.exe ...
[=] Generated 1 SIMPLE rules.
[=] All rules written to yargen_rules.yar
[+] yarGen run finished
```

We will notice that a file named `yargen_rules.yar` is generated by `yarGen` that incorporates unique strings, which are automatically extracted and inserted into the rule.

Creating Detection Rules

```
MisaelMacias@htb[/htb]$ cat yargen_rules.yar
/*
   YARA Rule Set
   Author: yarGen Rule Generator
   Date: 2023-08-02
   Identifier: Test
   Reference: https://github.com/Neo23x0/yarGen
*/

/* Rule Set ----------------------------------------------------------------- */

rule _home_htb_student_Samples_MalwareAnalysis_Test_shell {
   meta:
      description = "Test - file shell.exe"
      author = "yarGen Rule Generator"
      reference = "https://github.com/Neo23x0/yarGen"
      date = "2023-08-02"
      hash1 = "bd841e796feed0088ae670284ab991f212cf709f2391310a85443b2ed1312bda"
   strings:
      $x1 = "C:\\Windows\\System32\\cmd.exe" fullword ascii
      $s2 = "http://ms-windows-update.com/svchost.exe" fullword ascii
      $s3 = "C:\\Windows\\System32\\notepad.exe" fullword ascii
      $s4 = "/k ping 127.0.0.1 -n 5" fullword ascii
      $s5 = "iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com" fullword ascii
      $s6 = "  VirtualQuery failed for %d bytes at address %p" fullword ascii
      $s7 = "[-] Error code is : %lu" fullword ascii
      $s8 = "C:\\Program Files\\VMware\\VMware Tools\\" fullword ascii
      $s9 = "Failed to open the registry key." fullword ascii
      $s10 = "  VirtualProtect failed with code 0x%x" fullword ascii
      $s11 = "Connection sent to C2" fullword ascii
      $s12 = "VPAPAPAPI" fullword ascii
      $s13 = "AWAVAUATVSH" fullword ascii
      $s14 = "45.33.32.156" fullword ascii
      $s15 = "  Unknown pseudo relocation protocol version %d." fullword ascii
      $s16 = "AQAPRQVH1" fullword ascii
      $s17 = "connect" fullword ascii /* Goodware String - occured 429 times */
      $s18 = "socket" fullword ascii /* Goodware String - occured 452 times */
      $s19 = "tSIcK<L" fullword ascii
      $s20 = "Windows-Update/7.6.7600.256 %s" fullword ascii
   condition:
      uint16(0) == 0x5a4d and filesize < 60KB and
      1 of ($x*) and 4 of them
}
```

We can review the rule and modify it as necessary, adding more strings and conditions to enhance its reliability and effectiveness.

Detecting Malware Using Yara Rules

We can then use this rule to scan a directory as follows.

Creating Detection Rules

```
MisaelMacias@htb[/htb]$ yara /home/htb-student/yarGen-0.23.4/yargen_rules.yar /home/htb-student/Sam
home_htb_student_Samples_MalwareAnalysis_Test_shell /home/htb-student/Samples/MalwareAnalysis//shel
```

We will notice that `shell.exe` is returned!

Below are some references for YARA rules:

- Yara documentation : https://yara.readthedocs.io/en/stable/writingrules.html
- Yara resources - https://github.com/InQuest/awesome-yara
- The DFIR Report - https://github.com/The-DFIR-Report/Yara-Rules

Sigma

`Sigma` is a comprehensive and standardized rule format extensively used by security analysts and `Security Information and Event Management (SIEM)` systems. The objective is to detect and identify specific patterns or behaviors that could potentially signify security threats or events. The standardized format of `Sigma` rules enables security teams to define and disseminate detection logic across diverse security platforms.

To construct a `Sigma` rule based on certain actions - for instance, dropping a file in a temporary location - we can devise a sample rule along these lines.

Code: sigma

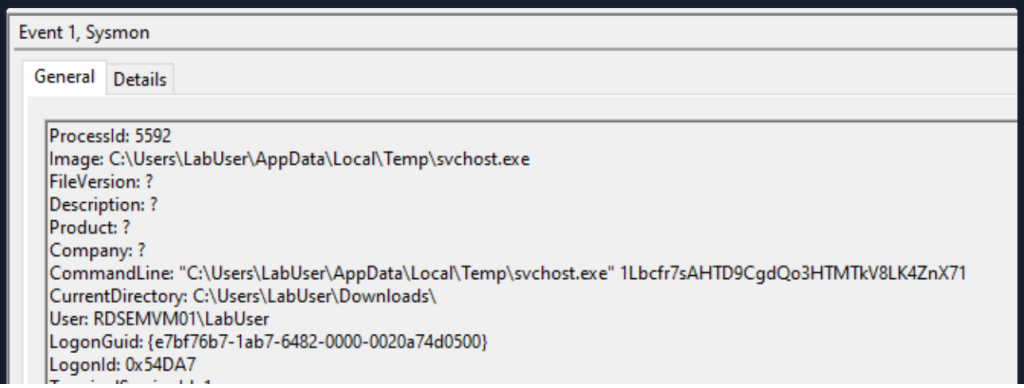
```
title: Suspicious File Drop in Users Temp Location
status: experimental
description: Detects suspicious activity where a file is dropped in the temp location

logsource:
    category: process_creation
detection:
    selection:
        TargetFilename:
            - '*\\AppData\\Local\\Temp\\svchost.exe'
    condition: selection
    level: high

falsepositives:
    - Legitimate exe file drops in temp location
```

In this instance, the rule is designed to identify when the file `svchost.exe` is dropped in the `Temp` directory.

During analysis, it's advantageous to have a system monitoring agent operating continuously. In this context, we've chosen `Sysmon` to gather the logs. `Sysmon` is a powerful tool that captures detailed event data and aids in the creation of `Sigma` rules. Its log categories encompass process creation (`EventID 1`), network connection (`EventID 3`), file creation (`EventID 11`), registry modification (`EventID 13`), among others. The scrutiny of these events assists in pinpointing indicators of compromise (`IOCs`) and understanding behavior patterns, thus facilitating the crafting of effective detection rules.

For instance, `Sysmon` has collected logs such as `process creation`, `process access`, `file creation`, and `network connection`, among others, in response to the activities conducted by `shell.exe`. This compiled information proves instrumental in enhancing our understanding of the sample's behavior and developing more precise and effective detection rules.

Process Create Logs:



TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=D13EAC51CD03EB893DE24FC827B8CDDB,SHA256=14ED6EC0E5BAFA1F567614D08E493D6A918B53E3E659B1
ParentProcessGuid: {e7bf76b7-f057-6489-0000-00109829ea09}
ParentProcessId: 6508
ParentImage: C:\Users\LabUser\Downloads\shell.exe
ParentCommandLine: "C:\Users\LabUser\Downloads\shell.exe"

Process Access Logs (not configured in the Windows targets of this module):

Event 10, Sysmon

General Details

SourceProcessGUID: {e7bf76b7-f057-6489-0000-00109829ea09}
SourceProcessId: 6508
SourceThreadId: 9328
SourceImage: C:\Users\LabUser\Downloads\shell.exe
TargetProcessGUID: {e7bf76b7-f084-6489-0000-0010b8b1ea09}
TargetProcessId: 9576
TargetImage: C:\Windows\System32\notepad.exe
GrantedAccess: 0x1F3FFF
CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9d4c4|C:\Windows\System32\KERNELBASE.dll+2c13e|C:\Users\LabUser\Downloa
\System32\KERNEL32.DLL+17034|C:\Windows\SYSTEM32\ntdll.dll+526a1

File Creation Logs:

Event 11, Sysmon

General Details

ProcessGuid: {e7bf76b7-f057-6489-0000-00109829ea09}
ProcessId: 6508
Image: C:\Users\LabUser\Downloads\shell.exe
TargetFilename: C:\Users\LabUser\AppData\Local\Temp\svchost.exe

Network Connection Logs:

Event 3, Sysmon

General Details

ProcessId: 6508
Image: C:\Users\LabUser\Downloads\shell.exe
User: RDSEMVM01\LabUser
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: 10.10.10.7
SourceHostname: RDSEMVM01
SourcePort: 64966
SourcePortName:
DestinationIsIpv6: false
DestinationIp: 10.10.10.100
DestinationHostname: www.inetsim.org
DestinationPort: 80
DestinationPortName: http

Log Name: Microsoft-Windows-Sysmon/Operation

Event 3, Sysmon

General Details

ProcessGuid: {e7bf76b7-f057-6489-0000-00109829ea09}
ProcessId: 6508
Image: C:\Users\LabUser\Downloads\shell.exe
User: RDSEMVM01\LabUser
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: 10.10.10.7
SourceHostname: RDSEMVM01
SourcePort: 64965
SourcePortName:
DestinationIsIpv6: false
DestinationIp: 45.33.32.156
DestinationHostname: www.inetsim.org
DestinationPort: 31337

Below are some references for Sigma rules:

- Sigma documentation : https://github.com/SigmaHQ/sigma-specification
- Sigma resources - https://github.com/SigmaHQ/sigma/tree/master/rules
- The DFIR Report - https://github.com/The-DFIR-Report/Sigma-Rules/tree/main/rules

PROTOCOL

UDP 1337 TCP 443

DOWNLOAD VPN CONNECTION FILE

Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 140ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Download VPN Connection File

Target(s): Click here to spawn the target system!

SSH to with user "htb-student" and password "HTB_@cademy_stdnt!"

+ 0 Generate a Yara rule automatically for dharma_sample.exe, located in the /home/htb-student/Samples/MalwareAnalysis directory, by utilizing yarGen.py. Following this, employ the created Yara rule for the identification of the malware present in the /home/htb-student/Samples/MalwareAnalysis directory, as an exercise. Enter "Done" when you are done practicing.

Done

Submit

Previous Next

Mark Complete & Next


Powered by HACKTHEBOX