
Threat Hunter Playbook

Search this book...



# Remote Service Control Manager Handle



## Hypothesis

Adversaries might be attempting to open up a handle to the service control manager (SCM) database on remote endpoints to check for local admin access in my environment.

## Technical Context

Often times, when an adversary lands on an endpoint, the current user does not have local administrator privileges over the compromised system. While some adversaries consider this situation a dead end, others find it very interesting to identify which machines on the network the current user has administrative access to. One common way to accomplish this is by attempting to open up a handle to the service control manager (SCM) database on remote endpoints in the network with SC_MANAGER_ALL_ACCESS (0xF003F) access rights. The Service Control Manager (SCM) is a remote procedure call (RPC) server, so that service configuration and service control programs can manipulate services on remote machines. Only processes with Administrator privileges are able to open a handle to the SCM database. This database is also known as the ServicesActive database. Therefore, it is very effective to check if the current user has administrative or local admin access to other endpoints in the network.

## Offensive Tradecraft

An adversary can simply use the Win32 API function OpenSCManagerA to attempt to establish a connection to the service control manager (SCM) on the specified computer and open the service control manager database. If this succeeds (A non-zero handle is returned), the current user context has local administrator acess to the remote host.

Additional reading

- https://github.com/OTRF/ThreatHunter-Playbook/tree/master/docs/library/windows/service_control_manager.md

## Pre-Recorded Security Datasets

| Metadata | Value |
|---|---|
| docs | https://securitydatasets.com/notebooks/atomic/windows/07_discovery/SDWIN-190518224039.html |
| link | https://raw.githubusercontent.com/OTRF/Security-Datasets/master/datasets/atomic/windows/discovery/host/empire_find_localadmin_smb_svcctl_OpenSCManager.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

Detects non-system users failing to get a handle of the SCM database.

| Data source | Event Provider | Relationship | Event |
|---|---|---|---|
| File | Microsoft-Windows-Security-Auditing | User requested access File | 4656 |

### Logic

```
SELECT `@timestamp`, Hostname, SubjectUserName, ProcessName, ObjectName
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 4656
    AND ObjectType = "SC_MANAGER OBJECT"
    AND ObjectName = "ServicesActive"
    AND AccessMask = "0xf003f"
    AND NOT SubjectLogonId = "0x3e4"
```

### Pandas Query

```
(
df[['@timestamp','Hostname','SubjectUserName','ProcessName','ObjectName']]

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4656)
    & (df['ObjectType'] == 'SC_MANAGER OBJECT')
    & (df['ObjectName'] == 'ServicesActive')
    & (df['AccessMask'] == '0xf003f')
    & (df['SubjectLogonId'] != '0x3e4')
]
.head()
)
```

## Analytic II

Look for non-system accounts performing privileged operations on protected subsystem objects such as the SCM database.

| Data source | Event Provider | Relationship | Event |
|---|---|---|---|
| File | Microsoft-Windows-Security-Auditing | User requested access File | 4674 |

### Logic

```
SELECT `@timestamp`, Hostname, SubjectUserName, ProcessName, ObjectName, Privil
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 4674
    AND ObjectType = "SC_MANAGER OBJECT"
    AND ObjectName = "ServicesActive"
    AND PrivilegeList = "SeTakeOwnershipPrivilege"
    AND NOT SubjectLogonId = "0x3e4"
```

### Pandas Query

```
(
df[['@timestamp','Hostname','SubjectUserName','ProcessName','ObjectName','Privi

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4674)
    & (df['ObjectType'] == 'SC_MANAGER OBJECT')
    & (df['ObjectName'] == 'ServicesActive')
    & (df['PrivilegeList'] == 'SeTakeOwnershipPrivilege')
    & (df['SubjectLogonId'] != '0x3e4')
]
.head()
)
```

## Analytic III

Look for inbound network connections to services.exe from other endpoints in the network. Same SourceAddress, but different Hostname.

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

### Logic

```
SELECT `@timestamp`, Hostname, Application, SourcePort, SourceAddress, DestPort
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 5156
    AND Application LIKE "%\\\services.exe"
    AND LayerRTID = 44
```

### Pandas Query

```
(
df[['@timestamp','Hostname','Application','SourcePort','SourceAddress','DestPor
```

```
[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 5156)
    & (df['Application'].str.lower().str.endswith('services.exe', na=False))
    & (df['LayerRTID'] == 44)
]
)
```

## Analytic IV

Look for several network connection maded by services.exe from different endpoints to the same destination.

| Data source | Event Provider | Relationship | Event |
|---|---|---|---|
| Process | Microsoft-Windows-Security-Auditing | Process connected to Port | 3 |
| Process | Microsoft-Windows-Security-Auditing | Process connected to Ip | 3 |

### Logic

```
SELECT `@timestamp`, Hostname, User, SourcePort, SourceIp, DestinationPort, Des
FROM dataTable
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
    AND EventID = 3
    AND Image LIKE "%\\\services.exe"
```

### Pandas Query

```
(
df[['@timestamp','Hostname','User','SourcePort','SourceIp','DestinationPort','D

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
    & (df['EventID'] == 3)
    & (df['Image'].str.lower().str.endswith('services.exe', na=False))
]
)
```

## Analytic V

Look for non-system accounts performing privileged operations on protected subsystem objects such as the SCM database from other endpoints in the network.

| Data source | Event Provider | Relationship | Event |
|---|---|---|---|
| Authentication log | Microsoft-Windows-Security-Auditing | User authenticated Host | 4624 |
| File | Microsoft-Windows-Security-Auditing | User requested access File | 4656 |

### Logic

```
SELECT o.`@timestamp`, o.Hostname, o.SubjectUserName, o.ObjectType,o.ObjectName
FROM dataTable o
INNER JOIN (
```

```
    SELECT Hostname,TargetUserName,TargetLogonId,IpAddress
    FROM dataTable
    WHERE LOWER(Channel) = "security"
        AND EventID = 4624
        AND LogonType = 3
        AND NOT TargetUserName LIKE "%$"
    ) a
ON o.SubjectLogonId = a.TargetLogonId
WHERE LOWER(o.Channel) = "security"
    AND o.EventID = 4656
    AND NOT o.SubjectLogonId = "0x3e4"
```

Pandas Query

```
handleRequestDf = (
df[['@timestamp','Hostname','SubjectUserName','SubjectLogonId']]

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4656)
    & (df['SubjectLogonId'] != '0x3e4')
]
)

networkLogonDf = (
df[['@timestamp', 'Hostname', 'TargetUserName', 'TargetLogonId', 'IpAddress']]

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4624)
    & (df['LogonType'] == 3)
    & (~df['SubjectUserName'].str.lower().str.endswith('$', na=False))
]
)

(
pd.merge(handleRequestDf, networkLogonDf,
  left_on = 'SubjectLogonId', right_on = 'TargetLogonId', how = 'inner')
)
```

# Known Bypasses

# False Positives

# Hunter Notes

- Event id 4656 gets generated only when the OpenSCManager API call fails to get a handle to the SCM database. There is not SACL for SCM database so success attempts will not be logged.
- Event id 4674 gets triggered when the SCM database is accessed. Filter known or common accounts that obtain a handle to SCM on a regular basis (i.e vulnerability scanners)
- You can join security events 4674 and security events 4624 on the LogonID field and filter results on logon type 3 or network to add more context to your query and look for handles to SCM from remote endpoints.
- Look for the same endpoint or IP address to many remote hosts to find potential aggressive attempts.
- You can also join security events 4674 where the object name is servicesactive (SCM database) with other security events on the object handle. This will allow you to identify what was actually done after the handle was opened. For example, the same handle can be used to create a service (i.e. PSEXESVC)

- Event id 5156 gets generated on the target as an inbound network event with process name services.exe. You might have to stack the SourceAddress field value based on your environment noise.

## Hunt Output

| Type | Link |
| --- | --- |
| Sigma Rule | https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_scm_database_handle_failure.yml |
| Sigma Rule | https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_scm_database_privileged_operation.y |

## References


- https://docs.microsoft.com/en-us/windows/win32/services/service-security-and-access-rights
- https://github.com/EmpireProject/Empire/blob/dev/data/module_source/situational_awareness/network/powerview.ps1#L15473
- https://github.com/rapid7/metasploit-framework/blob/master/modules/post/windows/gather/local_admin_search_enum.rb#L217
- https://github.com/nettitude/PoshC2_Python/blob/master/Modules/Get-System.ps1#L222
- https://www.pentestgeek.com/metasploit/find-local-admin-with-metasploit
- http://www.harmj0y.net/blog/penetesting/finding-local-admin-with-the-veil-framework/
- https://www.slideshare.net/harmj0y/derbycon-the-unintended-risks-of-trusting-active-directory
- https://docs.microsoft.com/en-us/dotnet/api/system.serviceprocess.servicebase.servicehandle?view=netframework-4.8
- https://community.rsa.com/community/products/netwitness/blog/2019/04/10/detecting-lateral-movement-in-rsa-netwitness-winexe


Previous
Remote Service creation

Next
Remote Interactive Task Manager LSASS Dump

By Roberto Rodriguez @Cyb3rWard0g