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
- Wuauclt CreateRemoteThread

Contents

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

WMI Module Load

Hypothesis

Adversaries might be leveraging WMI modules to execute WMI tasks bypassing controls monitoring for wmiprvse.exe or wmiapsrv.exe activity

Technical Context

WMI is the Microsoft implementation of the Web-Based Enterprise Management (WBEM) and Common Information Model (CIM). Both standards aim to provide an industry-agnostic means of collecting and transmitting information related to any managed component in an enterprise. An example of a managed component in WMI would be a running process, registry key, installed service, file information, etc. At a high level, Microsoft's implementation of these standards can be summarized as follows > Managed Components Managed components are represented as WMI objects â€" class instances representing highly structured operating system data. Microsoft provides a wealth of WMI objects that communicate information related to the operating system. E.g. Win32_Process, Win32_Service, AntiVirusProduct, Win32_StartupCommand, etc. WMI modules loaded by legit processes such as wmiprvse.exe or wmiapsrv.exe are the following

C:\Windows\System32\wmiclnt.dll C:\Windows\System32\wbem\WmiApRpl.dll C:\Windows\System32\wbem\wmiprov.dll C:\Windows\System32\wbem\wmiutils.dll

Offensive Tradecraft

Adversaries could leverage the WMI modules above to execute WMI tasks bypassing controls looking for wmiprvse.exe or wmiapsrv.exe activity.

Pre-Recorded Security Datasets

Metadata	Value
docs	https://securitydatasets.com/notebooks/atomic/windows/defense_evasion/SDWIN-190518200432.html
link	https://raw.githubusercontent.com/OTRF/Security-Datasets/master/datasets/atomic/windows/defense_evasion/host/empire_psinject_PEinjection.zip

Download Dataset

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

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

Look for processes (non wmiprvse.exe or WmiApSrv.exe) loading wmi modules.

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(ImageLoaded) LIKE "%wmiclnt.dll"
        OR lower(ImageLoaded) LIKE "%WmiApRpl.dll"
        OR lower(ImageLoaded) LIKE "%wmiprov.dll"
        OR lower(ImageLoaded) LIKE "%wmiutils.dll"
        OR lower(ImageLoaded) LIKE "%wbemcomn.dll"
        OR lower(ImageLoaded) LIKE "%WMINet_Utils.dll"
        OR lower(ImageLoaded) LIKE "%wbemsvc.dll"
        OR lower(ImageLoaded) LIKE "%fastprox.dll"
        OR lower(Description) LIKE "%wmi%"
    )
    AND NOT (
        lower(Image) LIKE "%wmiprvse.exe"
        OR lower(Image) LIKE "%wmiapsrv.exe"
        OR lower(Image) LIKE "%svchost.exe"
    )
```

Pandas Query

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

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 7)
    & (
      (df['ImageLoaded'].str.lower().str.endswith('wmiclnt.dll', na=Fal
      | (df['ImageLoaded'].str.lower().str.endswith('wmiaprpl.dll', na=
      | (df['ImageLoaded'].str.lower().str.endswith('wmiprov.dll', na=F
      | (df['ImageLoaded'].str.lower().str.endswith('wmiutils.dll', na=
      | (df['ImageLoaded'].str.lower().str.endswith('wbemcomn.dll', na=
      | (df['ImageLoaded'].str.lower().str.endswith('wminet_utils.dll',
      | (df['ImageLoaded'].str.lower().str.endswith('wbemsvc.dll', na=F
      | (df['ImageLoaded'].str.lower().str.endswith('fastprox.dll', na=
      | (df['ImageLoaded'].str.lower().str.contains('.*wmi.*', regex=Tr
    )
    & (
      (~df['Image'].str.lower().str.endswith('wmiprvse.exe', na=False))
      & (~df['Image'].str.lower().str.endswith('wmiapsrv.exe', na=False
      & (~df['Image'].str.lower().str.endswith('svchost.exe', na=False)
    )
]
)
```

Known Bypasses

False Positives

Hunter Notes

- Stack the processes loading WMI modules and document the activity in your environment.
- Stack child processes (if any) of non wmiprvse.exe loading wmi modules

Hunt Output

Type	Link
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/image_load/sysmon_wmi_module_load.yml

References

- https://posts.specterops.io/threat-hunting-with-jupyter-notebooks-part-4-sql-join-via-apache-sparksql-6630928c931e
- https://posts.specterops.io/real-time-sysmon-processing-via-ksql-and-helk-part-3-basic-use-case-8fbf383cb54f

‹ Previous
WMI Eventing

Next ›
Local Service Installation

By Roberto Rodriguez @Cyb3rWard0g