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

Remote WMI Wbemcomn DLL Hijack

Hypothesis

Threat actors might be copying files remotely to abuse a DLL hijack opportunity found on the WMI provider host (wmiprvse.exe).

Technical Context

Windows Management Instrumentation (WMI) is the Microsoft implementation of Web-Based Enterprise Management (WBEM), which is an industry initiative to develop a standard technology for accessing management information in an enterprise environment. WMI uses the Common Information Model (CIM) industry standard to represent systems, applications, networks, devices, and other managed components. WMI resides in a shared service host with several other services. To avoid stopping all the services when a provider fails, providers are loaded into a separate host process named "Wmiprvse.exe". More than one process with this name can be running. The shared host can run under one of the following system accounts in a Wmiprvse.exe host process:

- LocalSystem
- NetworkService
- LocalService When wmiprvse.exe handles a network connection, it runs under the NETWORK SERVICE account. A Threat actor could try to run code as a Network Service user leveraging the WMI provider host process.

Offensive Tradecraft

A threat actor could use a known DLL hijack vulnerability on the execution of wmiprvse.exe to accomplish code execution as a NETWORK SERVICE account. One way to perform a DLL hijack on the WMI provider host is via the wbemcomn DLL. When wmiprvse.exe triggers, it looks for `wbemcomn.dll` in the `C:\Windows\System32\wbem\` directory. That DLL does not exist in that folder. Therefore, a threat actor could easily copy its own DLL in that folder and execute it with the WMI provider host. When the malicious DLL is loaded, there are various approaches to hijacking execution, but most likely a threat actor would want the DLL to act as a proxy to the real DLL to minimize the chances of interrupting normal operations. One way to do this is by cloning the export table from one DLL to another one. One known tool that can help with it is Koppeling.

Pre-Recorded Security Datasets

Metadata	Value
docs	https://securitydatasets.com/notebooks/atomic/windows/lateral_movement/SDWIN-201009173318.html

link	https://raw.githubusercontent.com/OTRF/Security-Datasets/master/datasets/atomic/windows/lateral_movement/host/covenant_wmi_wbemcomn_dll_hijack.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

Look for non-system accounts SMB accessing a `C:\Windows\System32\wbem\wbemcomn.dll` with write (0x2) access mask via an administrative share (i.e C$).

Data source	Event Provider	Relationship	Event
File	Microsoft-Windows-Security-Auditing	User accessed File	5145

Logic

```
SELECT `@timestamp`, Hostname, ShareName, SubjectUserName, SubjectLogonId, IpAd
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 5145
    AND RelativeTargetName LIKE '%wbem\\\wbemcomn.dll'
    AND NOT SubjectUserName LIKE '%$'
    AND AccessMask = '0x2'
```

Pandas Query

```
(
df[['@timestamp','Hostname','ShareName','SubjectUserName','SubjectLogonId','IpA

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 5145)
    & (df['RelativeTargetName'].str.lower().str.endswith('wbem\\wbemcomn.dll',
    & (df['AccessMask'] == '0x2')
    & (~df['SubjectUserName'].str.endswith('$', na=False))
]
.head()
)
```

Analytic II

Look for `C:\Windows\System32\wbem\wbemcomn.dll` being accessed over the network with write (0x2) access mask via an administrative share (i.e C$) and created by the System process on the target system.

Data source	Event Provider	Relationship	Event
File	Microsoft-Windows-Security-Auditing	User accessed File	5145
File	Microsoft-Windows-Sysmon/Operational	Process created File	11

Logic

```
SELECT `@timestamp`, Hostname, ShareName, SubjectUserName, SubjectLogonId, IpAd
FROM dataTable b
INNER JOIN (
    SELECT LOWER(REVERSE(SPLIT(TargetFilename, '\'))[0]) as TargetFilename
    FROM dataTable
    WHERE Channel = 'Microsoft-Windows-Sysmon/Operational'
        AND Image = 'System'
        AND EventID = 11
        AND TargetFilename LIKE '%wbem\\wbemcomn.dll'
) a
ON LOWER(REVERSE(SPLIT(RelativeTargetName, '\'))[0]) = a.TargetFilename
WHERE LOWER(b.Channel) = 'security'
    AND b.EventID = 5145
    AND b.AccessMask = '0x2'
```

Pandas Query

```
fileAccessedDf = (
df[['@timestamp','Hostname','ShareName','SubjectUserName','SubjectLogonId','IpA

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 5145)
    & (df['RelativeTargetName'].str.lower().str.endswith('wbem\\wbemcomn.dll',
    & (df['AccessMask'] == '0x2')
    & (~df['SubjectUserName'].str.endswith('$', na=False))
]
)

fileAccessedDf['Filename'] = fileAccessedDf['RelativeTargetName'].str.split('\\

fileCreateDf = (
df[['@timestamp','Hostname','Image','TargetFilename']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 11)
    & (df['Image'].str.lower() == 'system')
    & (df['TargetFilename'].str.lower().str.endswith('wbem\\wbemcomn.dll', na=F
]
)

fileCreateDf['Filename'] = fileCreateDf['TargetFilename'].str.split('\\').str[-

(
pd.merge(fileAccessedDf, fileCreateDf,
    on = 'Filename', how = 'inner')
)
```

Analytic III

Look for `C:\Windows\System32\wbem\wbemcomn.dll` being accessed over the network with write (0x2) access mask via an administrative share (i.e C$), created by the System process and loaded by the WMI provider host (wmiprvse.exe). All happening on the target system.

Data source	Event Provider	Relationship	Event
File	Microsoft-Windows-Security-Auditing	User accessed File	5145
File	Microsoft-Windows-Sysmon/Operational	Process created File	11
File	Microsoft-Windows-Sysmon/Operational	Process loaded Dll	7

Logic

```
SELECT `@timestamp`, Hostname, ShareName, SubjectUserName, SubjectLogonId, IpAd
FROM dataTable d
INNER JOIN (
    SELECT LOWER(REVERSE(SPLIT(TargetFilename, '\'))[0]) as TargetFilename
    FROM dataTable b
    INNER JOIN (
        SELECT ImageLoaded
        FROM dataTable
        WHERE Channel = 'Microsoft-Windows-Sysmon/Operational'
            AND EventID = 7
            AND LOWER(Image) LIKE '%wmiprvse.exe'
            AND ImageLoaded LIKE '%wbem\\\wbemcomn.dll'
    ) a
    ON b.TargetFilename = a.ImageLoaded
    WHERE b.Channel = 'Microsoft-Windows-Sysmon/Operational'
        AND b.Image = 'System'
        AND b.EventID = 11
) c
ON LOWER(REVERSE(SPLIT(RelativeTargetName, '\'))[0]) = c.TargetFilename
WHERE LOWER(d.Channel) = 'security'
    AND d.EventID = 5145
    AND d.AccessMask = '0x2'
```

Pandas Query

```
fileCreateDf = (
df[['@timestamp','Hostname','Image','TargetFilename']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 11)
]
)

imageLoadedDf = (
df[['@timestamp','Hostname','ImageLoaded','Image']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 7)
    & (df['Image'].str.lower().str.endswith('wmiprvse.exe', na=False))
    & (df['ImageLoaded'].str.lower().str.endswith('wbem\\wbemcomn.dll', na=Fals
]
)

firstJoinDf = (
pd.merge(fileCreateDf, imageLoadedDf,
    left_on = 'TargetFilename', right_on = 'ImageLoaded', how = 'inner')
)

firstJoinDf['Filename'] = firstJoinDf['TargetFilename'].str.split('\\').str[-1]
```

```
fileAccessedDf = (
df[['@timestamp','Hostname','ShareName','SubjectUserName','SubjectLogonId','IpA

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 5145)
    & (df['AccessMask'] == '0x2')
]
)

fileAccessedDf['Filename'] = fileAccessedDf['RelativeTargetName'].str.split('\\

(
pd.merge(firstJoinDf, fileCreateDf,
    on = 'Filename', how = 'inner')
)
```

Known Bypasses

False Positives

Hunter Notes

- Baseline your environment to identify normal activity. Document all accounts creating files over the network via administrative shares.
- Baseline wmiprvse execution and modules loaded (i.e signed and un-signed)

Hunt Output

Type	Link
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_wmiprvse_wbemcomn_dll_hijack.yml
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/file_event/file_event_wmiprvse_wbemcomn_dll_hijack.ym
Sigma Rule	https://github.com/SigmaHQ/sigma/blob/master/rules/windows/image_load/image_load_wmiprvse_wbemcomn_dll_hijack

References

- https://docs.microsoft.com/en-us/windows/win32/wmisdk/about-wmi

Previous
Remote DCOM IErtUtil DLL Hijack

Next
SMB Create Remote File

By Roberto Rodriguez @Cyb3rWard0g