Hunting Evil with Sigma (Chainsaw Edition)

In cybersecurity, time is of the essence. Rapid analysis allows us to not just identify but also respond to threats before they escalate.

When we're up against the clock, racing to find a needle in a haystack of Windows Event Logs without access to a SIEM, Sigma rules combined with tools like Chainsaw and Zircolite are our best allies.

Both tools allow us to use Sigma rules to scan not just one, but multiple EVTX files concurrently, offering a broader and more comprehensive scan in a very efficient manner.

Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, let's RDP 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.

Scanning Windows Event Logs With Chainsaw

Chainsaw is a freely available tool designed to swiftly pinpoint security threats within Windows Event Logs. This tool enables efficient keyword-based event log searches and is equipped with integrated support for Sigma detection rules as well as custom Chainsaw rules. Therefore, it serves as a valuable asset for validating our Sigma rules by applying them to actual event logs. Let's download the Chainsaw from the official Github repository and run it with some sigma rules:

Chainsaw can be found inside the `C:\Tools\chainsaw` directory of this section's target.

Let's first run Chainsaw with `-h` flag to see the help menu.

Hunting Evil with Sigma (Chainsaw Edition)

```
PS C:\Tools\chainsaw> .\chainsaw_x86_64-pc-windows-msvc.exe -h
Rapidly work with Forensic Artefacts

Usage: chainsaw_x86_64-pc-windows-msvc.exe [OPTIONS] <COMMAND>

Commands:
  dump     Dump an artefact into a different format
  hunt     Hunt through artefacts using detection rules for threat detection
  lint     Lint provided rules to ensure that they load correctly
  search   Search through forensic artefacts for keywords
  analyse  Perform various analyses on artifacts
  help     Print this message or the help of the given subcommand(s)

Options:
      --no-banner                Hide Chainsaw's banner
      --num-threads <NUM_THREADS>  Limit the thread number (default: num of CPUs)
  -h, --help                     Print help
  -V, --version                  Print version

Examples:

    Hunt with Sigma and Chainsaw Rules:
        ./chainsaw hunt evtx_attack_samples/ -s sigma/ --mapping mappings/sigma-event-logs-all.yml

    Hunt with Sigma rules and output in JSON:
        ./chainsaw hunt evtx_attack_samples/ -s sigma/ --mapping mappings/sigma-event-logs-all.yml

    Search for the case-insensitive word 'mimikatz':
        ./chainsaw search mimikatz -i evtx_attack_samples/
```

? Go to Questions

Table of Contents

Introduction to YARA & Sigma

Leveraging YARA

YARA and YARA Rules

Developing YARA Rules

Hunting Evil with YARA (Windows Edition)

Hunting Evil with YARA (Linux Edition)

Hunting Evil with YARA (Web Edition)

Leveraging Sigma

Sigma and Sigma Rules

Developing Sigma Rules

Hunting Evil with Sigma (Chainsaw Edition)

Hunting Evil with Sigma (Splunk Edition)

Skills Assessment

Skills Assessment

```
    Search for Powershell Script Block Events (EventID 4014):
        ./chainsaw search -t 'Event.System.EventID: =4104' evtx_attack_samples/
```

Example 1: Hunting for Multiple Failed Logins From Single Source With Sigma

Let's put Chainsaw to work by applying our most recent Sigma rule, `win_security_susp_failed_logons_single_source2.yml` (available at `C:\Rules\sigma`), to `lab_events_2.evtx` (available at `C:\Events\YARASigma\lab_events_2.evtx`) that contains multiple failed login attempts from the same source.

Hunting Evil with Sigma (Chainsaw Edition)

```
PS C:\Tools\chainsaw> .\chainsaw_x86_64-pc-windows-msvc.exe hunt C:\Events\YARASigma\lab_events_2.e

    By Countercept (@FranticTyping, @AlexKornitzer)

[+] Loading detection rules from: C:\Rules\sigma\win_security_susp_failed_logons_single_source2.yml
[+] Loaded 1 detection rules
[+] Loading forensic artefacts from: C:\Events\YARASigma\lab_events_2.evtx (extensions: .evt, .evtx
[+] Loaded 1 forensic artefacts (69.6 KB)
[+] Hunting: [========================================] 1/1 -
[+] Group: Sigma
```

timestamp	detections	count	Event.System.Provider	Event
2021-05-20 12:49:52	+ Failed NTLM Logins with Different Accounts from Single Source System	5	Microsoft-Windows-Security-Auditing	4776

```
[+] 1 Detections found on 1 documents
PS C:\Tools\chainsaw>
```

Our Sigma rule was able to identify the multiple failed login attempts against `NOUSER`.

Using the `-s` parameter, we can specify a directory containing Sigma detection rules (or one Sigma detection rule) and Chainsaw will automatically load, convert and run these rules against the provided event logs. The mapping file (specified through the `--mapping` parameter) tells Chainsaw which fields in the event logs to use for rule matching.

Example 2: Hunting for Abnormal PowerShell Command Line Size With Sigma (Based on Event ID 4688)

Firstly, let's set the stage by recognizing that PowerShell, being a highly flexible scripting language, is an attractive target for attackers. Its deep integration with Windows APIs and .NET Framework makes it an ideal candidate for a variety of post-exploitation activities.

To conceal their actions, attackers utilize complex encoding layers or misuse cmdlets for purposes they weren't designed for. This leads to abnormally long PowerShell commands that often incorporate Base64 encoding, string merging, and several variables containing fragmented parts of the command.

A Sigma rule that can detect abnormally long PowerShell command lines can be found inside the `C:\Rules\sigma` directory of this section's target, saved as `proc_creation_win_powershell_abnormal_commandline_size.yml`.

Code: yaml

```yaml
title: Unusually Long PowerShell CommandLine
id: d0d28567-4b9a-45e2-8bbc-fb1b66a1f7f6
status: test
description: Detects unusually long PowerShell command lines with a length of 1000 characters or mo
```

```yaml
references:
    - https://speakerdeck.com/heirhabarov/hunting-for-powershell-abuse
author: oscd.community, Natalia Shornikova / HTB Academy, Dimitrios Bougioukas
date: 2020/10/06
modified: 2023/04/14
tags:
    - attack.execution
    - attack.t1059.001
    - detection.threat_hunting
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        EventID: 4688
        NewProcessName|endswith:
            - '\powershell.exe'
            - '\pwsh.exe'
            - '\cmd.exe'
    selection_powershell:
        CommandLine|contains:
            - 'powershell.exe'
            - 'pwsh.exe'
    selection_length:
        CommandLine|re: '.{1000,}'
    condition: selection and selection_powershell and selection_length
falsepositives:
    - Unknown
level: low
```

Sigma Rule Breakdown:

- `logsource`: The rule looks into logs under the category of `process_creation` and is designed to work against Windows machines.

Code: yaml

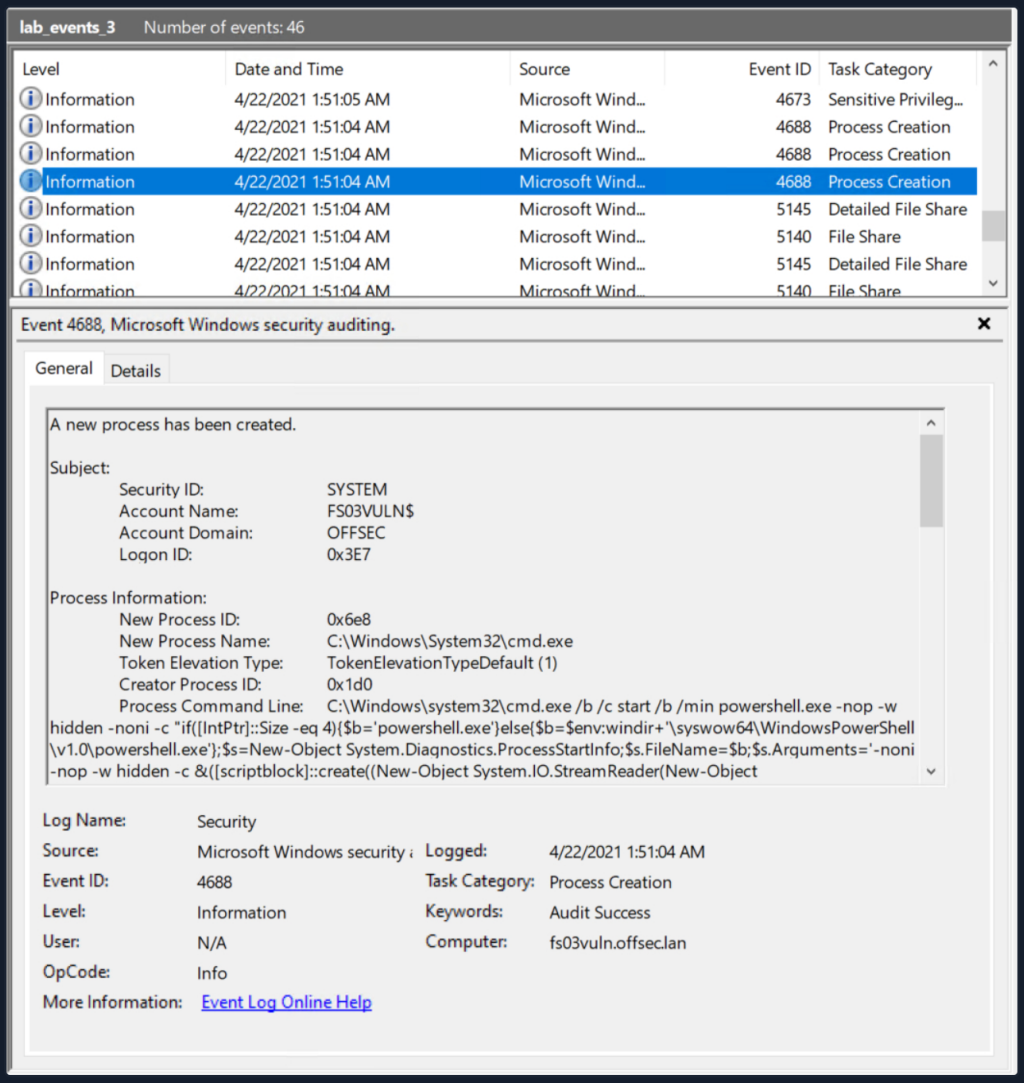
```yaml
logsource:
    category: process_creation
    product: windows
```

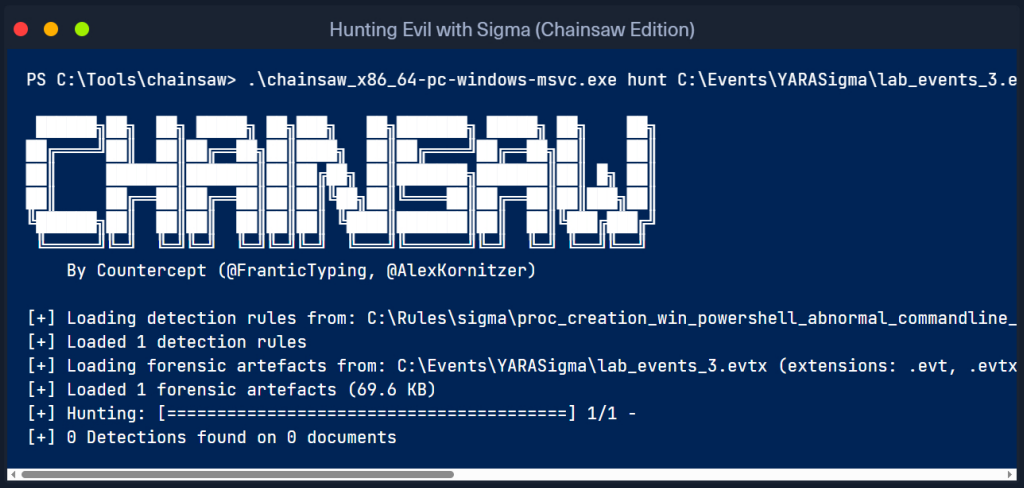
- `detection`: The `selection` section checks if any Windows events with ID `4688` exist and also checks if the `NewProcessName` field ends with `\powershell.exe`, `\pwsh.exe`, or `\cmd.exe`. The `selection_powershell` section checks if the executed command line includes PowerShell-related executables and finally, the `selection_length` section checks if the `CommandLine` field of the `4688` event contains 1,000 characters or more. The `condition` section checks if the selection criteria inside the `selection`, `selection_powershell`, and `selection_length` sections are all met.

Code: yaml

```yaml
detection:
    selection:
        EventID: 4688
        NewProcessName|endswith:
            - '\powershell.exe'
            - '\pwsh.exe'
            - '\cmd.exe'
    selection_powershell:
        CommandLine|contains:
            - 'powershell.exe'
            - 'pwsh.exe'
    selection_length:
        CommandLine|re: '.{1000,}'
    condition: selection and selection_powershell and selection_length
```

Let's put Chainsaw to work by applying the abovementioned Sigma rule, `proc_creation_win_powershell_abnormal_commandline_size.yml` (available at `C:\Rules\sigma`), to `lab_events_3.evtx` (available at `C:\Events\YARASigma\lab_events_3.evtx`, thanks to mdecrevoisier) that contains `4688` events with abnormally long PowerShell commands.





Our Sigma doesn't seem to be able to identify the abnormally long PowerShell commands within these `4688` events.

Does this mean that our Sigma rule is flawed? No! As discussed previously, Chainsaw's mapping file (specified through the `--mapping` parameter) tells it which fields in the event logs to use for rule matching.

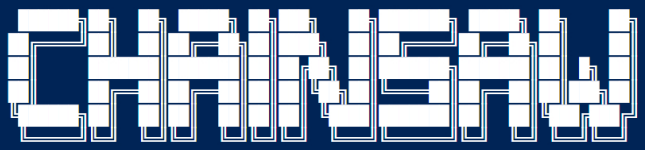
It looks like the `NewProcessName` field was missing from the `sigma-event-logs-all.yml` mapping file.

We introduced the `NewProcessName` field into a `sigma-event-logs-all-new.yml` mapping file inside the `C:\Tools\chainsaw\mappings` directory of this section's target.

Let's run Chainsaw again with this new mapping file.

Hunting Evil with Sigma (Chainsaw Edition)

```
PS C:\Tools\chainsaw> .\chainsaw_x86_64-pc-windows-msvc.exe hunt C:\Events\YARASigma\lab_events_3.e
```



```
    By Countercept (@FranticTyping, @AlexKornitzer)

[+] Loading detection rules from: C:\Rules\sigma\proc_creation_win_powershell_abnormal_commandline_
[+] Loaded 1 detection rules
[+] Loading forensic artefacts from: C:\Events\YARASigma\lab_events_3.evtx (extensions: .evtx, .evt
[+] Loaded 1 forensic artefacts (69.6 KB)
[+] Hunting: [========================================] 1/1 -
[+] Group: Sigma
```

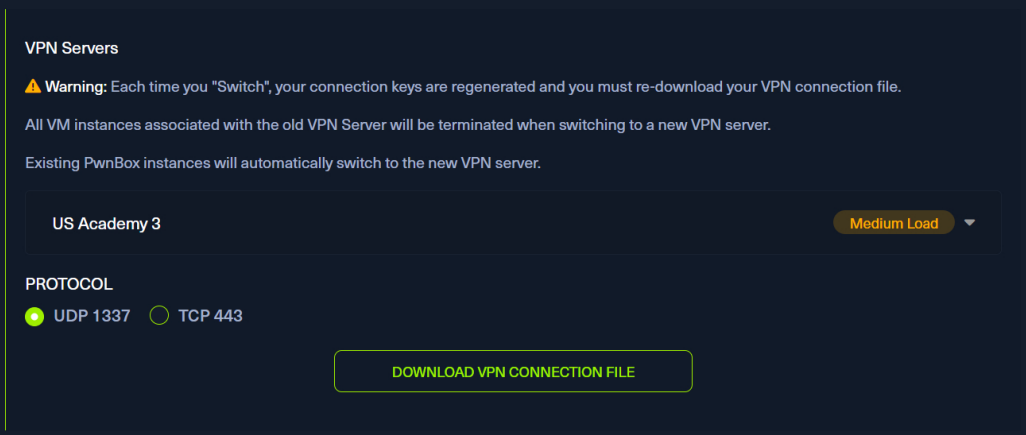
timestamp	detections	count	Event.System.Provider	Even
2021-04-22 08:51:04	+ Unusually Long PowerShell CommandLine	1	Microsoft-Windows-Security-Auditing	4688
2021-04-22 08:51:04	+ Unusually Long PowerShell CommandLine	1	Microsoft-Windows-Security-Auditing	4688

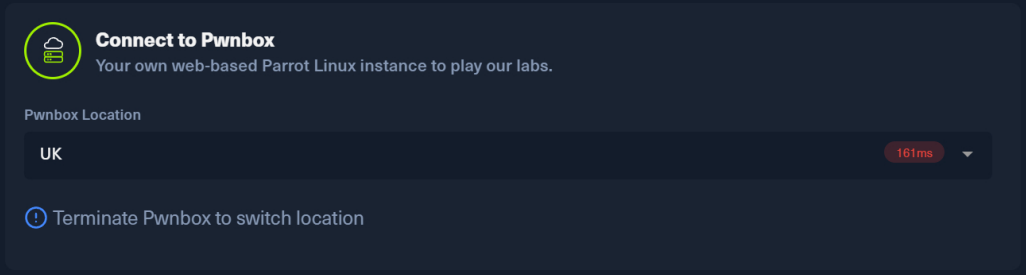
2021-04-22 08:51:05	+ Unusually Long PowerShell CommandLine	1	Microsoft-Windows-Security-Auditing	4688

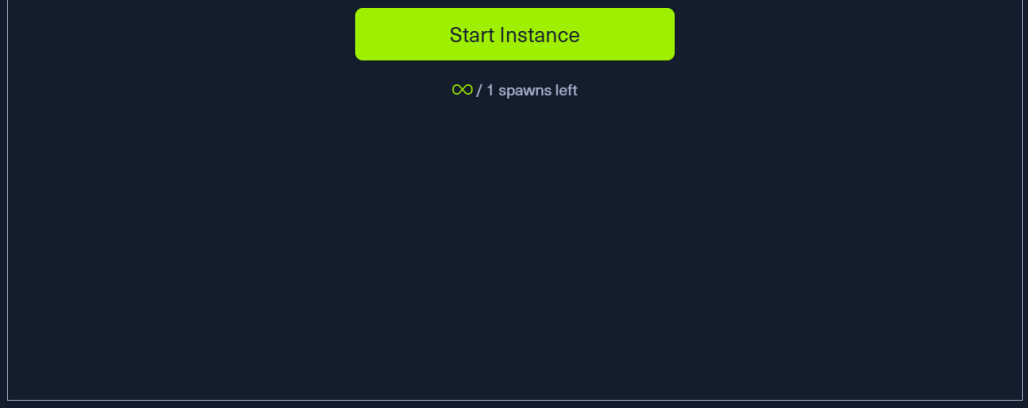
```
[+] 3 Detections found on 3 documents
```

Our Sigma rule successfully uncovered all three abnormally long PowerShell commands that exist inside `lab_events_3.evtx`

Remember that configuration when it comes to using or tranlating Sigma rules is of paramount importance!







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!

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

+ 2 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.

c:\document\virus\

Submit

Previous Next

Mark Complete & Next

Powered by HACKTHEBOX