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

PowerShell Remote Session

Hypothesis

Adversaries might be leveraging remote powershell sessions to execute code on remote systems throughout my environment

Technical Context

Offensive Tradecraft

Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. In addition, it can be used to execute code remotely via Windows Remote Management (WinRM) services. Therefore, it is important to understand the basic artifacts left when PowerShell is used to execute code remotely via a remote powershell session.

Pre-Recorded Security Datasets

Metadata	Value
docs	https://securitydatasets.com/notebooks/atomic/windows/execution/SDWIN-190518211456.html
link	https://raw.githubusercontent.com/OTRF/Security-Datasets/master/datasets/atomic/windows/lateral_movement/host/empire_psremoting_stager.zip

Download Dataset

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

url = 'https://raw.githubusercontent.com/OTRF/Security-Datasets/master/datasets
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

Process wsmprovhost hosts the active remote session on the target. Therefore, it is important to monitor for any the initialization of the PowerShell host wsmprovhost.

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, Channel
FROM dataTable
WHERE (Channel = "Microsoft-Windows-PowerShell/Operational" OR Channel = "Windo
    AND (EventID = 400 OR EventID = 4103)
    AND Message LIKE "%HostApplication%wsmprovhost%"
```

Pandas Query

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

[(df['Channel'].isin(['Microsoft-Windows-PowerShell/Operational','Windows Power
    & (df['EventID'].isin([400,4103]))
    & (df['Message'].str.contains('.*HostApplication.*wsmprovhost.*', regex=Tru
]
.head()
)
```

Analytic II

Monitor for any incoming network connection where the destination port is either 5985 or 5986. That will be hosted most likely by the System process.

Data source	Event Provider	Relationship	Event
Process	Microsoft-Windows-Security-Auditing	Process connected to Port	5156

Logic

```
SELECT `@timestamp`, Hostname, Application, SourceAddress, DestAddress, LayerNa
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 5156
    AND (DestPort = 5985 OR DestPort = 5986)
    AND LayerRTID = 44
```

Pandas Query

```
(
df[['@timestamp','Hostname','Application','SourceAddress','DestAddress','LayerN
[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 5156)
    & (
        (df['DestPort'] == 5985)
        | (df['DestPort'] == 5986)
    )
    & (df['LayerRTID'] == 44)
]
.head()
)
```

Analytic III

Process wsmprovhost hosts the active remote session on the target. Therefore, from a process creation perspective, it is to document any instances of wsmprovhost being spawned and spawning other processes.

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

Logic

```
SELECT `@timestamp`, Hostname, ParentProcessName, NewProcessName
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 4688
    AND (ParentProcessName LIKE "%wsmprovhost.exe" OR NewProcessName LIKE "%wsm
```

Pandas Query

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

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4688)
    & (
        (df['ParentProcessName'].str.lower().str.endswith('wsmprovhost.exe', na
        | (df['NewProcessName'].str.lower().str.endswith('wsmprovhost.exe', na=
    )
]
.head()
)
```

Analytic IV

Process wsmprovhost hosts the active remote session on the target. Therefore, from a process creation perspective, it is to document any instances of wsmprovhost being spawned and spawning other processes.

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

Logic

```
SELECT `@timestamp`, Hostname, ParentImage, Image
FROM dataTable
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
    AND EventID = 1
    AND (ParentImage LIKE "%wsmprovhost.exe" OR Image LIKE "%wsmprovhost.exe")
```

Pandas Query

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

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

Analytic V

Monitor for outbound network connection where the destination port is either 5985 or 5986 and the use is not NT AUTHORITY\NETWORK SERVICE or NT AUTHORITY\SYSTEM.

Data source	Event Provider	Relationship	Event
Process	Microsoft-Windows-Sysmon/Operational	User connected to Port	3

Logic

```
SELECT `@timestamp`, Hostname, User, Initiated, Image, SourceIp, DestinationIp
FROM dataTable
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
    AND EventID = 3
    AND (DestinationPort = 5985 OR DestinationPort = 5986)
    AND NOT User = "NT AUTHORITY\\\\NETWORK SERVICE"
```

Pandas Query

```
(
df[['@timestamp','Hostname','User','Initiated','Image','SourceIp','DestinationI

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 3)
    & (
        (df['DestinationPort'] == 5985)
        | (df['DestinationPort'] == 5986)
    )
    & (~df['User'].isin(['NT AUTHORITY\\NETWORK SERVICE', 'NT AUTHORITY\\SYSTEM
]
)
```

Known Bypasses

False Positives

Hunter Notes

- Explore the data produced in your lab environment with the analytics above and document what normal looks like from a PowerShell perspective. Then, take your findings and explore your production environment.
- If powershell activity locally or remotely via winrm happens all the time in your environment, I suggest to categorize the data you collect by business unit or department to document profiles.
- Layer 44 translatest to layer filter FWPM_LAYER_ALE_AUTH_RECV_ACCEPT_V4 / FWPM_LAYER_ALE_AUTH_RECV_ACCEPT_V6. This filtering layer allows for authorizing accept requests for incoming TCP connections, as well as authorizing incoming non-TCP traffic based on the first packet received. Looking for destination ports related to remote PowerShell Sessions and Layer 44 is very helpful.

Hunt Output

Type	Link
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/powershell/powershell_module/posh_pm_remote_powers
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/powershell/powershell_classic/posh_pc_remote_powershe
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/network_connection/sysmon_remote_powershell_session
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/win_remote_powershell_session_proces
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_remote_powershell_session.yml

References

- https://docs.microsoft.com/en-us/powershell/scripting/learn/remoting/running-remote-commands?view=powershell-6#windows-powershell-remoting
- https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_remote_requirements?view=powershell-6
- https://docs.microsoft.com/en-us/windows/win32/fwp/management-filtering-layer-identifiers-

‹ Previous
WDigest Downgrade

Next ›
Alternate PowerShell Hosts

By Roberto Rodriguez @Cyb3rWard0g