Threat Hunter Playbook

Search this book...

KNOWLEDGE LIBRARY

Windows

PRE-HUNT ACTIVITIES

Data Management

GUIDED HUNTS

Windows

LSASS Memory Read Access

DLL Process Injection via CreateRemoteThread and LoadLibrary

Active Directory Object Access via Replication Services

Active Directory Root Domain Modification for Replication Services

Registry Modification to Enable Remote Desktop Conections

Local PowerShell Execution

WDigest Downgrade

PowerShell Remote Session

Alternate PowerShell Hosts

Domain DPAPI Backup Key Extraction

SysKey Registry Keys Access

SAM Registry Hive Handle Request

WMI Win32_Process Class and Create Method for Remote Execution

WMI Eventing

WMI Module Load

Local Service Installation

Remote Service creation

Remote Service Control Manager Handle

Remote Interactive Task Manager LSASS Dump

Registry Modification for Extended NetNTLM Downgrade

Access to Microphone Device

Remote WMI ActiveScriptEventConsumers

Remote DCOM IErtUtil DLL Hijack

Remote WMI Wbemcomn DLL Hijack

SMB Create Remote File

WuauclI CreateRemoteThread Execution

TUTORIALS

Jupyter Notebooks

Local PowerShell Execution

Hypothesis

Adversaries might be leveraging PowerShell to execute code within my environment

Technical Context

Offensive Tradecraft

Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Therefore, it is important to understand the basic artifacts left when PowerShell is used in your environment.

Pre-Recorded Security Datasets

Metadata	Value
docs	https://securitydatasets.com/notebooks/atomic/windows/execution/SDWIN-190518182022.html
link	https://raw.githubusercontent.com/OTRF/Security-Datasets/master/datasets/atomic/windows/execution/host/empire_launcher_vbs.zip

Download Dataset

```
import requests
from zipfile import ZipFile
from io import BytesIO

url = 'https://raw.githubusercontent.com/OTRF/Security-Datasets/master/dataset
zipFileRequest = requests.get(url)
zipFile = ZipFile(BytesIO(zipFileRequest.content))
datasetJSONPath = zipFile.extract(zipFile.namelist()[0])
```

Read Dataset

```
import pandas as pd
from pandas.io import json

df = json.read_json(path_or_buf=datasetJSONPath, lines=True)
```

Analytics

A few initial ideas to explore your data and validate your detection logic:

Analytic I

Within the classic PowerShell log, event ID 400 indicates when a new PowerShell host process has started. You can filter on powershell.exe as a host application if you want to or leave it without a filter to capture every single PowerShell host.

Data source	Event Provider	Relationship	Event
Powershell	Windows PowerShell	Application host started	400
Powershell	Microsoft-Windows-PowerShell/Operational	User started Application host	4103

Logic

```
SELECT `@timestamp`, Hostname
FROM dataTable
WHERE (Channel = "Microsoft-Windows-PowerShell/Operational" OR Channel = "Winc
    AND (EventID = 400 OR EventID = 4103)
```

Pandas Query

```
(
df[['@timestamp','Hostname']]

[((df['Channel'] == 'Windows PowerShell') | (df['Channel'] == 'Microsoft-Windo
    & (
      (df['EventID'] == 400)
      | (df['EventID'] == 4103)
    )
]
.head()
)
```

Analytic II

Look for non-interactive powershell session might be a sign of PowerShell being executed by another application in the background.

Data source	Event Provider	Relationship	Event
Process	Microsoft-Windows-Security-Auditing	Process created Process	4688

Logic

```
SELECT `@timestamp`, Hostname, NewProcessName, ParentProcessName
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 4688
    AND NewProcessName LIKE "%powershell.exe"
    AND NOT ParentProcessName LIKE "%explorer.exe"
```

Pandas Query

```
(
df[['@timestamp','Hostname','NewProcessName','ParentProcessName']]

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4688)
    & (df['NewProcessName'].str.lower().str.endswith('powershell.exe', na=Fals
    & (~df['ParentProcessName'].str.lower().str.endswith('explorer.exe', na=Fa
]
.head()
)
```

Analytic III

Look for non-interactive powershell session might be a sign of PowerShell being executed by another application in the background.

Data source	Event Provider	Relationship	Event
Process	Microsoft-Windows-Sysmon/Operational	Process created Process	1

Logic

```
SELECT `@timestamp`, Hostname, Image, ParentImage
FROM dataTable
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
    AND EventID = 1
    AND Image LIKE "%powershell.exe"
    AND NOT ParentImage LIKE "%explorer.exe"
```

Pandas Query

```
(
df[['@timestamp','Hostname','Image','ParentImage']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 1)
    & (df['Image'].str.lower().str.endswith('powershell.exe', na=False))
    & (~df['ParentImage'].str.lower().str.endswith('explorer.exe', na=False))
]
.head()
)
```

Analytic IV

Monitor for processes loading PowerShell DLL `*system.management.automation*`.

Data source	Event Provider	Relationship	Event
Module	Microsoft-Windows-Sysmon/Operational	Process loaded Dll	7

Logic

```
SELECT `@timestamp`, Hostname, Image, ImageLoaded
FROM dataTable
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
    AND EventID = 7
    AND (lower(Description) = "system.management.automation" OR lower(ImageLoa
```

Pandas Query

```
(
df[['@timestamp','Hostname','Image','ImageLoaded']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 7)
    & (
        (df['Description'].str.lower() == 'system.management.automation')
        | (df['ImageLoaded'].str.lower().str.contains('.*system.management.au
    )
]
.head()
)
```

Analytic V

Monitoring for PSHost* pipes is another interesting way to find PowerShell execution.

Data source	Event Provider	Relationship	Event
Named Pipe	Microsoft-Windows-Sysmon/Operational	Process created Pipe	17

Logic

```
SELECT `@timestamp`, Hostname, Image, PipeName
FROM dataTable
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
    AND EventID = 17
    AND lower(PipeName) LIKE "\\\\pshost%"
```

Pandas Query

```
(
df[['@timestamp','Hostname','Image','PipeName']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 17)
    & (df['PipeName'].str.lower().str.startswith('\pshost', na=False))
]
.head()
)
```

Analytic VI

The `PowerShell Named Pipe IPC` event will indicate the name of the PowerShell AppDomain that started. Sign of PowerShell execution.

Data source	Event Provider	Relationship	Event
Powershell	Microsoft-Windows-PowerShell/Operational	Application domain started	53504

Logic

```
SELECT `@timestamp`, Hostname, Message
FROM dataTable
WHERE Channel = "Microsoft-Windows-PowerShell/Operational"
    AND EventID = 53504
```

Pandas Query

```
(
df[['@timestamp','Hostname','Message']]

[(df['Channel'] == 'Microsoft-Windows-PowerShell/Operational')
    & (df['EventID'] == 53504)
]
.head()
)
```

Known Bypasses

False Positives

Hunter Notes

- Explore the data produced in your environment with the analytics above and document what normal looks like from a PowerShell perspective.
- If execution of PowerShell happens all the time in your environment, I suggest to categorize the data you collect by business unit to build profiles and be able to filter out potential noise.
- You can also stack the values of the command line arguments being used. You can hash the command line arguments too and stack the values.

Hunt Output

Type	Link
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/pipe_created/sysmon_powershell_execution_pipe.yml
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/win_non_interactive_powershell.yml

References

- https://github.com/darkoperator/Presentations/blob/master/PSConfEU 2019 Tracking PowerShell Usage.pdf
- https://posts.specterops.io/abusing-powershell-desired-state-configuration-for-lateral-movement-ca42ddbe6f06

Previous
Registry Modification to Enable Remote Desktop Conections

Next
WDigest Downgrade

By Roberto Rodriguez @Cyb3rWard0g