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

Powered by Jupyter Book

Alternate PowerShell Hosts

Contents

Hypothesis
Technical Context
Offensive Tradecraft
Pre-Recorded Security Datasets
Analytics
Known Bypasses
False Positives
Hunter Notes
Hunt Output
References

Hypothesis

Adversaries might be leveraging alternate PowerShell Hosts to execute PowerShell evading traditional PowerShell detections that look for powershell.exe in my environment.

Technical Context

Offensive Tradecraft

Adversaries can abuse alternate signed PowerShell Hosts to evade application whitelisting solutions that block powershell.exe and naive logging based upon traditional PowerShell hosts. Characteristics of a PowerShell host (Matt Graeber @mattifestation) >

- These binaries are almost always C#/.NET .exes/.dlls
- These binaries have System.Management.Automation.dll as a referenced assembly
- These may not always be “built in” binaries

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/atomic/windows/lateral_m
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. Excluding PowerShell.exe is a good way to find alternate PowerShell hosts.

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 = "Windows PowerShell")
    AND (EventID = 400 OR EventID = 4103)
    AND NOT Message LIKE "%Host Application%powershell%"
```

Pandas Query

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

[((df['Channel'] == 'Microsoft-Windows-PowerShell/Operational')|(Windows PowerShell))
    & ((df['EventID'] == 400)|(df['EventID'] == 4103))
    & (~df['Message'].str.contains('.*Host Application%powershell.*', regex=True))
]
.head()
)
```

Analytic II

Looking for processes loading a specific PowerShell DLL is a very effective way to document the use of PowerShell in your environment.

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

Logic

```
SELECT `@timestamp`, Hostname, Image, Description
FROM dataTable
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
    AND EventID = 7
    AND (lower(Description) = "system.management.automation" OR lower(ImageLoaded) LIKE "%system.managem
    AND NOT Image LIKE "%powershell.exe"
```

Pandas Query

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

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 7)
    & (
      (df['Description'].str.lower() == 'system.management.automation')
      | (df['ImageLoaded'].str.contains('.*system.management.automation.*', regex=True))
    )
    & (~df['Image'].str.lower().str.endswith('powershell.exe', na=False))
]
)
```

Analytic III

Monitoring for `PSHost*` pipes is another interesting way to find other alternate PowerShell hosts in your environment.

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%"
    AND NOT Image LIKE "%powershell.exe"
```

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))
    & (~df['Image'].str.lower().str.endswith('powershell.exe', na=False))
]
.head()
)
```

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 alternate powershell hosts. Then, take your findings and explore your production environment.
- You can also run the script below named PowerShellHostFinder.ps1 by Matt Graber and audit PS host binaries in your environment.

Hunt Output

Type	Link
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/powershell/powershell_module/posh_pm_alternate_powershell_hosts.yml
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/powershell/powershell_classic/posh_pc_alternate_powershell_hosts.yml
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/image_load/sysmon_alternate_powershell_hosts_moduleload.yml

Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/pipe_created/sysmon_alternate_powershell_hosts_pipe.yml

References

- https://twitter.com/mattifestation/status/971840487882506240
- https://gist.githubusercontent.com/mattifestation/fcae777470f1bdeb9e4b32f93c245fd3/raw/abbe79c660829ab9aad58581baf681655f6ba305/PowerShellHostFinder.ps1

Previous
PowerShell Remote Session

Next
Domain DPAPI Backup Key Extraction

By Roberto Rodriguez @Cyb3rWard0g