Practical 1 – Threat Hunting with Open-Source Tools

Objective

The main objective of this practical is to perform threat hunting on a Windows system using open-source tools. Students will simulate suspicious PowerShell activity, collect logs, and create Sigma rules to detect such activity in Elastic Security or Security Onion.

Key learning goals:

- Understand threat hunting workflow
- Collect and analyze Windows event logs
- Write Sigma rules for detection
- Simulate detection of malicious PowerShell execution

Concept

Threat hunting is the proactive process of searching for threats and malicious activity within a network or endpoint before alerts are triggered.

PowerShell is commonly used in attacks because it is a **built-in administrative tool**. Suspicious PowerShell execution is often logged in **Windows Event Logs**, particularly:

- **Event ID 4688:** Process creation
- **Event ID 4104:** PowerShell script block logging

Open-source threat hunting tools used in this lab:

1. **Elastic Security:** Ingests and searches logs to detect suspicious activity.
2. **Security Onion:** Provides a network and host-based monitoring platform for DFIR analysis.

3. **Sigma Rules:** A YAML-based detection standard that can be converted to multiple SIEM query formats.

The lab focuses on detecting suspicious PowerShell activity by manually simulating execution and creating Sigma rules.

VM Setup

- **Windows VM:** Logs PowerShell execution activity in Event Viewer.
- **Kali Linux VM:** Optional, used to simulate attacker activity (e.g., running PowerShell scripts remotely or generating test events).

Step 1 – Simulate Suspicious PowerShell Activity

On the **Windows VM**:

1. Open **PowerShell as Administrator**
2. Run a test command to simulate suspicious execution:

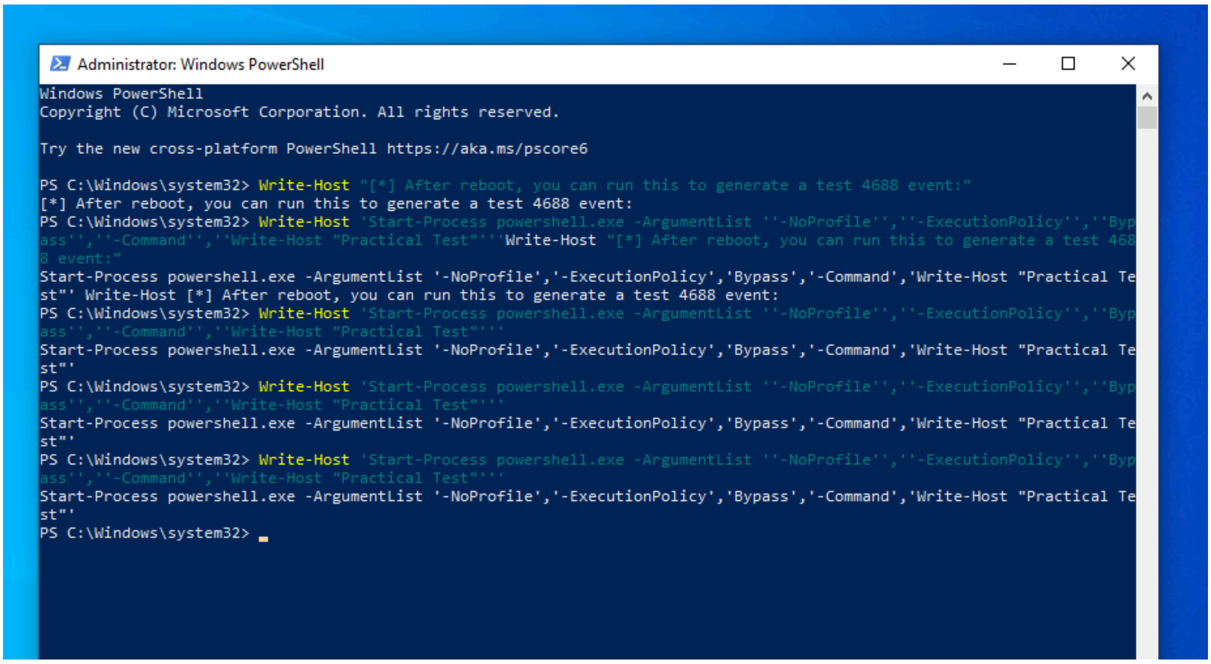
```
powershell.exe -Command "Write-Host 'Suspicious PowerShell Test'"
```

3. Optional: Create a test script file `test_script.ps1` on the Desktop:

```
Write-Host "Suspicious PowerShell Execution"
```

4. Execute the script:

```
powershell.exe -ExecutionPolicy Bypass -File
C:\Users\vboxuser\Desktop\test.ps1
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> Write-Host "[*] After reboot, you can run this to generate a test 4688 event:"
[*] After reboot, you can run this to generate a test 4688 event:
PS C:\Windows\system32> Write-Host 'Start-Process powershell.exe -ArgumentList ''-NoProfile'',''-ExecutionPolicy'',''Byp
ass'',''-Command'',''Write-Host "Practical Test"'''Write-Host "[*] After reboot, you can run this to generate a test 468
8 event:"
Start-Process powershell.exe -ArgumentList '-NoProfile','-ExecutionPolicy','Bypass','-Command','Write-Host "Practical Te
st"' Write-Host [*] After reboot, you can run this to generate a test 4688 event:
PS C:\Windows\system32> Write-Host 'Start-Process powershell.exe -ArgumentList ''-NoProfile'',''-ExecutionPolicy'',''Byp
ass'',''-Command'',''Write-Host "Practical Test"'''
Start-Process powershell.exe -ArgumentList '-NoProfile','-ExecutionPolicy','Bypass','-Command','Write-Host "Practical Te
st"'
PS C:\Windows\system32> Write-Host 'Start-Process powershell.exe -ArgumentList ''-NoProfile'',''-ExecutionPolicy'',''Byp
ass'',''-Command'',''Write-Host "Practical Test"'''
Start-Process powershell.exe -ArgumentList '-NoProfile','-ExecutionPolicy','Bypass','-Command','Write-Host "Practical Te
st"'
PS C:\Windows\system32> Write-Host 'Start-Process powershell.exe -ArgumentList ''-NoProfile'',''-ExecutionPolicy'',''Byp
ass'',''-Command'',''Write-Host "Practical Test"'''
Start-Process powershell.exe -ArgumentList '-NoProfile','-ExecutionPolicy','Bypass','-Command','Write-Host "Practical Te
st"'
PS C:\Windows\system32> _
```

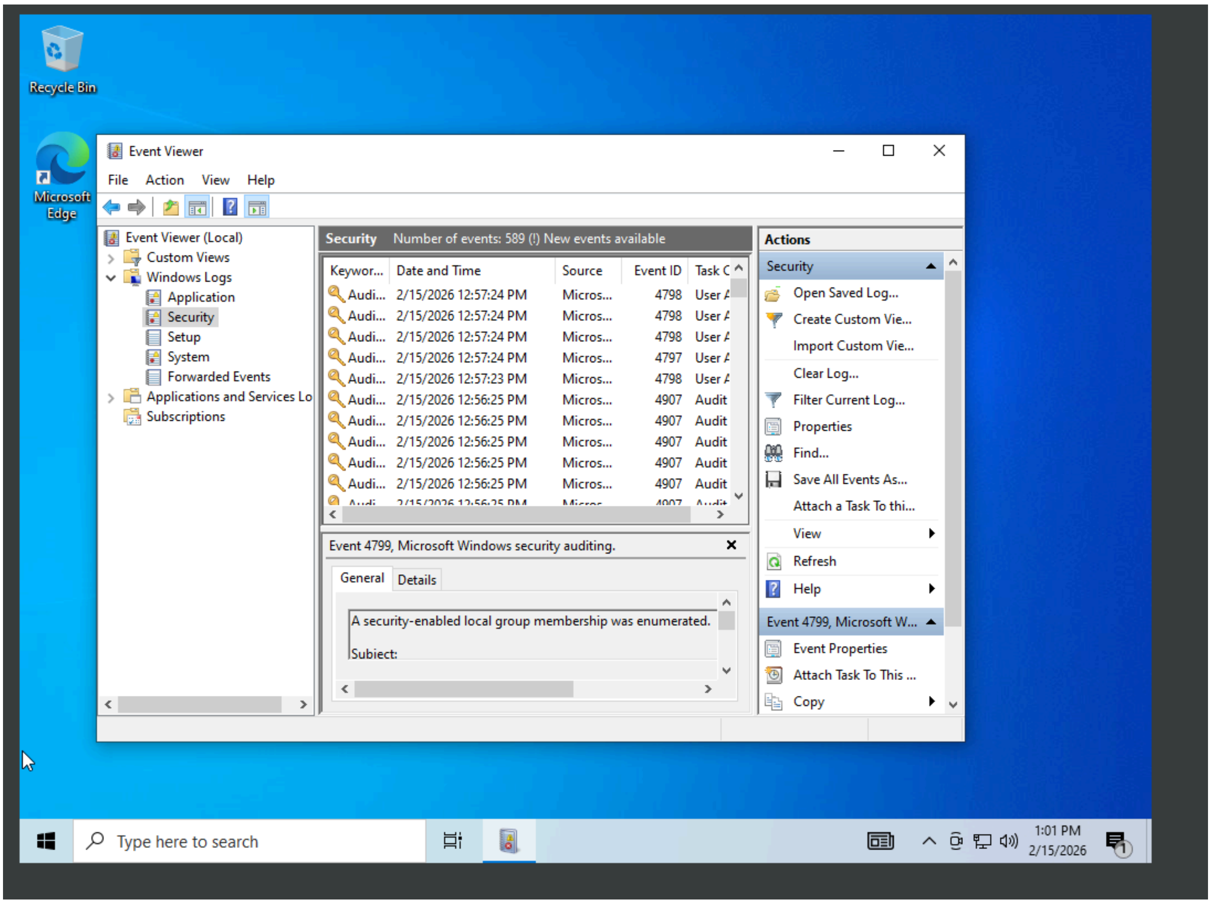
This generates **Event ID 4688** process creation logs in Windows Event Viewer.

Step 2 – Collect Event Logs

1. Open **Event Viewer** → **Windows Logs** → **Security**
2. Filter events for **Event ID 4688** (process creation)
3. Note the fields for Sigma rule creation:
   - **NewProcessName**: Path of executed process (`powershell.exe`)
   - **CommandLine**: Shows the executed command or script
   - **SubjectUserName**: User who executed the process

Key output example:

Field	Value
Event ID	4688
Process Name	`C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`
Command Line	`-ExecutionPolicy Bypass -File C:\Users\vboxuser\Desktop\test_script.ps1`



Security Number of events: 589 (!) New events available

Keywor...	Date and Time	Source	Event ID	Task C
Audi...	2/15/2026 12:57:24 PM	Micros...	4798	User A
Audi...	2/15/2026 12:57:24 PM	Micros...	4798	User A
Audi...	2/15/2026 12:57:24 PM	Micros...	4798	User A
Audi...	2/15/2026 12:57:24 PM	Micros...	4797	User A
Audi...	2/15/2026 12:57:23 PM	Micros...	4798	User A
Audi...	2/15/2026 12:56:25 PM	Micros...	4907	Audit
Audi...	2/15/2026 12:56:25 PM	Micros...	4907	Audit
Audi...	2/15/2026 12:56:25 PM	Micros...	4907	Audit
Audi...	2/15/2026 12:56:25 PM	Micros...	4907	Audit
Audi...	2/15/2026 12:56:25 PM	Micros...	4907	Audit

Event 4799, Microsoft Windows security auditing.

A security-enabled local group membership was enumerated.

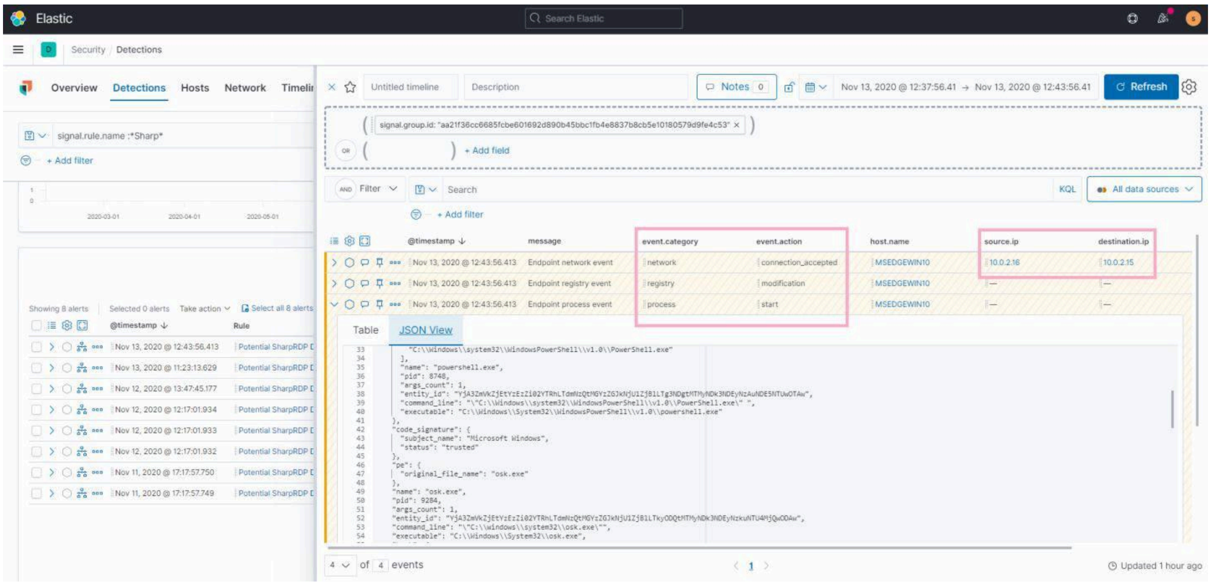
Subject:

Step 3 – Ingest Logs into Elastic Security

1. Install Elastic Security and connect to the Windows VM logs.
2. Ingest the captured **Windows Event Logs** (CSV, Winlogbeat, or direct ingestion).
3. Search for suspicious PowerShell events using **Event ID 4688** and the command line field.

Example query in Elastic Search:

```
event.code: "4688" AND process.name: "powershell.exe"
```



signal.rule.name :*Sharp*

signal.group.id: "aa21f36cc6685fcbe601692d890b45bbc1fb4e8837b8cb5e10180579d9fe4c53"

Showing 8 alerts | Selected 0 alerts | Take action | Select all 8 alerts

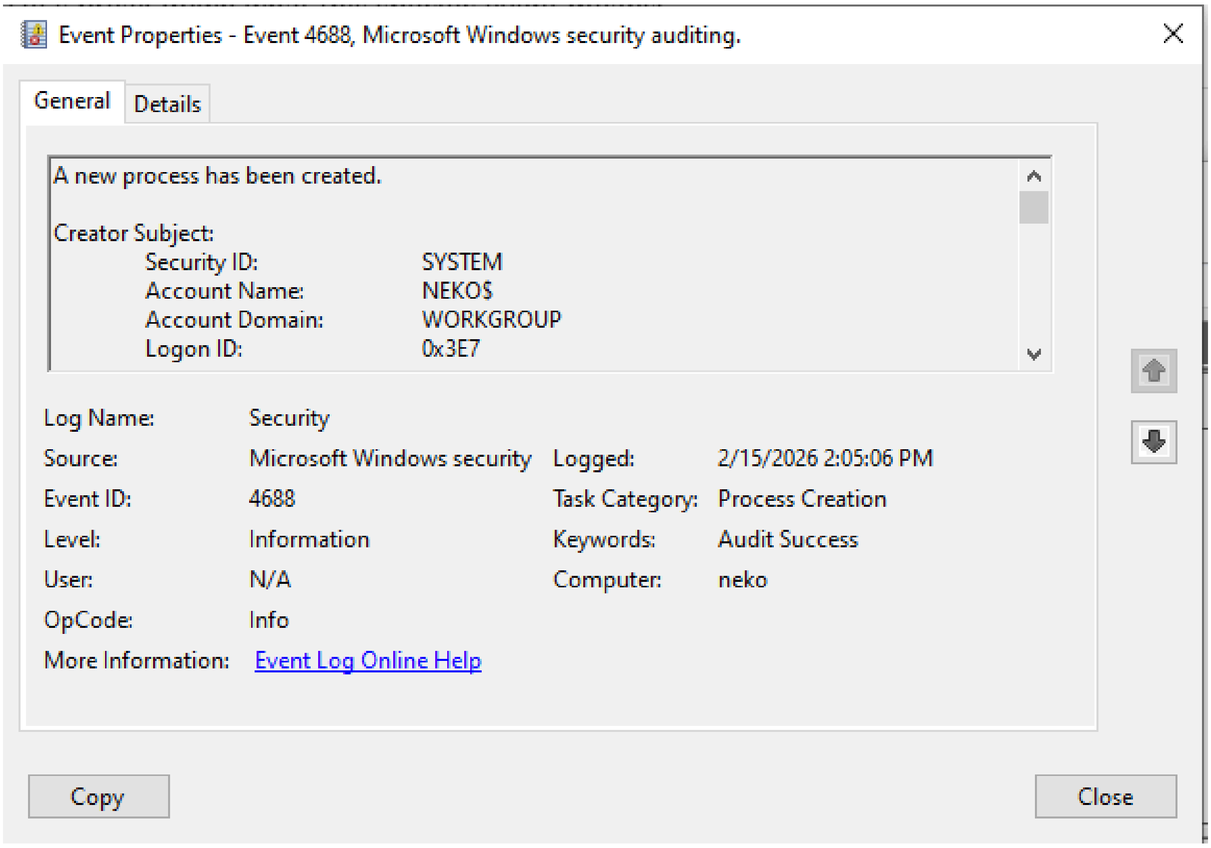
@timestamp	Rule
Nov 13, 2020 @ 12:43:56.413	Potential SharpRDP [
Nov 13, 2020 @ 11:23:13.629	Potential SharpRDP [
Nov 12, 2020 @ 13:47:45.177	Potential SharpRDP [
Nov 12, 2020 @ 12:17:01.934	Potential SharpRDP [
Nov 12, 2020 @ 12:17:01.933	Potential SharpRDP [
Nov 12, 2020 @ 12:17:01.932	Potential SharpRDP [
Nov 11, 2020 @ 17:17:57.750	Potential SharpRDP [
Nov 11, 2020 @ 17:17:57.749	Potential SharpRDP [

@timestamp	message	event.category	event.action	host.name	source.ip	destination.ip
Nov 13, 2020 @ 12:43:56.413	Endpoint network event	network	connection_accepted	MSEDGEWIN10	10.0.2.16	10.0.2.15
Nov 13, 2020 @ 12:43:56.413	Endpoint registry event	registry	modification	MSEDGEWIN10	—	—
Nov 13, 2020 @ 12:43:56.413	Endpoint process event	process	start	MSEDGEWIN10	—	—

```
      "C:\\Windows\\system32\\WindowsPowerShell\\v1.0\\PowerShell.exe"
    ],
    "name": "powershell.exe",
    "pid": 8748,
    "args_count": 1,
    "entity_id": "YjA3ZmVkZjEtYzEzZi02YTRhLTdmNzQtMGYzZGJkNjU1ZjB1LTg3NDgtMTMyNDk3NDEyNzAuNDE5NTUwOTAw",
    "command_line": "\"C:\\Windows\\system32\\WindowsPowerShell\\v1.0\\PowerShell.exe\" ",
    "executable": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
  },
  "code_signature": {
    "subject_name": "Microsoft Windows",
    "status": "trusted"
  },
  "pe": {
    "original_file_name": "osk.exe"
  },
  "name": "osk.exe",
  "pid": 9284,
  "args_count": 1,
  "entity_id": "YjA3ZmVkZjEtYzEzZi02YTRhLTdmNzQtMGYzZGJkNjU1ZjB1LTkyODQtMTMyNDk3NDEyNzkuNTU4MjQwODAw",
  "command_line": "\"C:\\windows\\system32\\osk.exe\"",
  "executable": "C:\\Windows\\System32\\osk.exe",
```

4 of 4 events

Updated 1 hour ago



Event Properties - Event 4688, Microsoft Windows security auditing.

```
A new process has been created.

Creator Subject:
        Security ID:            SYSTEM
        Account Name:           NEKO$
        Account Domain:         WORKGROUP
        Logon ID:               0x3E7
```

```
Log Name:          Security
Source:            Microsoft Windows security   Logged:         2/15/2026 2:05:06 PM
Event ID:          4688                         Task Category:  Process Creation
Level:             Information                  Keywords:       Audit Success
User:              N/A                          Computer:       neko
OpCode:            Info
More Information:  Event Log Online Help
```

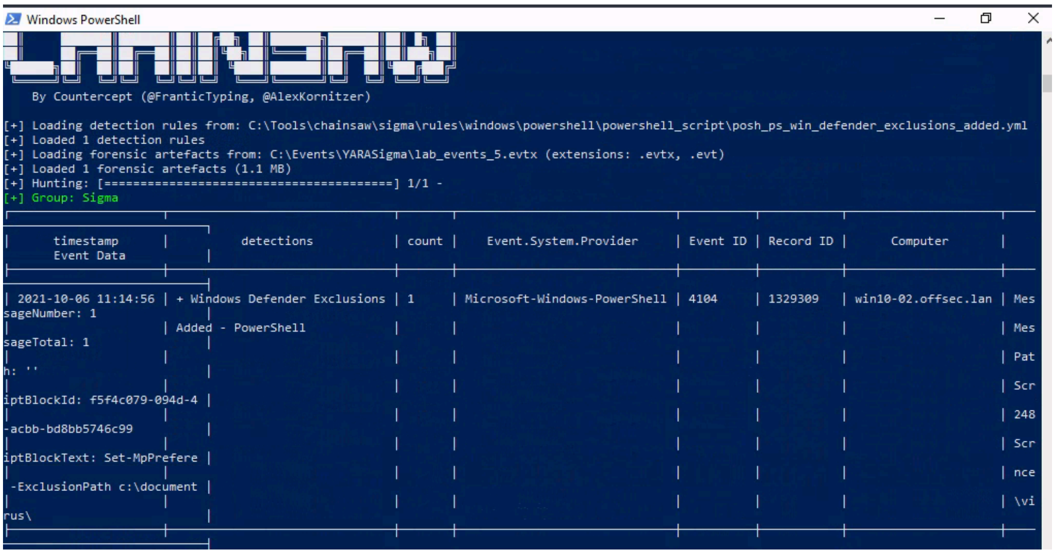
Step 4 – Create Sigma Rule

Sigma rules are SIEM-agnostic YAML detection rules. A sample Sigma rule for suspicious PowerShell execution:

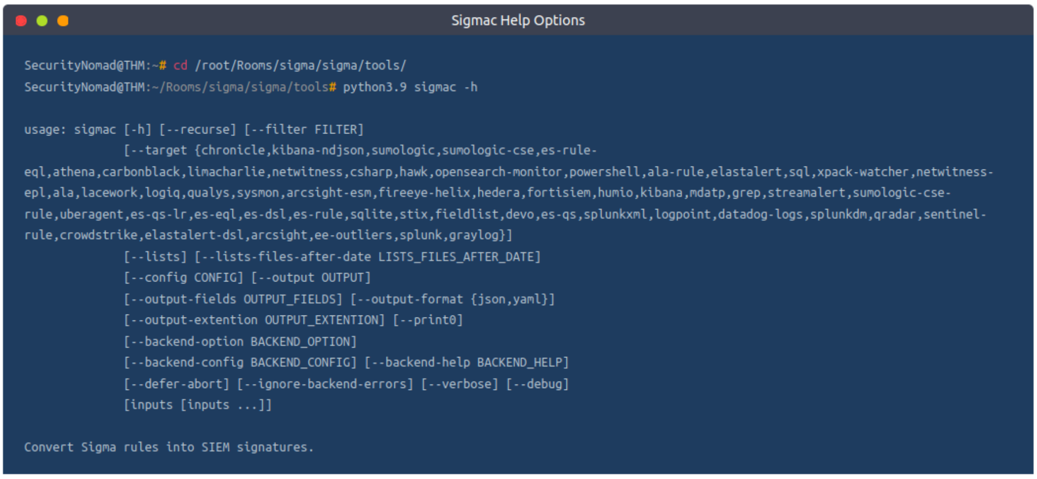
```
title: Suspicious PowerShell Activity

id: 12345678-90ab-cdef-1234-567890abcdef

status: experimental

description: Detects PowerShell execution with bypass policy

author: vboxuser

date: 2026-02-15

logsource:

    product: windows
    category: process_creation
detection:
    selection:
        Image|endswith: '\powershell.exe'
        CommandLine|contains: '-ExecutionPolicy Bypass'
    condition: selection
fields:
    - CommandLine
    - ParentImage
    - User
level: high
```

Explanation:

- `Image|endswith` ensures detection of PowerShell execution.
- `CommandLine|contains` detected bypass usage.
- `level: high` indicates that this is a high-risk event.



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

[+] Loading detection rules from: C:\Tools\chainsaw\sigma\rules\windows\powershell\powershell_script\posh_ps_win_defender_exclusions_added.yml
[+] Loaded 1 detection rules
[+] Loading forensic artefacts from: C:\Events\YARASigma\lab_events_5.evtx (extensions: .evtx, .evt)
[+] Loaded 1 forensic artefacts (1.1 MB)
[+] Hunting: [========================================] 1/1 -
[+] Group: Sigma
├──────────────────┬────────────────────────────┬───────┬───────────────────────────────┬─────────┬─────────────┬────────────────────┬
│     timestamp    │         detections         │ count │     Event.System.Provider     │ Event ID │ Record ID │      Computer       │
      Event Data   │
├──────────────────┼────────────────────────────┼───────┼───────────────────────────────┼─────────┼─────────────┼────────────────────┼
│ 2021-10-06 11:14:56 │ + Windows Defender Exclusions │ 1     │ Microsoft-Windows-PowerShell │ 4104    │ 1329309     │ win10-02.offsec.lan │ Mes
sageNumber: 1      │
│                  │ Added - PowerShell         │       │                               │         │             │                    │ Mes
sageTotal: 1       │
│                  │                            │       │                               │         │             │                    │ Pat
h: ''              │
│                  │                            │       │                               │         │             │                    │ Scr
iptBlockId: f5f4c079-094d-4 │
│                  │                            │       │                               │         │             │                    │ 248
-acbb-bd8bb5746c99 │
│                  │                            │       │                               │         │             │                    │ Scr
iptBlockText: Set-MpPrefere │
│                  │                            │       │                               │         │             │                    │ nce
 -ExclusionPath c:\document │
│                  │                            │       │                               │         │             │                    │ \vi
rus\               │
├──────────────────┼────────────────────────────┼───────┼───────────────────────────────┼─────────┼─────────────┼────────────────────┼
```

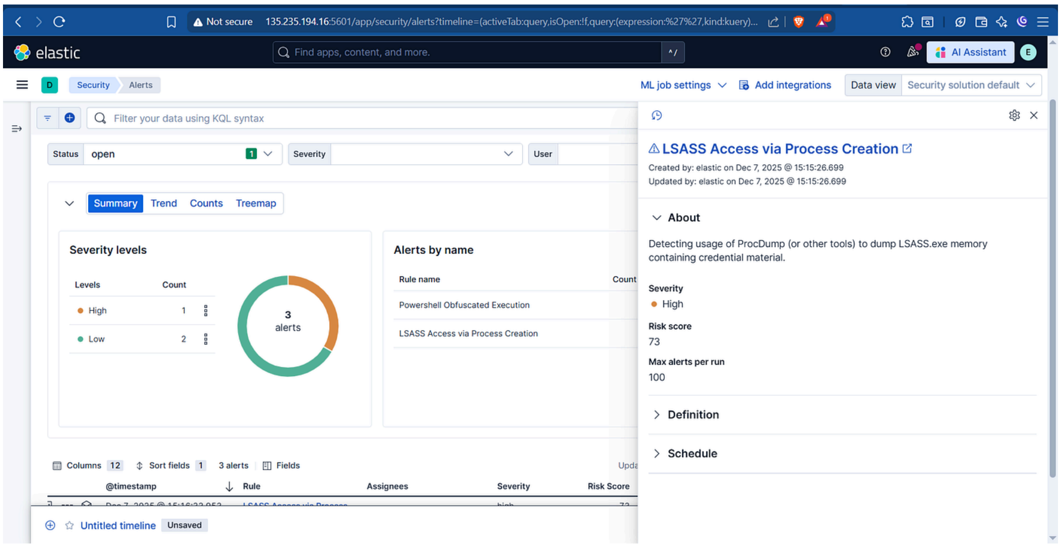


```
SecurityNomad@THM:~# cd /root/Rooms/sigma/sigma/tools/
SecurityNomad@THM:~/Rooms/sigma/sigma/tools# python3.9 sigmac -h

usage: sigmac [-h] [--recurse] [--filter FILTER]
              [--target {chronicle,kibana-ndjson,sumologic,sumologic-cse,es-rule-
eql,athena,carbonblack,limacharlie,netwitness,csharp,hawk,opensearch-monitor,powershell,ala-rule,elastalert,sql,xpack-watcher,netwitness-
epl,ala,lacework,logiq,qualys,sysmon,arcsight-esm,fireeye-helix,hedera,fortisiem,humio,kibana,mdatp,grep,streamalert,sumologic-cse-
rule,uberagent,es-qs-lr,es-eql,es-dsl,es-rule,sqlite,stix,fieldlist,devo,es-qs,splunkxml,logpoint,datadog-logs,splunkdm,qradar,sentinel-
rule,crowdstrike,elastalert-dsl,arcsight,ee-outliers,splunk,graylog}]
              [--lists] [--lists-files-after-date LISTS_FILES_AFTER_DATE]
              [--config CONFIG] [--output OUTPUT]
              [--output-fields OUTPUT_FIELDS] [--output-format {json,yaml}]
              [--output-extention OUTPUT_EXTENTION] [--print0]
              [--backend-option BACKEND_OPTION]
              [--backend-config BACKEND_CONFIG] [--backend-help BACKEND_HELP]
              [--defer-abort] [--ignore-backend-errors] [--verbose] [--debug]
              [inputs [inputs ...]]

Convert Sigma rules into SIEM signatures.
```

Step 5 – Validate Detection

1. Apply the Sigma rule in Elastic Security (or convert to SIEM query).
2. Run PowerShell execution again with `-ExecutionPolicy Bypass`.
3. Observe the alert in Elastic Security or Security Onion.

This validates that the Sigma rule successfully detects suspicious PowerShell execution.


Not secure
135.235.194.16:5601/app/security/alerts?timeline=(activeTab:query,isOpen:!f,query:(expression:%27%27,kind:kuery)...
elastic
Find apps, content, and more.
AI Assistant
Security
Alerts
ML job settings
Add integrations
Data view
Security solution default
Filter your data using KQL syntax
Status
open
Severity
User
Summary
Trend
Counts
Treemap
3
alerts
Columns 12
Sort fields 1
3 alerts
Fields
Untitled timeline
Unsaved

Severity levels

Levels	Count
● High	1
● Low	2

Alerts by name

Rule name	Count
Powershell Obfuscated Execution	
LSASS Access via Process Creation	

@timestamp	Rule	Assignees	Severity	Risk Score

LSASS Access via Process Creation

Created by: elastic on Dec 7, 2025 @ 15:15:26.699
Updated by: elastic on Dec 7, 2025 @ 15:15:26.699

About

Detecting usage of ProcDump (or other tools) to dump LSASS.exe memory containing credential material.

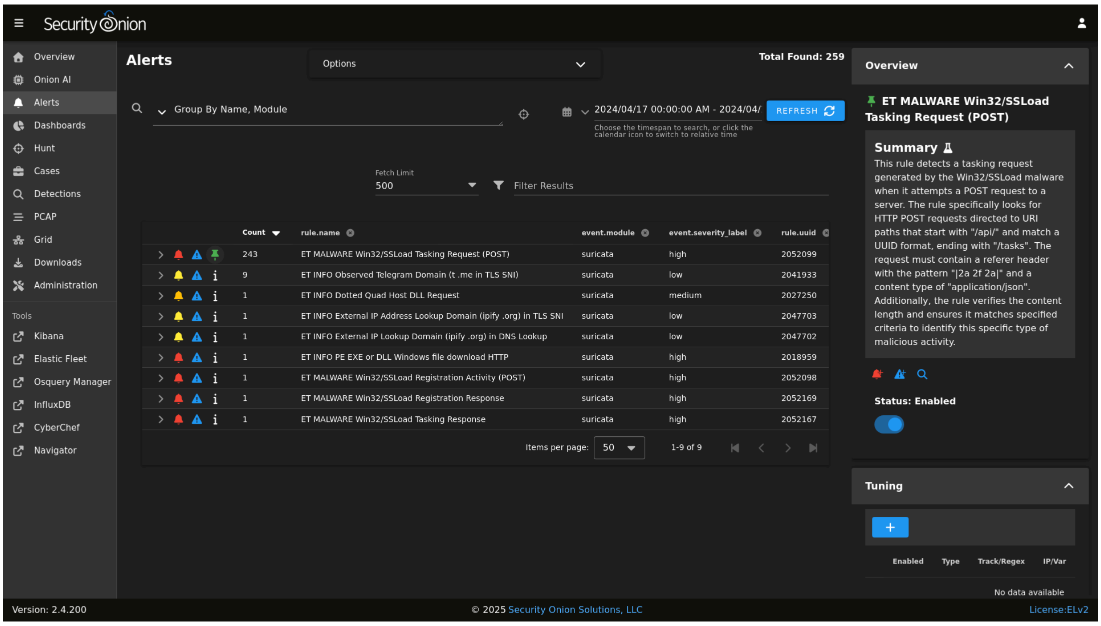
Severity
● High

Risk score
73

Max alerts per run
100

Definition

Schedule



Alerts

Total Found: 259

Group By Name, Module

2024/04/17 00:00:00 AM - 2024/04/

Choose the timespan to search, or click the calendar icon to switch to relative time

Fetch Limit
500

Count	rule.name	event.module	event.severity_label	rule.uuid
243	ET MALWARE Win32/SSLoad Tasking Request (POST)	suricata	high	2052099
9	ET INFO Observed Telegram Domain (t .me in TLS SNI)	suricata	low	2041933
1	ET INFO Dotted Quad Host DLL Request	suricata	medium	2027250
1	ET INFO External IP Address Lookup Domain (ipify .org) in TLS SNI	suricata	low	2047703
1	ET INFO External IP Lookup Domain (ipify .org) in DNS Lookup	suricata	low	2047702
1	ET INFO PE EXE or DLL Windows file download HTTP	suricata	high	2018959
1	ET MALWARE Win32/SSLoad Registration Activity (POST)	suricata	high	2052098
1	ET MALWARE Win32/SSLoad Registration Response	suricata	high	2052169
1	ET MALWARE Win32/SSLoad Tasking Response	suricata	high	2052167

Items per page: 50   1-9 of 9

Overview

ET MALWARE Win32/SSLoad Tasking Request (POST)

Summary

This rule detects a tasking request generated by the Win32/SSLoad malware when it attempts a POST request to a server. The rule specifically looks for HTTP POST requests directed to URI paths that start with "/api/" and match a UUID format, ending with "/tasks". The request must contain a referer header with the pattern "|2a 2f 2a|" and a content type of "application/json". Additionally, the rule verifies the content length and ensures it matches specified criteria to identify this specific type of malicious activity.

Status: Enabled

No data available

Version: 2.4.200

© 2025 Security Onion Solutions, LLC

License:ELv2

Attack Path Summary

In this practical, suspicious PowerShell activity was simulated on a Windows VM. A test script was executed using the `-ExecutionPolicy Bypass` flag to mimic a typical attack vector used by malware or phishing campaigns. Event ID 4688 captured process creation details, including process name, command line, and executing user. Logs were ingested into Elastic Security to enable threat hunting and query-based analysis. A Sigma rule was created to detect the execution of `powershell.exe` with bypass flags, and its detection capabilities were validated. This demonstrates how threat hunters can proactively identify malicious activity by correlating process artifacts, command-line parameters, and user behavior across multiple endpoints, bridging manual inspection and automated detection.

Conclusion

- Threat hunting was successfully conducted using **Elastic Security, Security Onion, and Sigma rules**.
- Suspicious PowerShell execution was detected and logged in Event Viewer (Event ID 4688).
- Sigma rules provided a standardized, SIEM-agnostic method to detect risky PowerShell commands.
- The lab highlights how **process and command-line analysis** can support proactive threat detection.
- Students gained hands-on experience simulating, detecting, and validating a typical attack vector in a controlled environment.