matrixfox / **msfconsole-commands.md**

Created 10 years ago

Code Revisions 1 Stars 3 Forks 4

msfconsole-commands.md

MSFconsole core commands tutorial

The msfconsole has many different command options to chose from. The following are a core set of Metasploit commands with reference to their output.

https://www.offensive-security.com/wp-content/uploads/2015/04/msfconsole-core-commands.png msfconsole core commands | Metasploit Unleashed

```
back          Move back from the current context
banner        Display an awesome metasploit banner
cd            Change the current working directory
color         Toggle color
connect       Communicate with a host
edit          Edit the current module with $VISUAL or $EDITOR
exit          Exit the console
get           Gets the value of a context-specific variable
getg          Gets the value of a global variable
go_pro        Launch Metasploit web GUI

grep          Grep the output of another command
help          Help menu
info          Displays information about one or more module
irb           Drop into irb scripting mode
jobs          Displays and manages jobs
kill          Kill a job
load          Load a framework plugin
loadpath      Searches for and loads modules from a path
makerc        Save commands entered since start to a file
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
quit          Exit the console
reload_all    Reloads all modules from all defined module paths
rename_job    Rename a job
resource      Run the commands stored in a file
route         Route traffic through a session
save          Saves the active datastores
search        Searches module names and descriptions
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
show          Displays modules of a given type, or all modules
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
unload        Unload a framework plugin
unset         Unsets one or more context-specific variables
unsetg        Unsets one or more global variables
use           Selects a module by name
version       Show the framework and console library version numbers
```

back

Once you have finished working with a particular module, or if you inadvertently select the wrong module, you can issue the ‘back’ command to move out of the current context. This, however is not required. Just as you can in commercial routers, you can switch modules from within other modules. As a reminder, variables will only carry over if they are set globally.

```
msf auxiliary(ms09_001_write) > back
msf >
```

banner

Simply displays a randomly selected banner

```
msf > banner
 _                                                    _
/ \    /\         __                         _   __  /_/ __
```

```
| |\  / | _____   \ \           ___   _____ | | /  \ _   \ \
| | \/| | | ___\ |- -|   /\    / __\ | -__/ | || | || | |- -|
|_|   | | | _|__  | |_  / -\ __\ \   | |    | | \__/| |  | |_
      |/  |____/  \___\/ /\ \\___/   \/     \__|    |_\  \___\

Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with
Metasploit Pro -- type 'go_pro' to launch it now.

       =[ metasploit v4.11.4-2015071402                   ]
+ -- --=[ 1467 exploits - 840 auxiliary - 232 post        ]
+ -- --=[ 432 payloads - 37 encoders - 8 nops             ]
```

check

There aren’t many exploits that support it, but there is also a ‘check’ option that will check to see if a target is vulnerable to a particular exploit instead of actually exploiting it.

```
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

    Name     Current Setting  Required  Description
    ----     ---------------  --------  -----------
    RHOST    172.16.194.134   yes       The target address
    RPORT    445              yes       Set the SMB service port
    SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER,
SRVSVC)

Exploit target:

    Id  Name
    --  ----
    0   Automatic Targeting

msf exploit(ms08_067_netapi) > check

[*] Verifying vulnerable status... (path: 0x0000005a)
[*] System is not vulnerable (status: 0x00000000)
[*] The target is not exploitable.
msf  exploit(ms08_067_netapi) >
```

color

You can enable or disable if the output you get through the msfconsole will contain colors.

```
msf > color
Usage: color <'true'|'false'|'auto'>

Enable or disable color output.
```

connect

There is a miniature netcat clone built into the msfconsole that supports SSL, proxies, pivoting, and file sends. By issuing the ‘connect’ command with an ip address and port number, you can connect to a remote host from within msfconsole the same as you would with netcat or telnet.

```
msf > connect 192.168.1.1 23
[*] Connected to 192.168.1.1:23
DD-WRT v24 std (c) 2008 NewMedia-NET GmbH
Release: 07/27/08 (SVN revision: 10011)
DD-WRT login:
```

You can see all the additional options by issuing the “-h” parameter.

```
msf > connect -h
Usage: connect [options]

Communicate with a host, similar to interacting via netcat, taking
advantage of
any configured session pivoting.

OPTIONS:

    -C        Try to use CRLF for EOL sequence.
    -P <opt>  Specify source port.
    -S <opt>  Specify source address.
    -c <opt>  Specify which Comm to use.
    -h        Help banner.
    -i <opt>  Send the contents of a file.
    -p <opt>  List of proxies to use.
    -s        Connect with SSL.
    -u        Switch to a UDP socket.
    -w <opt>  Specify connect timeout.
    -z        Just try to connect, then return.

msf >
```

edit

The edit command will edit the current module with $VISUAL or $EDITOR. By default this will open the current module in Vim.

```
msf exploit(ms10_061_spoolss) > edit
[*] Launching /usr/bin/vim /usr/share/metasploit-framework/modules/
exploits/windows/smb/ms10_061_spoolss.rb

##
# This module requires Metasploit: http//metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'
require 'msf/windows_error'

class Metasploit3 < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::DCERPC
  include Msf::Exploit::Remote::SMB
  include Msf::Exploit::EXE
  include Msf::Exploit::WbemExec

  def initialize(info = {})
```

exit

The exit command will simply exit msfconsole.

```
msf exploit(ms10_061_spoolss) > exit
root@kali:~#
```

help

The help command will give you a list and small description of all available commands.

```
msf > help

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

    Command       Description
    -------       -----------
    ?             Help menu
    back          Move back from the current context
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
...snip...

Database Backend Commands
=========================

    Command           Description
    -------           -----------
    creds             List all credentials in the database
    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)
...snip...
```

info

The info command will provide detailed information about a particular module including all options, targets, and other information. Be sure to always read the module description prior to using it as some may have un-desired effects.

The info command also provides the following information:

- The author and licensing information
- Vulnerability references (ie: CVE, BID, etc)
- Any payload restrictions the module may have

```
msf  exploit(ms09_050_smb2_negotiate_func_index) > info exploit/windows/
```

```
smb/ms09_050_smb2_negotiate_func_index

       Name: Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table
Dereference
     Module: exploit/windows/smb/ms09_050_smb2_negotiate_func_index
    Version: 14774
   Platform: Windows
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Good

Provided by:
  Laurent Gaffie <laurent.gaffie@gmail.com>
  hdm <hdm@metasploit.com>
  sf <stephen_fewer@harmonysecurity.com>

Available targets:
  Id  Name
  --  ----
  0   Windows Vista SP1/SP2 and Server 2008 (x86)

Basic options:
  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST                   yes       The target address
  RPORT  445              yes       The target port
  WAIT   180              yes       The number of seconds to wait for
the attack to complete.

Payload information:
  Space: 1024

Description:
  This module exploits an out of bounds function table dereference in
  the SMB request validation code of the SRV2.SYS driver included with
  Windows Vista, Windows 7 release candidates (not RTM), and Windows
  2008 Server prior to R2. Windows Vista without SP1 does not seem
  affected by this flaw.

References:
  http://www.microsoft.com/technet/security/bulletin/MS09-050.mspx
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=2009-3103
  http://www.securityfocus.com/bid/36299
  http://www.osvdb.org/57799
  http://seclists.org/fulldisclosure/2009/Sep/0039.html
  http://www.microsoft.com/technet/security/Bulletin/MS09-050.mspx

msf  exploit(ms09_050_smb2_negotiate_func_index) >
```

irb

Running the irb command will drop you into a live Ruby interpreter shell where you can issue commands and create Metasploit scripts on the fly. This feature is also very useful for understanding the internals of the Framework.

```
msf > irb
[*] Starting IRB shell...

>> puts "Hello, metasploit!"
Hello, metasploit!
=> nil
>> Framework::Version
=> "4.8.2-2014022601"
```

jobs

Jobs are modules that are running in the background. The jobs command provides the ability to list and terminate these jobs.

```
msf > jobs -h
Usage: jobs [options]

Active job manipulation and interaction.

OPTIONS:

    -K        Terminate all running jobs.
    -h        Help banner.
    -i <opt>  Lists detailed information about a running job.
    -k <opt>  Terminate the specified job name.
    -l        List all running jobs.
    -v        Print more detailed info.  Use with -i and -l

msf >
```

kill

The kill command will kill any running jobs when supplied with the job id.

```
msf exploit(ms10_002_aurora) > kill 0
```

```
Stopping job: 0...

[*] Server stopped.
```

load

The load command loads a plugin from Metasploit’s plugin directory. Arguments are passed as key=val on the shell.

```
msf > load
Usage: load <path> [var=val var=val ...]

Loads a plugin from the supplied path.  If path is not absolute, first
looks
in the user's plugin directory (/root/.msf4/plugins) then
in the framework root plugin directory (/usr/share/metasploit-framework/
plugins).
The optional var=val options are custom parameters that can be passed to
plugins.

msf > load pcap_log
[*] PcapLog plugin loaded.
[*] Successfully loaded plugin: pcap_log
```

loadpath

The loadpath command will load a third-part module tree for the path so you can point Metasploit at your 0-day exploits, encoders, payloads, etc.

```
msf > loadpath /home/secret/modules

Loaded 0 modules.
```

unload

Conversely, the unload command unloads a previously loaded plugin and removes any extended commands.

```
msf > unload pcap_log
Unloading plugin pcap_log...unloaded.
```

resource

The resource command runs resource (batch) files that can be loaded through msfconsole.

```
msf > resource
Usage: resource path1 [path2 ...]

Run the commands stored in the supplied files.  Resource files may also
contain
ruby code between  tags.

See also: makerc
```

Some attacks such as Karmetasploit use resource files to run a set of commands in a karma.rc file to create an attack. Later on we will discuss how, outside of Karmetasploit, that can be very useful.

```
msf > resource karma.rc
[*] Processing karma.rc for ERB directives.
resource (karma.rc)> db_connect msf3:PASSWORD@127.0.0.1:7175/msf3
resource (karma.rc)> use auxiliary/server/browser_autopwn
...snip...
```

Batch files can greatly speed up testing and development times as well as allow the user to automate many tasks. Besides loading a batch file from within msfconsole, they can also be passed at startup using the ‘-r’ flag. The simple example below creates a batch file to display the Metasploit version number at startup.

```
root@kali:~# echo version > version.rc
root@kali:~# msfconsole -r version.rc


 _                                                    _
/ \    /\         __                         _   __  /_/ __
| |\  / | _____   \ \           ___   _____ | | /  \ _   \ \
| | \/| | | ___\ |- -|   /\    / __\ | -__/ | || | || | |- -|
|_|   | | | _|__  | |_  / -\ __\ \   | |    | | \__/| |  | |_
      |/  |____/  \___\/ /\ \\___/   \/     \__|    |_\  \___\

Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with
Metasploit Pro -- type 'go_pro' to launch it now.
```

```
       =[ metasploit v4.8.2-2014021901 [core:4.8 api:1.0] ]
+ -- --=[ 1265 exploits - 695 auxiliary - 202 post ]
+ -- --=[ 330 payloads - 32 encoders - 8 nops      ]

[*] Processing version.rc for ERB directives.
resource (version.rc)> version
Framework: 4.8.2-2014022601
Console  : 4.8.2-2014022601.15168
msf >
```

route

The “route” command in Metasploit allows you to route sockets through a session or ‘comm’, providing basic pivoting capabilities. To add a route, you pass the target subnet and network mask followed by the session (comm) number.

```
meterpreter > route -h
Usage: route [-h] command [args]

Display or modify the routing table on the remote machine.

Supported commands:

   add    [subnet] [netmask] [gateway]
   delete [subnet] [netmask] [gateway]
   list

meterpreter >

meterpreter > route

Network routes
==============

    Subnet           Netmask          Gateway
    ------           -------          -------
    0.0.0.0          0.0.0.0          172.16.1.254
    127.0.0.0        255.0.0.0        127.0.0.1
    172.16.1.0       255.255.255.0    172.16.1.100
    172.16.1.100     255.255.255.255  127.0.0.1
    172.16.255.255   255.255.255.255  172.16.1.100
    224.0.0.0        240.0.0.0        172.16.1.100
    255.255.255.255  255.255.255.255  172.16.1.100
```

search

The msfconsole includes an extensive regular-expression based search functionality. If you have a general idea of what you are looking for you can search for it via 'search '. In the output below, a search is being made for MS Bulletin MS09-011. The search function will locate this string within the module names, descriptions, references, etc.

Note the naming convention for Metasploit modules uses underscores versus hyphens.

```
msf > search usermap_script

Matching Modules
================

   Name                                Disclosure Date  Rank
Description
   ----                                ---------------  ----
-----------
   exploit/multi/samba/usermap_script  2007-05-14       excellent  Samba
"username map script" Command Execution

msf >
```

help

You can further refine your searches by using the built-in keyword system.

```
msf > help search
Usage: search [keywords]

Keywords:
  name      :  Modules with a matching descriptive name
  path      :  Modules with a matching path or reference name
  platform  :  Modules affecting this platform
  type      :  Modules of a specific type (exploit, auxiliary, or post)
  app       :  Modules that are client or server attacks
  author    :  Modules written by this author
  cve       :  Modules with a matching CVE ID
  bid       :  Modules with a matching Bugtraq ID
  osvdb     :  Modules with a matching OSVDB ID

Examples:
  search cve:2009 type:exploit app:client
```

```
msf >
```

name

To search using a descriptive name, use the “name” keyword.

```
msf > search name:mysql

Matching Modules
================

   Name                                             Disclosure Date
Rank       Description
   ----                                             ---------------
----       -----------
   auxiliary/admin/mysql/mysql_enum
normal     MySQL Enumeration Module
   auxiliary/admin/mysql/mysql_sql
normal     MySQL SQL Generic Query
   auxiliary/analyze/jtr_mysql_fast
normal     John the Ripper MySQL Password Cracker (Fast Mode)
   auxiliary/scanner/mysql/mysql_authbypass_hashdump  2012-06-09
normal     MySQL Authentication Bypass Password Dump
   auxiliary/scanner/mysql/mysql_hashdump
normal     MYSQL Password Hashdump
   auxiliary/scanner/mysql/mysql_login
normal     MySQL Login Utility
   auxiliary/scanner/mysql/mysql_schemadump
normal     MYSQL Schema Dump
   auxiliary/scanner/mysql/mysql_version
normal     MySQL Server Version Enumeration
   exploit/linux/mysql/mysql_yassl_getname            2010-01-25
good       MySQL yaSSL CertDecoder::GetName Buffer Overflow
   exploit/linux/mysql/mysql_yassl_hello              2008-01-04
good       MySQL yaSSL SSL Hello Message Buffer Overflow
   exploit/windows/mysql/mysql_payload                2009-01-16
excellent  Oracle MySQL for Microsoft Windows Payload Execution
   exploit/windows/mysql/mysql_yassl_hello            2008-01-04
average    MySQL yaSSL SSL Hello Message Buffer Overflow
msf >
```

path

Use the “path” keyword to search within the module paths.

```
msf > search path:scada

Matching Modules
================

   Name                                                Disclosure Date
Rank     Description
   ----                                                ---------------
----     -----------
   auxiliary/admin/scada/igss_exec_17                  2011-03-21
normal   Interactive Graphical SCADA System Remote Command Injection
   exploit/windows/scada/citect_scada_odbc             2008-06-11
normal   CitectSCADA/CitectFacilities ODBC Buffer Overflow
...snip...
```

platform

You can use “platform” to narrow down your search to modules that affect a specific platform.

```
msf > search platform:aix

Matching Modules
================

   Name                                  Disclosure Date  Rank
Description
   ----                                  ---------------  ----
-----------
   payload/aix/ppc/shell_bind_tcp                         normal  AIX
Command Shell, Bind TCP Inline
   payload/aix/ppc/shell_find_port                        normal  AIX
Command Shell, Find Port Inline
   payload/aix/ppc/shell_interact                         normal  AIX
execve shell for inetd
...snip...
```

type

Using the “type” lets you filter by module type such as auxiliary, post, exploit, etc.

```
msf > search type:post

Matching Modules
================

   Name                                                Disclosure Date
Rank    Description
   ----                                                ---------------
----    -----------
   post/linux/gather/checkvm
normal  Linux Gather Virtual Environment Detection
   post/linux/gather/enum_cron
normal  Linux Cron Job Enumeration
   post/linux/gather/enum_linux
normal  Linux Gather System Information
...snip...
```

author

Searching with the “author” keyword lets you search for modules by your favorite author.

```
msf > search author:dookie

Matching Modules
================

   Name                                                       Disclosure
Date  Rank     Description
   ----
---------------  ----     -----------
   exploit/osx/http/evocam_webserver                          2010-06-01
average  MacOS X EvoCam HTTP GET Buffer Overflow
   exploit/osx/misc/ufo_ai                                    2009-10-28
average  UFO: Alien Invasion IRC Client Buffer Overflow Exploit
   exploit/windows/browser/amaya_bdo                          2009-01-28
normal   Amaya Browser v11.0 bdo tag overflow
...snip...
```

multiple

You can also combine multiple keywords together to further narrow down the returned results.

```
msf > search cve:2011 author:jduck platform:linux

Matching Modules
================

   Name                                          Disclosure Date  Rank
Description
   ----                                          ---------------  ----
-----------
   exploit/linux/misc/netsupport_manager_agent  2011-01-08       average
NetSupport Manager Agent Remote Buffer Overflow
```

sessions

The ‘sessions’ command allows you to list, interact with, and kill spawned sessions. The sessions can be shells, Meterpreter sessions, VNC, etc.

```
    msf > sessions -h
    Usage: sessions [options]

    Active session manipulation and interaction.

    OPTIONS:

        -K        Terminate all sessions
        -c <opt>  Run a command on the session given with -i, or all
        -d <opt>  Detach an interactive session
        -h        Help banner
        -i <opt>  Interact with the supplied session ID
        -k <opt>  Terminate session
        -l        List all active sessions
        -q        Quiet mode
        -r        Reset the ring buffer for the session given with -i,
or all
        -s <opt>  Run a script on the session given with -i, or all
        -u <opt>  Upgrade a win32 shell to a meterpreter session
        -v        List verbose fields
```

To list any active sessions, pass the ‘-l’ options to ‘sessions’.

```
msf exploit(3proxy) > sessions -l

Active sessions
===============

  Id  Description    Tunnel
  --  -----------    ------
  1   Command shell  192.168.1.101:33191 -> 192.168.1.104:4444
```

To interact with a given session, you just need to use the ‘-i’ switch followed by the Id number of the session.

```
msf exploit(3proxy) > sessions -i 1
[*] Starting interaction with 1...

C:\WINDOWS\system32>
```

set

The ‘set’ command allows you to configure Framework options and parameters for the current module you are working with.

```
msf auxiliary(ms09_050_smb2_negotiate_func_index) > set RHOST
172.16.194.134
RHOST => 172.16.194.134
msf auxiliary(ms09_050_smb2_negotiate_func_index) > show options

Module options (exploit/windows/smb/ms09_050_smb2_negotiate_func_index):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  172.16.194.134   yes       The target address
   RPORT  445              yes       The target port
   WAIT   180              yes       The number of seconds to wait for
the attack to complete.

Exploit target:

   Id  Name
   --  ----
   0   Windows Vista SP1/SP2 and Server 2008 (x86)
```

Metasploit also allows you the ability to set an encoder to use at run-time. This is particularly useful in exploit development when you aren't quite certain as to which payload encoding methods will work with an exploit.

```
msf  exploit(ms09_050_smb2_negotiate_func_index) > show encoders

Compatible Encoders
===================

   Name                    Disclosure Date  Rank       Description
   ----                    ---------------  ----       -----------
   generic/none                             normal     The "none"
Encoder
   x86/alpha_mixed                          low        Alpha2
Alphanumeric Mixedcase Encoder
   x86/alpha_upper                          low        Alpha2
Alphanumeric Uppercase Encoder
   x86/avoid_utf8_tolower                   manual     Avoid UTF8/
tolower
   x86/call4_dword_xor                      normal     Call+4 Dword XOR
Encoder
   x86/context_cpuid                        manual     CPUID-based
Context Keyed Payload Encoder
   x86/context_stat                         manual     stat(2)-based
Context Keyed Payload Encoder
   x86/context_time                         manual     time(2)-based
Context Keyed Payload Encoder
   x86/countdown                            normal     Single-byte XOR
Countdown Encoder
   x86/fnstenv_mov                          normal     Variable-length
Fnstenv/mov Dword XOR Encoder
   x86/jmp_call_additive                    normal     Jump/Call XOR
Additive Feedback Encoder
   x86/nonalpha                             low        Non-Alpha Encoder
   x86/nonupper                             low        Non-Upper Encoder
   x86/shikata_ga_nai                       excellent  Polymorphic XOR
Additive Feedback Encoder
   x86/single_static_bit                    manual     Single Static Bit
   x86/unicode_mixed                        manual     Alpha2
Alphanumeric Unicode Mixedcase Encoder
   x86/unicode_upper                        manual     Alpha2
Alphanumeric Unicode Uppercase Encoder
```

unset

The opposite of the ‘set’ command, of course, is ‘unset’. ‘Unset’ removes a parameter previously configured with ‘set’. You can remove all assigned variables with ‘unset all’.

```
msf > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf > set THREADS 50
THREADS => 50
msf > set

Global
======

  Name     Value
  ----     -----
  RHOSTS   192.168.1.0/24
  THREADS  50

msf > unset THREADS
Unsetting THREADS...
msf > unset all
Flushing datastore...
msf > set

Global
======

No entries in data store.

msf >
```

setg

In order to save a lot of typing during a pentest, you can set global variables within msfconsole. You can do this with the ‘setg’ command. Once these have been set, you can use them in as many exploits and auxiliary modules as you like. You can also save them for use the next time your start msfconsole. However, the pitfall is forgetting you have saved globals, so always check your options before you run or exploit. Conversely, you can use the unsetg command to unset a global variable. In the examples that follow, variables are entered in all-caps (ie: LHOST), but Metasploit is case-insensitive so it is not necessary to do so.

```
msf > setg LHOST 192.168.1.101
LHOST => 192.168.1.101
```

```
msf > setg RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf > setg RHOST 192.168.1.136
RHOST => 192.168.1.136
```

After setting your different variables, you can run the 'save' command to save your current environment and settings. With your settings saved, they will be automatically loaded on startup which saves you from having to set everything again.

```
msf > save
Saved configuration to: /root/.msf4/config
msf >
```

show

Entering 'show' at the msfconsole prompt will display every module within Metasploit.

```
msf > show

Encoders
========

   Name                    Disclosure Date  Rank       Description
   ----                    ---------------  ----       -----------
   cmd/generic_sh                           good       Generic Shell
Variable Substitution Command Encoder
   cmd/ifs                                  low        Generic ${IFS}
Substitution Command Encoder
   cmd/printf_php_mq                        manual     printf(1) via PHP
magic_quotes Utility Command Encoder
...snip...
```

There are a number of 'show' commands you can use but the ones you will use most frequently are 'show auxiliary', 'show exploits', 'show payloads', 'show encoders', and 'show nops'.

auxiliary

Executing 'show auxiliary' will display a listing of all of the available auxiliary modules within Metasploit. As mentioned earlier, auxiliary modules include scanners, denial of service modules, fuzzers, and more.

```
msf > show auxiliary
Auxiliary
=========

   Name                                                  Disclosure Date
Rank    Description
   ----                                                  ---------------
----    -----------
   admin/2wire/xslt_password_reset                       2007-08-15
normal  2Wire Cross-Site Request Forgery Password Reset Vulnerability
   admin/backupexec/dump
normal  Veritas Backup Exec Windows Remote File Access
   admin/backupexec/registry
normal  Veritas Backup Exec Server Registry Access
...snip...
```

exploits

Naturally, ‘show exploits’ will be the command you are most interested in running since at its core, Metasploit is all about exploitation. Run ‘show exploits’ to get a listing of all exploits contained in the framework.

```
msf > show exploits

Exploits
========

   Name
Disclosure Date  Rank       Description
   ----
---------------  ----       -----------
   aix/rpc_cmsd_opcode21
2009-10-07       great      AIX Calendar Manager Service Daemon
(rpc.cmsd) Opcode 21 Buffer Overflow
   aix/rpc_ttdbserverd_realpath
2009-06-17       great      ToolTalk rpc.ttdbserverd
_tt_internal_realpath Buffer Overflow (AIX)
   bsdi/softcart/mercantec_softcart
2004-08-19       great      Mercantec SoftCart CGI Overflow
...snip...
```

Using msfconsole payloads

Running ‘show payloads’ will display all of the different payloads for all platforms available within Metasploit.

```
msf > show payloads

Payloads
========

   Name                                             Disclosure Date
Rank    Description
   ----                                             ---------------
----    -----------
   aix/ppc/shell_bind_tcp
normal  AIX Command Shell, Bind TCP Inline
   aix/ppc/shell_find_port
normal  AIX Command Shell, Find Port Inline
   aix/ppc/shell_interact
normal  AIX execve shell for inetd
...snip...
```

payloads

As you can see, there are a lot of payloads available. Fortunately, when you are in the context of a particular exploit, running ‘show payloads’ will only display the payloads that are compatible with that particular exploit. For instance, if it is a Windows exploit, you will not be shown the Linux payloads.

```
msf  exploit(ms08_067_netapi) > show payloads

Compatible Payloads
===================

   Name                                             Disclosure Date
Rank    Description
   ----                                             ---------------
----    -----------
   generic/custom
normal  Custom Payload
   generic/debug_trap
normal  Generic x86 Debug Trap
   generic/shell_bind_tcp
normal  Generic Command Shell, Bind TCP Inline
...snip...
```

options

If you have selected a specific module, you can issue the 'show options' command to display which settings are available and/or required for that specific module.

```
msf exploit(ms08_067_netapi) > show options

Module options:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER,
SRVSVC)

Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting
```

targets

If you aren't certain whether an operating system is vulnerable to a particular exploit, run the 'show targets' command from within the context of an exploit module to see which targets are supported.

```
msf  exploit(ms08_067_netapi) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic Targeting
   1   Windows 2000 Universal
   10  Windows 2003 SP1 Japanese (NO NX)
   11  Windows 2003 SP2 English (NO NX)
   12  Windows 2003 SP2 English (NX)
...snip...
```

advanced

If you wish the further fine-tune an exploit, you can see more advanced options by running 'show advanced'.

```
msf exploit(ms08_067_netapi) > show advanced

Module advanced options:

   Name           : CHOST
   Current Setting:
   Description    : The local client address

   Name           : CPORT
   Current Setting:
   Description    : The local client port

...snip...
```

encoders

Running 'show encoders' will display a listing of the encoders that are available within MSF.

```
msf > show encoders
Compatible Encoders
===================

   Name                    Disclosure Date  Rank       Description
   ----                    ---------------  ----       -----------
   cmd/generic_sh                           good       Generic Shell
Variable Substitution Command Encoder
   cmd/ifs                                  low        Generic ${IFS}
Substitution Command Encoder
   cmd/printf_php_mq                        manual     printf(1) via PHP
magic_quotes Utility Command Encoder
   generic/none                             normal     The "none"
Encoder
   mipsbe/longxor                           normal     XOR Encoder
   mipsle/longxor                           normal     XOR Encoder
   php/base64                               great      PHP Base64
encoder
   ppc/longxor                              normal     PPC LongXOR
Encoder
```

```
   ppc/longxor_tag                            normal     PPC LongXOR
Encoder
   sparc/longxor_tag                          normal     SPARC DWORD XOR
Encoder
   x64/xor                                    normal     XOR Encoder
   x86/alpha_mixed                            low        Alpha2
Alphanumeric Mixedcase Encoder
   x86/alpha_upper                            low        Alpha2
Alphanumeric Uppercase Encoder
   x86/avoid_utf8_tolower                     manual     Avoid UTF8/
tolower
   x86/call4_dword_xor                        normal     Call+4 Dword XOR
Encoder
   x86/context_cpuid                          manual     CPUID-based
Context Keyed Payload Encoder
   x86/context_stat                           manual     stat(2)-based
Context Keyed Payload Encoder
   x86/context_time                           manual     time(2)-based
Context Keyed Payload Encoder
   x86/countdown                              normal     Single-byte XOR
Countdown Encoder
   x86/fnstenv_mov                            normal     Variable-length
Fnstenv/mov Dword XOR Encoder
   x86/jmp_call_additive                      normal     Jump/Call XOR
Additive Feedback Encoder
   x86/nonalpha                               low        Non-Alpha Encoder
   x86/nonupper                               low        Non-Upper Encoder
   x86/shikata_ga_nai                         excellent  Polymorphic XOR
Additive Feedback Encoder
   x86/single_static_bit                      manual     Single Static Bit
   x86/unicode_mixed                          manual     Alpha2
Alphanumeric Unicode Mixedcase Encoder
   x86/unicode_upper                          manual     Alpha2
Alphanumeric Unicode Uppercase Encoder
```

nops

Lastly, issuing the 'show nops' command will display the NOP Generators that Metasploit has to offer.

```
msf > show nops
NOP Generators
==============

   Name             Disclosure Date  Rank    Description
   ----              ---------------  ----    -----------
   armle/simple                       normal  Simple
   php/generic                        normal  PHP Nop Generator
   ppc/simple                         normal  Simple
   sparc/random                       normal  SPARC NOP generator
   tty/generic                        normal  TTY Nop Generator
   x64/simple                         normal  Simple
   x86/opty2                          normal  Opty2
   x86/single_byte                    normal  Single Byte
```

use

When you have decided on a particular module to make use of, issue the ‘use’ command to select it. The ‘use’ command changes your context to a specific module, exposing type-specific commands. Notice in the output below that any global variables that were previously set are already configured.

```
msf > use dos/windows/smb/ms09_001_write
msf auxiliary(ms09_001_write) > show options

Module options:

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  445              yes       Set the SMB service port

msf auxiliary(ms09_001_write) >
```

At any time you need assistance you can use the msfconsole help command to display available options.