



| Data source | Event Provider                      | Relationship             | Event |
|-------------|-------------------------------------|--------------------------|-------|
| Process     | Microsoft-Windows-Security-Auditing | Process accessed Process | 4663  |

|         |                                     |                                  |      |
|---------|-------------------------------------|----------------------------------|------|
| Process | Microsoft-Windows-Security-Auditing | Process requested_access Process | 4656 |
|---------|-------------------------------------|----------------------------------|------|

Logic

```
SELECT `@timestamp`, Hostname, SubjectUserName, ProcessName, ObjectName, AccessMask
FROM dataTable
WHERE LOWER(Channel) = "security"
      AND (EventID = 4663 OR EventID = 4656)
      AND ObjectName LIKE "%lsass.exe"
      AND NOT SubjectUserName LIKE "%$"
```

Pandas Query

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

[(df['Channel'].str.lower() == 'security')
 & ((df['EventID'] == 4663) | (df['EventID'] == 4656))
 & (df['ObjectName'].str.contains('.*lsass.exe', regex=True))
 & (~df['SubjectUserName'].str.endswith('.*$', na=False))
]
.head()
)
```

Analytic II

Look for processes opening handles and accessing Lsass with potential dlls in memory (i.e UNKNOWN in CallTrace).

| Data source | Event Provider                       | Relationship             | Event |
|-------------|--------------------------------------|--------------------------|-------|
| Process     | Microsoft-Windows-Sysmon/Operational | Process accessed Process | 10    |

Logic

```
SELECT `@timestamp`,Hostname,SourceImage,TargetImage,GrantedAccess,SourceProcessName
FROM dataTable
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
      AND EventID = 10
      AND TargetImage LIKE "%lsass.exe"
      AND CallTrace LIKE "%UNKNOWN%"
```

Pandas Query

```
(
df[['@timestamp', 'Hostname', 'SourceImage', 'TargetImage', 'GrantedAccess', 'SourceProcessName']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
 & (df['EventID'] == 10)
 & (df['TargetImage'].str.contains('.*lsass.exe', regex=True))
 & (df['CallTrace'].str.contains('.*UNKNOWN*', regex=True))
]
.head()
)
```

Analytic III

Look for processes loading a few known DLLs loaded by tools like Mimikatz to interact with credentials.

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

Logic

```
SELECT ProcessGuid,Image, COUNT(DISTINCT ImageLoaded) AS hits
FROM dataTable
```

```
WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
AND EventID = 7
AND (
    ImageLoaded LIKE "%samlib.dll"
    OR ImageLoaded LIKE "%vaultcli.dll"
    OR ImageLoaded LIKE "%hid.dll"
    OR ImageLoaded LIKE "%wincard.dll"
    OR ImageLoaded LIKE "%cryptdll.dll"
)
AND '@timestamp' BETWEEN "2019-03-00 00:00:00.000" AND "2019-06-20 00:00:00.000"
GROUP BY ProcessGuid,Image ORDER BY hits DESC LIMIT 10
```

Pandas Query

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

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
 & (df['EventID'] == 7)
 & (
    (df['ImageLoaded'].str.contains('.*samlib.dll', regex=True))
    | (df['ImageLoaded'].str.contains('.*vaultcli.dll', regex=True))
    | (df['ImageLoaded'].str.contains('.*hid.dll', regex=True))
    | (df['ImageLoaded'].str.contains('.*wincard.dll', regex=True))
    | (df['ImageLoaded'].str.contains('.*cryptdll.dll', regex=True))
    )
 & (df['@timestamp'].between('2020-06-00 00:00:00.000','2020-08-20 00:00:00.000'))
]
.groupby(['ProcessGuid','Image'])['ImageLoaded'].count().sort_values(ascending=False)
).to_frame()
```

Analytic IV

Join processes opening a handle and accessing LSASS with potential DLLs loaded in memory and processes loading a few known DLLs loaded by tools like Mimikatz to interact with credentials.

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

|         |                                      |                             |    |
|---------|--------------------------------------|-----------------------------|----|
| Process | Microsoft-Windows-Sysmon/Operational | Process accessed<br>Process | 10 |
|---------|--------------------------------------|-----------------------------|----|

Logic

```
SELECT a.'@timestamp', a.Hostname, m.Image, a.SourceProcessGUID
FROM dataTable a
INNER JOIN (
    SELECT ProcessGuid,Image, COUNT(DISTINCT ImageLoaded) AS hits
    FROM dataTable
    WHERE Channel = "Microsoft-Windows-Sysmon/Operational"
    AND EventID = 7
    AND (
        ImageLoaded LIKE "%samlib.dll"
        OR ImageLoaded LIKE "%vaultcli.dll"
        OR ImageLoaded LIKE "%hid.dll"
        OR ImageLoaded LIKE "%wincard.dll"
        OR ImageLoaded LIKE "%cryptdll.dll"
    )
    AND '@timestamp' BETWEEN "2020-06 00:00:00.000" AND "2020-10-20 00:00:00.000"
    GROUP BY ProcessGuid,Image ORDER BY hits DESC LIMIT 10
) m
ON a.SourceProcessGUID = m.ProcessGuid
WHERE a.Channel = "Microsoft-Windows-Sysmon/Operational"
AND a.EventID = 10
AND a.TargetImage LIKE "%lsass.exe"
AND a.CallTrace LIKE "%UNKNOWN%"
AND m.hits >= 3
```

Pandas Query

```
imageLoadDf = (
df[['@timestamp','ProcessGuid','Image','ImageLoaded']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
 & (df['EventID'] == 7)
 & (
    (df['ImageLoaded'].str.contains('.*samlib.dll', regex=True))
    | (df['ImageLoaded'].str.contains('.*vaultcli.dll', regex=True))
    | (df['ImageLoaded'].str.contains('.*hid.dll', regex=True))
    | (df['ImageLoaded'].str.contains('.*wincard.dll', regex=True))
    | (df['ImageLoaded'].str.contains('.*cryptdll.dll', regex=True))
    )
 & (df['@timestamp'].between('2020-06-00 00:00:00.000','2020-08-20 00:00:00.000'))
]
.groupby(['ProcessGuid','Image'])['ImageLoaded'].count().sort_values(ascending=False)
```

```
).to_frame()

processAccessDf = (
df[['@timestamp', 'Hostname', 'SourceImage', 'TargetImage', 'GrantedAccess', 'SourceProcessGUID']]

[(df['Channel'] == 'Microsoft-Windows-Sysmon/Operational')
 & (df['EventID'] == 10)
 & (df['TargetImage'].str.contains('.*lsass.exe', regex=True))
 & (df['CallTrace'].str.contains('.*UNKNOWN*', regex=True))
]
)

(
pd.merge(imageLoadDf, processAccessDf,
left_on = 'ProcessGuid', right_on = 'SourceProcessGUID', how = 'inner')
)
```

## Known Bypasses

| Idea  | Playbook |
|---|----------|
| Inject into known processes accessing LSASS on a regular basis such as pmfexe.exe and as System |          |

## False Positives

- The Microsoft Monitoring Agent binary pmfexe.exe is one of the most common ones that accesses Lsass.exe with at least 0x10 permissions as System. That could be useful to blend in through the noise.

## Hunter Notes

- Looking for processes accessing LSASS with the 0x10(VmRead) rights from a non-system account is very suspicious and not as common as you might think.
- GrantedAccess code 0x1010 is the new permission Mimikatz v.20170327 uses for command "sekurlsa::logonpasswords". You can specifically look for that from processes like PowerShell to create a basic signature.
  - 0x00000010 = VMRead
  - 0x00001000 = QueryLimitedInfo
- GrantedAccess code 0x1010 is less common than 0x1410 in large environment.
- Out of all the Modules that Mimikatz needs to function, there are 5 modules that when loaded all together by the same process is very suspicious.
  - samlib.dll, vaultcli.dll, hid.dll, winscard.dll, cryptdll.dll
- For signatures purposes, look for processes accessing Lsass.exe with a potential CallTrace  
Pattern> /C:\Windows\SYSTEM32\ntdll.dll+[a-zA-Z0-9]{1,}|C:\Windows\System32\KERNELBASE.dll+[a-zA-Z0-9]{1,}|UNKNOWN.\*/- You could use a stack counting technique to stack all the values of the permissions invoked by processes accessing Lsass.exe. You will have to do some filtering to reduce the number of false positives. You could then group the results with other events such as modules being loaded (EID 7). A time window of 1-2 seconds could help to get to a reasonable number of events for analysis.

## Hunt Output

| Type       | Link  |
|------------|---|
| Sigma Rule | <a href="https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_lsass_access_non_system_account.yml">https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_lsass_access_non_system_account.yml</a> |

## References

- <https://tyranidslair.blogspot.com/2017/10/bypassing-sacl-auditing-on-lsass.htmls>
- [https://adsecurity.org/?page\\_id=1821#SEKURLSALogonPasswords](https://adsecurity.org/?page_id=1821#SEKURLSALogonPasswords)
- <https://github.com/PowerShellMafia/PowerSploit/tree/dev>
- <http://clymb3r.wordpress.com/2013/04/09/modifying-mimikatz-to-be-loaded-using-invoke-reflectedllinjection-ps1/>

[◀ Windows](#)

[DLL Process Injection via ▶](#)  
[CreateRemoteThread and LoadLibrary](#)

By Roberto Rodriguez @Cyb3rWard0g  
© Copyright 2022.