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

# Domain DPAPI Backup Key Extraction

Contents

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

## Hypothesis

Adversaries might be extracting the DPAPI domain backup key from my DC to be able to decrypt any domain user master key files.

## Technical Context

Starting with MicrosoftÂ® WindowsÂ® 2000, the operating system began to provide a data protection application-programming interface (API). This Data Protection API (DPAPI) is a pair of function calls (CryptProtectData / CryptUnprotectData) that provide operating system-level data protection services to user and system processes. DPAPI initially generates a strong key called a MasterKey, which is protected by the user's password. DPAPI uses a standard cryptographic process called Password-Based Key Derivation to generate a key from the password. This password-derived key is then used with Triple-DES to encrypt the MasterKey, which is finally stored in the user's profile directory.

When a computer is a member of a domain, DPAPI has a backup mechanism to allow unprotection of the data. When a MasterKey is generated, DPAPI talks to a Domain Controller. Domain Controllers have a domain-wide public/private key pair, associated solely with DPAPI. The local DPAPI client gets the Domain Controller public key from a Domain Controller by using a mutually authenticated and privacy protected RPC call. The client encrypts the MasterKey with the Domain Controller public key. It then stores this backup MasterKey along with the MasterKey protected by the user's password.

## Offensive Tradecraft

If an adversary obtains domain admin (or equivalent) privileges, the domain backup key can be stolen and used to decrypt any domain user master key. Tools such as Mimikatz with the method/module lsadump::backupkeys can be used to extract the domain backup key. It uses the LsaOpenPolicy/LsaRetrievePrivateData API calls (instead of MS-BKRP) to retrieve the value for the G BCKUPKEY_P LSA secrets.

Additional reading

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

## Pre-Recorded Security Datasets

| Metadata | Value |
|---|---|
| docs | https://securitydatasets.com/notebooks/atomic/windows/credential_access/SDWIN-190518235535.html |

| | |
|---|---|
| link | https://raw.githubusercontent.com/OTRF/Security-Datasets/master/datasets/atomic/windows/credential_access/host/empire_mimikatz_backupkeys_dcerpc_smb_lsarpc.zi |

## 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

Monitor for any SecretObject with the string BCKUPKEY in the ObjectName.

| Data source | Event Provider | Relationship | Event |
|---|---|---|---|
| Windows active directory | Microsoft-Windows-Security-Auditing | User accessed AD Object | 4662 |

### Logic

```
SELECT `@timestamp`, Hostname, ObjectServer, ObjectType, ObjectName
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 4662
    AND AccessMask = "0x2"
    AND lower(ObjectName) LIKE "%bckupkey%"
```

### Pandas Query

```
(
df[['@timestamp','Hostname','ObjectServer','ObjectType','ObjectName']]

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4662)
    & (df['AccessMask'] == '0x2')
    & (df['Message'].str.lower().str.contains('.*bckupkey.*', regex=True))
]
.head()
)
```

## Analytic II

We can get the user logon id of the user that accessed the *bckupkey* object and JOIN it with a successful logon event (4624) user logon id to find the source IP.

| Data source | Event Provider | Relationship | Event |
|---|---|---|---|
| Authentication log | Microsoft-Windows-Security-Auditing | User authenticated Host | 4624 |
| Windows active directory | Microsoft-Windows-Security-Auditing | User accessed AD Object | 4662 |

### Logic

```
SELECT o.`@timestamp`, o.Hostname, o.ObjectName, a.IpAddress
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(Channel) = "security"
    AND o.EventID = 4662
    AND o.AccessMask = "0x2"
    AND lower(o.ObjectName) LIKE "%bckupkey%"
    AND o.Hostname = a.Hostname
```

### Pandas Query

```
backupKeyDf = (
df[['@timestamp','Hostname','ObjectName','IpAddress','SubjectLogonId']]

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4662)
    & (df['AccessMask'] == '0x2')
    & (df['ObjectName'].str.lower().str.contains('.*bckupkey.*', regex=True))
]
.head()
)

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

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

(
pd.merge(backupKeyDf, networkLogonDf,
    left_on = ['SubjectLogonId','Hostname'], right_on = ['TargetLogonId','Hostr
)
```

## Analytic III

Monitoring for access to the protected_storage named pipe via SMB is very interesting to identify potential DPAPI activity over the network. Mimikatz uses the Lsarpc named pipe now.

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

### Logic

```
SELECT `@timestamp`, Hostname, SubjectUserName, ShareName, RelativeTargetName,
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 5145
    AND ShareName LIKE "%IPC%"
    AND RelativeTargetName = "protected_storage"
```

### Pandas Query

```
(
df[['@timestamp','Hostname','SubjectUserName','ShareName','RelativeTargetName']

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 5145)
    & (df['RelativeTargetName'].str.lower() == 'protected_storage')
    & (df['ShareName'].str.contains('.*IPC.*', regex=True))
]
)
```

## Analytic IV

This event generates every time that a backup is attempted for the DPAPI Master Key. When a computer is a member of a domain, DPAPI has a backup mechanism to allow unprotection of the data. When a Master Key is generated, DPAPI communicates with a domain controller. It migt be aleready created and this event might not trigger.

| Data source | Event Provider | Relationship | Event |
|---|---|---|---|
| File | Microsoft-Windows-Security-Auditing | User requested access File | 4692 |

### Logic

```
SELECT `@timestamp`, Hostname, SubjectUserName
FROM dataTable
WHERE LOWER(Channel) = "security"
    AND EventID = 4692
```

### Pandas Query

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

[(df['Channel'].str.lower() == 'security')
    & (df['EventID'] == 4692)
]
)
```

## Known Bypasses

# False Positives

# Hunter Notes

- Backup key can be displayed as base64 blob or exported as a .pvk file on disk (Mimikatz-like)
- Windows security event 4692 (Backup of data protection master key was attempted) also generates every time a new DPAPI Master Key is generated
- When a computer is a member of a domain, DPAPI has a backup mechanism to allow unprotection of the data. When a Master Key is generated, DPAPI communicates with a domain controller.

# Hunt Output

| Type | Link |
|---|---|
| Sigma Rule | https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_dpapi_domain_backupkey_extraction. |
| Sigma Rule | https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_protected_storage_service_access.ym |
| Sigma Rule | https://github.com/SigmaHQ/sigma/blob/master/rules/windows/builtin/security/win_dpapi_domain_masterkey_backup_at |

# References


- https://www.harmj0y.net/blog/redteaming/operational-guidance-for-offensive-user-dpapi-abuse/
- https://digital-forensics.sans.org/summit-archives/dfirprague14/Give_Me_the_Password_and_Ill_Rule_the_World_Francesco_Picasso.pdf
- https://docs.microsoft.com/en-us/windows/desktop/devnotes/pstore
- https://github.com/gentilkiwi/mimikatz/blob/641a3b29acd326d07269300d94dceafea041f760/mimikatz/modules/kuhl_m_lsadu
- https://github.com/GhostPack/SharpDPAPI/blob/6388040a92e59fc0d5a82b4ec31599aa6778fd3b/SharpDPAPI/lib/Backup.cs#L4
- https://github.com/gentilkiwi/mimikatz/blob/641a3b29acd326d07269300d94dceafea041f760/mimikatz/modules/kuhl_m_lsadu L1926
- https://github.com/gentilkiwi/mimikatz/blob/641a3b29acd326d07269300d94dceafea041f760/mimikatz/modules/kuhl_m_lsadu
- https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-lsad/88c6bd18-6c40-4a82-ae19-fe7bfec5108b


Previous
Alternate PowerShell Hosts

Next
SysKey Registry Keys Access

By Roberto Rodriguez @Cyb3rWard0g
© Copyright 2022.