Automating Post Exploitation with Metasploit

Metasploit Primer

Disclaimer

The author of this class is not responsible for the use of this information. It is provided here is for the use of security professionals to automate post exploitation tasks while performing authorized security assessments and tasks. Not all API calls available will be covered during this class, only those that are considered to be the most useful ones based on experience. The Metasploit Framework is in constant evolution, this course covers the current version of the framework at the time of it's delivery.

Metasploit History

- First Released in Perl on October 2003 with 11 Exploits
- Version 2.0 released on April 2004 with the help of another researcher known as Spoon
- Mat Miller (Skape) joins the project & is the original author of Meterpreter
- Version 3.0 was released on March 2007 completely re-written in Ruby

Metasploit History

- October 2009 Rapid7 acquires the Metasploit Project
- April 2010 the first commercial version released: Metasploit Express
- November 2010 the second commercial version of Metasploit released: Metasploit Pro
- August 2011 Metasploit 4.0 released

Terminology

- Exploit - means by which attacker takes advantage of a software flaw or misconfiguration to produce unintended results
- Payload - software attacker wishes to run on a target system.
- Shellcode - a small piece of code used as the payload in the exploitation of a vulnerability, mainly written in assembly

Terminology

- Module - Piece of software inside the framework that performs a specific task. The Types of modules are:
  - Exploit Module - Exploits flaw or mis-configuration on target system
  - Auxiliary Module - Performs non exploit tasks like discovery, scanning and fuzzing among others
  - Post Module - Performs Post Exploitation tasks thru a given session on a target host

Terminology

- Listener - Component that listens or initiates the connection of a shell created by a payload

Interfaces

- Interfaces to the Framework
  - –msfconsole - Console interface to the framework, provides the greatest flexibility
  - –msfcli - Single execution of modules for one shot use
  - –msfgui - Java based GUI written by ScriptJunkie (Matt Weeks)
  - –armitage - Java based GUI Written by Raphael Mudge

Utilities

- Framework Utilities
  - msfvenon - Replacement for msfpayload and msfencode for generating payloads in different formats and encoding them
  - msfbinscan - replacement for msfelfscan, msfmachscan and msfpescan to aid in exploit development

msfconsole

- Interface we will use thru the class.
- The commands are broken in to several parts
  - Core Commands
  - Database Commands
  - Post Commands
  - Auxiliary Commands
  - Exploit Commands
- The commands available are dependent on the shell. Shell is context specific to post, auxiliary or exploit.

msfconsole

- The prompt will change depending on the module select, nop or payload selected

```
msf>
msf  post(name) >
msf  auxiliary(name) >
msf  exploit(name) >
msf  nop(name) >
msf  payload(name) >
```

- Console itself has several options

```
msf > show options

Global Options:
===============

   Option           Current Setting  Description
   ------           ---------------  -----------
   ConsoleLogging                    Log all console input and output
   LogLevel                          Verbosity of logs (default 0, max 5)
   MinimumRank                       The minimum rank of exploits that will run wit......
   SessionLogging                    Log all input and output for sessions
   TimestampOutput                   Prefix all console output with a timestamp
```

msfconsole

- System commands can be executed from the msfconsole and output will be shown.

```
msf > ping -c 2 192.168.1.1
[*] exec: ping -c 2 192.168.1.1

PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=0.310 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.376 ms

--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.310/0.343/0.376/0.033 ms
```

- The shell can be cleared with the clear command or with Crtl-L
- No output redirection is supported for console commands.

msfconsole - Core Commands

```
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
    exit          Exit the console
    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
    quit          Exit the console
    reload_all    Reloads all modules from all defined module paths
    resource      Run the commands stored in a file
    route         Route traffic through a session
    save          Saves the active datastores
```

msfconsole - Core Commands

```
search          Searches module names and descriptions
sessions        Dump session listings and display information about sessions
set             Sets a 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 variables
unsetg          Unsets one or more global variables
use             Selects a module by name
version         Show the framework and console library version numbers
```

- To get **`help`** on each of the commands one can either use the help command or execute command with **`-h`** option

```
msf > help reload_all
Usage: reload_all

Reload all modules from all configured module paths.  This may take a while.
See also: loadpath
```

msfconsole - Search

- The search command is used to find modules **`search <regex>`**
- The **`-t`** option specifies the type of module (**`-t all|auxiliary|encoder|exploit|nop|payload`**)
- The **`-r`** option specifies a minimum safety rank (**`-r min_rank`**)
- Ranking definitions:
  - Excellent: Will never crash a service
  - Great: Has default target, detects target, or uses app-specific return address
  - Good: Has default target and is the common type of selected target software
  - Normal: Reliable, but no 'default target' or auto-detect of target
  - Average: Generally unreliable or difficult to exploit
  - Low: Nearly impossible to exploit (under 50%)
  - Manual: Unstable, difficult to exploit, and is essentially denial of service

msfconsole

- msfconsole shell supports tab auto complete so you can complete using tab for unique command names, payloads, exploits, encoders, options etc.
- You can hit tab twice to see all available options

```
msf > use post/windows/manage/<tab><tab>
use post/windows/manage/autoroute                    use post/windows/manage/migrate      ....
```

- As modules, payloads, encoders, plugin and nops are selected the available commands change.
- To move back in context use the `back` command

msfconsole - Plugins

- Plugins expand the console functionality and in most cases adds new commands

```
msf > load editor
[*] Successfully loaded plugin: editor
msf > ?

Editor Commands
===============

Command       Description
-------       -----------
edit          A handy editor commmand
......
```

- To unload a plugin the command is unload and it's name

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

msfconsole - Plugins

- To get a list of plugins available for use from all default paths do a load <tab> <tab>

```
msf > load
load auto_add_route  load ips_filter      load openvas         load sounds
load db_credcollect  load lab             load pcap_log        load thread
load db_tracker      load msfd            load token_hunter    load editor
load msgrpc          load sample          load wmap
load event_tester    load nessus          load session_tagger  load xmlrpc
load ffautoregen     load nexpose         load socket_logger
```

msfconsole - Logging

- All logging information as well as a users own copy of plugins, modules, scripts and configuration scripts are saved to the `.msf4` directory under the users home folder
- For troubleshooting the log in `.msf4/logs/framework.log` will show information on errors when running modules and loading that can be useful to determine problems
- When session logging is enabled by `set SessionLogging true` session logs are saved to `~/.msf4/logs/sessions/<timestamp>_<targetIP>_[meterpreter|shell].log`

msfconsole - Logging

- To add a timestamp to the screen output `set TimestampOutput true`
- To log everything entered and shown in msfconsole `set ConsoleLogging true`
- Logs are stored in the user's home directory in `~/.msf4/logs/console.log`
- The `spool` command can be used to log all output also and it can be issued from any context

```
msf > spool /tmp/test.log
[*] Spooling to file /tmp/test.log...
```

msfconsole - Logging

- Remember to clear all logs after a pentest!
- It is recommended to set timestamp for output and use the `spool` command for logging specifying a secure place for the log and a useful name for later reporting
- For some modules, scripts and plugins the use of `spool` or setting `ConsoleLogging` are the only ways to capture their output for later reporting

msfconsole

- To select a module, payload, encoder or nop the command is `use`

```
msf > use post/windows/manage/multi_meterpreter_inject
msf  post(multi_meterpreter_inject) >
```

- To get information on a module the `info` command is used

```
msf  post(multi_meterpreter_inject) > info
       Name: Windows Manage Inject in Memory Multiple Payloads
     Module: post/windows/manage/multi_meterpreter_inject
    Version: 13341
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  Carlos Perez <carlos_perez@darkoperator.com>
.........
```

msfconsole - Commands

- As each module type is selected new variables, options and commands are added, to see them use `?` command
- Use `show` to see other parameters

```
Exploit Commands
================

    Command       Description
    -------       -----------
    check         Check to see if a target is vulnerable
    exploit       Launch an exploit attempt
    pry           Open a Pry session on the current module
    rcheck        Reloads the module and checks if the target is vulnerable
    reload        Just reloads the module
    rexploit      Reloads the module and launches an exploit attempt
```

msfconsole - Commands

```
Post Commands
=============

    Command       Description
    -------       -----------
    exploit       This is an alias for the run command
    pry           Open a Pry session on the current module
    reload        Reload the current module from disk
    rerun         Reloads and launches the module
    rexploit      This is an alias for the rerun command
    run           Launches the post exploitation module


Auxiliary Commands
==================

    Command       Description
    -------       -----------
    exploit       This is an alias for the run command
    pry           Open a Pry session on the current module
    reload        Reloads the auxiliary module
    rerun         Reloads and launches the auxiliary module
    rexploit      This is an alias for the rerun command
    run           Launches the auxiliary module
```

msfconsole - Module Options

- To see the options available for the selected module `show options` is used

```
msf  post(multi_meterpreter_inject) > show options

Module options (post/windows/manage/multi_meterpreter_inject):

   Name     Current Setting                     Required  Description
   ----     ---------------                     --------  -----------
   HANDLER  false                               no        Start new multi/handler job on local box.
   IPLIST   192.168.1.241                       yes       List of semicolom separated IP list.
   LPORT    4444                                no        Port number for the payload LPORT variable.
   PAYLOAD  windows/meterpreter/reverse_tcp     no        Payload to inject in to process memory
   PIDLIST                                      no        List of semicolom separated PID list.
   SESSION                                      yes       The session to run this module on.
```

msfconsole - Module Options

- To see the additional options available for the selected module `show advanced options` is used

```
msf  post(multi_meterpreter_inject) > show advanced

Module advanced options:

   Name           : PROCESSNAME
   Current Setting: notepad.exe
   Description    : Name of process to create if no PID available

   Name           : VERBOSE
   Current Setting: false
   Description    : Enable detailed status messages

   Name           : WORKSPACE
   Current Setting:
   Description    : Specify the workspace for this module
```

msfconsole - Module Options

- Options are case sensitive.
- Use tab completion when setting an option since the shell will not check if the name is correct or not until execution
- To set a option the `set <option> <value>` is used
- To list all variables that have been set the command `set <return>` or `show options`

msfconsole - Module Options

- To see a particular option value **`set <option>`**
- To remove a value or several values **`unset <opt1> <opt2>`**
- To remove all values **`unset all`**

```
msf  post(multi_meterpreter_inject) > set SESSION 1
SESSION => 1
msf  post(multi_meterpreter_inject) > set SESSION
SESSION => 1
msf  post(multi_meterpreter_inject) > unset SESSION
Unsetting SESSION...
```

msfconsole - Module Options

- Common Options
  - PAYLOAD: The payload we want to deliver via an exploit module
  - RHOST: The machine to deliver an exploit or interact with
  - RHOSTS: Range of hosts, file with hosts or CIDR notation of hosts to communicate with
  - RPORT: The remote TCP port (typically where an exploitable service is listening)
  - LPORT: The local TCP port for the payload (make sure the target machine can reach this port)
  - TARGET: The target type for an exploit module
  - SRVHOST: The address to listen for clients to connect to

msfconsole - Global Options

- Options set outside the context of a module, payload, encoder or nop (in `msf>`) are set globally
- Options set using the **`set`** command while in the context of a module, payload, encoder or nop are only to that specific instance of the module
- To set a global variable from inside the context of a module the **`setg`** command is used

msfconsole - Global Options

- Options that have been set can be saved in the case you need to close msfconsole or will use those settings with each start of the framework

- ```
  msf  exploit(handler) > save
  Saved configuration to: /Users/carlos/.msf4/config
  msf  exploit(handler) > cat /Users/carlos/.msf4/config
  [*] exec: cat /Users/carlos/.msf4/config

  [framework/core]

  [framework/ui/console]
  ActiveModule=exploit/multi/handler

  [multi/handler]
  VERBOSE=false
  WfsDelay=0
  EnableContextEncoding=false
  DisablePayloadHandler=false
  ExitOnSession=true
  ListenerTimeout=0
  PAYLOAD=windows/meterpreter/reverse_tcp
  LHOST=192.168.1.100
  LPORT=443
  ```
  ```

msfconsole - Reload

- When working on a module and a change is made the **`reload`** command should be use to apply the changes for the selected module
- If a new module is created and msfconsole is up the **`reload_all`** command would be used to load from all default paths

```
msf  post(multi_meterpreter_inject) > reload
[*] Reloading module...
msf  post(multi_meterpreter_inject) > reload_all
[*] Reloading modules from all module paths...
```

msfconsole - Running Modules

- Execution: modules can be executed in the background with the **`-j`** option and the values of the options can be specified with a comma separated list in **`VAR=VAL`** format with the **`-o`** option
- Do use the **`-h`** option on the run or exploit command depending on the module selected for other options at the time of executing

msfconsole - Running Modules

- Tips for when running modules
  - For exploit modules try to run them as jobs with the `-j` options and the `-z` flag to specify no interaction with the session
  - For exploits or auxiliary modules that will listen on non ephemeral ports run msfconsole as root
  - If the exploit will start a web listener for client side set the port to 80 or 443 and set SSL to true to reduce detection by IPS
  - if the ENCODE option is available test several encoders in lab for the most effective one for the exploit being used

msfconsole - Managing Sessions

- msfconsole can handle multiple sessions of different types at once
- To manage sessions the **`sessions`** command is used
- 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

msfconsole - Managing Sessions

- For listing current sessions and getting information like ID, Type, Information that was gathered and IP Connection info the **-l** Option is used

```
msf  exploit(handler) > sessions -l

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

  Id  Type                   Information                    Connection
  --  ----                   -----------                    ----------
  1   meterpreter x86/win32  CARLOS-192FCD91\Administr..... 192.168.1.100:4444 -> 192.168.1.115:1767
  2   shell linux                                           192.168.1.100:4448 -> 192.168.1.117:55327
  3   shell linux                                           192.168.1.100:4448 -> 192.168.1.119:53401
  4   shell windows                                         192.168.1.100:3333 -> 192.168.1.115:1769
  5   shell windows          Microsoft Windows XP [Ver..... 192.168.1.100:3333 -> 192.168.1.115:1770
```

- The **-v** option can be used to see even more information like the exploit that generated the session

msfconsole - Managing Sessions

- To interact with a session the command is **`sessions -i <ID>`**
- You can run a shell command against a given session or all sessions with the with the **`-c <command>`** options
- You can run a given Meterpreter Script against a session or all sessions with the **`-s <script>`** options
- You can upgrade a Windows shell session with the **`-u`**

Payloads

Payloads - Type

- There are typically 2 types of payloads: the single and stager
- Single tend to execute actions on target system, are self-contained, and are simple in function
- Stager will load a stager on the target system. This will load stages thru the network. The stager being small & simple and the stager providing the complex functionality

Payloads - Types

- There are 3 main types of Payloads shells that we will cover
  - Regular Shell
    - Reverse, Bind
  - Terminal Shell
    - SSH, Telnet
  - Meterpreter
    - Windows, Java*, PHP* and POSIX(Experimental)
- They can run over different types of channels
  - TCP, UDP, HTTP, HTTPS
- Not all channel types are supported by all the payloads

* Have more in common with a regular shell than the Original Meterpreter

Payloads - Shell

- Channel is not Encrypted
- Depend on a shell interpreter on the target
  - /bin/bash, /bin/sh, cmd.exe
- You are limited by the commands on the target box and the tools you can upload to the target
- Detectable by most IDS/IPS
- Single threaded
- Forensically noisy
- Certain apps expecting a Terminal/TTY can break a shell (vi, vipw, wmic)

Payloads -Terminal Shell

- These are the sessions obtained thru SSH and Telnet Auxiliary Modules
- SSH is encrypted and Telnet clear text
- Can be used with other tools and standalone
- Not threaded
- Forensically noisy
- Can be confused with regular management traffic

Meterpreter

- Not all Meterpreter Payloads are the same
  - None Windows versions rely on system commands for certain data
  - Not all commands shown in the Meterpreter Session UI are supported
- Encrypted channel
- Flexibility in channel types (TCP, UDP, HTTP, HTTPS)
- Leverages Libraries and API's on target box
- Threaded

Meterpreter - Advantages

- Supports all versions of Windows above Windows 2000
- Supports x86 and X64 systems with Extensions compiled for each
- Runs in memory
- Development history
  - Written by Skape for Metasploit 2.x
  - Common extensions merged for 3.x
  - Sniffer extension added in 3.3
  - Railgun Extension 3.4
  - Search/Webcam/Audio from 3.5 to 3.7

Meterpreter - Advantage

- Advanced dynamically extensible payload
  - –Uses in-memory Reflective DLL injection for stagers
  - –Uses memset's, which zero out the packet data before the memory is freed
  - –Encrypted channel using TLS (Self-Signed)
  - –Extended at runtime over the network
  - –Communicates over stager socket
  - –Comprehensive client-side Ruby API

Meterpreter - Advantages

- Communication channel is encrypted
- Scrubs freed memory
- Does not touch the File System for most uses
- Uses native Windows API with Core libraries and extensions
- Can load third party or Windows DLL's local on the box
- Uses Reflective DLL Injection for Extensions, Railgun DLLs are no
- Has access to process memory space

Meterpreter - How it Works

- The target executes the initial stager
- The stager loads the middle stage
- The middle stage loads the DLL injector
- Patches the Windows API for in-memory DLL injection
- The DLL injector loads the Meterpreter core
- The Meterpreter loads extensions
  - Always loads stdapi, sometimes loads priv if the privilege allows it
- Runs AutoRunScript if one defined on the Handler

Meterpreter - Extensions

- Meterpreter has several extension:
  - stdapi - adds commands for interacting with the file system, networking commands, control over user interface, basic system commands and control over the microphone and webcamera on a system
  - priv - adds commands for privilege escalation, dumping hash database and command to modify file MACE attributes
  - espia - Adds commands for capturing sound and pictures from webcam
  - incognito - Adds commands for manipulation of authentication tokens
  - sniffer - Adds commands for sniffing traffic on a target host (Broken at the time of this writing)

Meterpreter - Shell

- Meterpreter provides it's own shell, just like msfconsole it provides a history of commands, tab completion and you can clear the screen with Ctrl-L
- To see a list of command you can use ? or help
- All of the commands shown use the Win32 API to execute the calls from memory via the libraries loaded without writing to disk

Meterpreter - Shell

- To help facilitate the automation of post exploitation Meterpreter can run Resource files, Meterpreter Scripts leveraging the API and the Post Modules
- Meterpreter can executes scripts located in the following paths
  - `<msfroot>/scripts/meterpreter`
  - `~/.msf4/scripts/meterpreter`
- Meterpreter can execute post modules located in the following paths
  - `<msfroot>/modules/post`
  - `~/.msf4/modules/post`
- Meterpreter will not let you run post modules that are not specified to run in Meterpreter

Meterpreter - Help

- Some Meterpreter commands take the `-h` option

```
meterpreter > execute -h
Usage: execute -f file [options]

Executes a command on the remote machine.

OPTIONS:

    -H        Create the process hidden from view.
    -a <opt>  The arguments to pass to the command.
    -c        Channelized I/O (required for interaction).
    -d <opt>  The 'dummy' executable to launch when using -m.
    -f <opt>  The executable command to run.
    -h        Help menu.
    -i        Interact with the process after creating it.
    -k        Execute process on the meterpreters current desktop
    -m        Execute from memory.
    -s <opt>  Execute process in a given session as the session user
    -t        Execute process with currently impersonated thread token
```

Great care should be taken when running commands with `-h` since not all support it. Example `clearev, reboot, exit, shutdown ..etc`

Meterpreter - Help

- All Meterpreter scripts take the -h option

- ```
  meterpreter > run scheduleme -h
  Scheduleme -- provides most common scheduling types used during a pentest
  This script can upload a given executable or script and schedule it to be
  executed. All scheduled task are run as System so the Meterpreter process
  must be System or local admin for local schedules and Administrator for
  remote schedules

  OPTIONS:

      -c  <opt>  Command to execute at the given time. If options for execution needed use double quotes
      -d         Daily.
      -e  <opt>  Executable or script to upload to target host, will not work with remote schedule
      -h         Help menu.
      -hr <opt>  Every specified hours 1-23.
      -i         Run command immediately and only once.
      -l         When a user logs on.
      -m  <opt>  Every specified amount of minutes 1-1439
      -o  <opt>  Options for executable when upload method used
      -p         Password for account provided.
      -r         Remote Schedule. Executable has to be already on remote target
      -s         At system startup.
      -t  <opt>  Remote system to schedule job.
      -u         Username of account with administrative privileges.
  ```
  ```

Meterpreter - Help

- For Post Modules the info command will provide the module information and options

- `meterpreter > info post/windows/manage/enable_rdp`

```
       Name: Windows Manage Enable Remote Desktop
     Module: post/windows/manage/enable_rdp
    Version: 13357
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  Carlos Perez <carlos_perez@darkoperator.com>

Description:
  This module enables the Remote Desktop Service (RDP). It provides
  the options to create an account and configure it to be a member of
  the Local Administrators and Remote Desktop Users group. It can also
  forward the target's port 3389/tcp.

Module options (post/windows/manage/enable_rdp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   ENABLE    true             no        Enable the RDP Service and Firewall Exception.
   FORWARD   false            no        Forward remote port 3389 to local Port.
   LPORT     3389             no        Local port to forward remote connection.
   PASSWORD                   no        Password for the user created.
   SESSION                    yes       The session to run this module on.
   USERNAME                   no        The username of the user to create.
```

Meterpreter - IRB

- IRB inside of the meterpreter shell will set the current session object to client

```
meterpreter > irb
[*] Starting IRB shell
[*] The 'client' variable holds the meterpreter client

>>
>>
```

- Sadly as with the msfconsole IRB shell the post mixins are not available

Meterpreter - Core Commands

- `use` and `load` invoke Meterpreter extensions
- `exit` and `quit` will terminate the current Meterpreter session
- `background` will bring you back to the msfconsole shell without killing the current session
- `resource` will run a given Meterpreter shell resource file

Meterpreter - Core Commands

- The commands `disable_unicode_encoding` and `enable_unicode_encoding` are used when working on target hosts of languages that use unicode characters

Questions?