SANS

PENETRATION TESTING

BLUEPRINT: BUILDING A BETTER PEN TESTER

High-value penetration testing involves modeling the techniques used by real-world computer attackers to find vulnerabilities, and, under controlled circumstances, to exploit those flaws in a professional, safe manner according to a carefully designed scope and rules of engagement. This process helps to determine business risk and potential impact of attacks, all with the goal of helping the organization improve its security stance.

Here are tips for each phase of penetration testing to help you provide higher business value in your work.

PRE-ENGAGEMENT

Discuss **black-box** versus **crystal/white-box** testing while building your **rules of engagement**, noting that crystal box testing often provides more detailed results, is safer, and delivers better business value.

Make sure you get **written permission** to test any third parties that own or operate target systems (MSSPs, cloud providers, ISPs, shared hosting environments, border routers, DNS servers, etc.)

Discuss with target system personnel the particularly sensitive information they have in their environment (such as PII) and how you can measure access to it without actually downloading it. Consider going after generic sample records planted to demonstrate your access instead of the actual sensitive data.

Keep your skills fresh by setting aside an hour or two per week to participate in **Capture the Flag** competitions, including the ***free*** SANS Holiday Hack Challenge at **www.holidayhackchallenge.com** or the numerous free CtFs at **http://www.amanhardikar.com/mindmaps/Practice.html**

RECONNAISSANCE

Carefully **consider all interactions with third-party servers and searches** to ensure you do not divulge sensitive information about the target or violate a non-disclosure arrangement by using them. You may want to **consider using the TOR network** to obscure your relationship with the target organization.

Look for common office documents posted on target websites by using Google searches for:

```
site:<TargetDomain> ext:doc | ext:docx |
ext:xls | ext:xlsx | ext:pdf
```

Use the **Shodan search engine's "net:" directive** to look for unusual or interesting devices in the target network address ranges. Also, use **unique footer information** (such as a common copyright notice on target web pages) to find additional pages via Shodan using the "html:" directive.

Remember to **check social networking sites** (especially LinkedIn, Facebook, and Twitter) to learn about target personnel and the technologies they use.

Double-check that all IP addresses included in the scope belong to the target organization and aren't a mistake. Use **whois** lookups and **traceroute** to check that the addresses make sense and actually belong to the target organization.

In LinkedIn, **look for long-term IT and InfoSec employees to see which technologies they are familiar with**, including firewalls, development environments, and more.

REPORTING

Don't wait for the end of your penetration test to write the report. Instead, write the report as you test, setting aside time each day to write one to three pages. Not only will you produce a better report, your pen test itself will also be better.

Include screenshots in your report to illustrate findings clearly. **Annotate screenshots** with arrows and circles pointing out the important aspects of the illustration.

To add extra value to your recommendations, **consider including steps an operations person can take to verify that a recommended fix is in place**, such as a command to check for the presence of a patch. For some findings, this can be hard to do, so in those cases recommend that the given issue be retested.

Write for the proper audience in each section:

- The Executive Summary should be for the decision-makers who are allocating resources.
- Findings should be written from a technical perspective, informed by business issues.
- Recommendations should take into account the operations team and their processes.

Use a template to guide a voice conversation to **identify the scope** and **rules of engagement**.

Conduct a daily debriefing call with target system personnel to exchange ideas and lessons learned. If daily is too frequent, consider calls two or three times per week.

VULNERABILITY ANALYSIS

Run a sniffer such as tcpdump while you are scanning a target so you can **continually verify** that your scanner is still running appropriately.

While open ports such as **TCP 445** often indicate a Windows machine, this is not always the case. The target could be a **Samba daemon** or **another SMB-based target**.

Verify discovered vulnerability findings by **researching how to check the issue manually** or through a bash, PowerShell, Nmap Scripting Engine (NSE) script, or other script.

Try to **identify false positives** by running a different tool to corroborate a finding.

Put vulnerabilities that you have identified in the context of how critical the asset is, as this helps you assign priority and assess risk.

Identify targets by IP address (IPv4 and IPv6 if you have it), domain name, and (if you have it) MAC address (especially for compromised client machines using DHCP).

If you are using a **virtual machine** for your attacks, **configure it for bridged networking** to avoid filling up NAT tables and to ensure reverse shell connections can come back to you.

POST-EXPLOITATION

When you gain access to a target machine, don't use it to scan for more targets yet, as that might get you detected prematurely. Instead, plunder it for information about other potential targets based on network activity:

DNS cache (Windows): `c:\> ipconfig /displaydns`
ARP cache: `arp -a`
Established TCP connections: `netstat -na`
Routing table: `netstat -nr`

When you gain access to a target, if a sniffer is installed on the machine (like tcpdump or Wireshark's tshark tool), **run it to look for network traffic** to identify other possible target machines, as well as cleartext protocols containing sensitive or useful information.

Even without root, system, or admin privileges on a target machine, you can still usually perform very useful post-exploitation activities, including getting a list of users, determining installed (and possibly vulnerable) software, and pivoting through the system.

When you get on a Windows box, look for ESTABLISHED TCP connections to ports 445 (SMB) and 3389 (RDP), as these other systems may be excellent systems to pivot to, provided they are in scope:

```
c:\> netstat -na | find "EST" | find ":445"
c:\> netstat -na | find "EST" | find ":3389"
```

While they can be very useful for management demonstrations, **be careful turning on video cameras and capturing audio from compromised target machines**. Conduct that level of invasive access only with written permission, and have it reviewed by your legal team to ensure compliance with local laws.

EXPLOITATION

When creating payloads that evade anti-malware tools, **do NOT submit your sample to online scanning sites** like virustotal.com to check for evasion, as that may defeat your payload as new signature updates are distributed.

Set up a **command** or **script** that **checks the availability of the target service** every few seconds while you are attacking it. That way, if you do crash it, you'll notice quickly and can work with target system personnel to get it restarted.

Build your payloads so that they make a reverse connection back to you, increasing the chance you'll get through a firewall that allows outbound connections.

For your payloads, **use a protocol that is likely allowed outbound from the target environment**, such as HTTPS (with a proxy-aware payload like those available in PowerShell Empire, Metasploit, and the Veil Framework) or DNS (such as the DNScat tool).

To lower the chance of crashing Windows target systems and services, once you gain admin-level credentials and SMB access to them, **use psexec or similar Windows features (WMIC, sc, etc.) to cause them to run code**, instead of a buffer overflow or related exploit.

If your exploit fails, read the output of your exploitation tool carefully to see where it errors out. Also, run a sniffer such as tcpdump to see how far along it gets in making a connection, sending the exploit, and loading the stager and stage. If your stager worked but your stage couldn't be loaded, your anti-virus evasion tactics may be failing.

PASSWORD ATTACKS

Create a **word list fine-tuned to the target organization** based on words from its website.

Create a **word list fine-tuned for users based** on their social networking profiles.

When you successfully crack a password using word-mangling rules, **add that password to your dictionary for further password attacks on that penetration test**. That way, if you encounter the same password in a different hash format, you won't have to wait for word-mangling to re-discover that password.

Remember, passwords can be gathered using a variety of techniques, including **automated guessing, cracking**, **sniffing**, and **keystroke logging**.

For password guessing, always **consider the account lockout policy** and try to avoid it by using **password spraying techniques** (a large number of accounts and targets with a small number of passwords).

As **soon as you get hashes** from targets, **start a password cracker** to try to determine the passwords. Don't let any time go by until you start cracking the hashes you've gotten.

Sometimes you **don't need a password** for authentication because simply using the hash can get the job done, as with **pass-the-hash attacks against Windows and SMB targets**, and with **hashes of passwords stored in cookies** for some websites.

If you have a compatible GPU on your system, **consider using a GPU-based password cracking** tool, such as **Hashcat**, as you'll get 20 to 100 times the performance.

Poster!

SANS PENETRATION TESTING

BLUEPRINT: BUILDING A BETTER PEN TESTER

PENT-PSTR-SANS18-BP-V1

SANS PEN TEST CURRICULUM

Certification	Course	Title	URL	Notes
	SEC460	Enterprise Threat and Vulnerability Assessment	www.sans.org/SEC460	← NEW!
GCIH	SEC504	Hacker Tools, Techniques, Exploits, and Incident Handling	www.sans.org/SEC504	ONDEMAND
GWAPT	SEC542	Web App Penetration Testing and Ethical Hacking	www.sans.org/SEC542	ONDEMAND
	SEC550	Active Defense, Offensive Countermeasures and Cyber Deception	www.sans.org/SEC550	
GPEN	SEC560	Network Penetration Testing and Ethical Hacking	www.sans.org/SEC560	ONDEMAND
	SEC561	Immersive Hands-on Hacking Techniques	www.sans.org/SEC561	
	SEC562	CyberCity Hands-on Kinetic Cyber Range Exercise	www.sans.org/SEC562	← CyberCity is available for private training only.
	SEC564	Red Team Operations and Threat Emulation	www.sans.org/SEC564	
	SEC567	Social Engineering for Penetration Testers	www.sans.org/SEC567	
GPYC	SEC573	Automating Information Security with Python	www.sans.org/SEC573	ONDEMAND
GMOB	SEC575	Mobile Device Security and Ethical Hacking	www.sans.org/SEC575	ONDEMAND
	SEC580	Metasploit Kung Fu for Enterprise Pen Testing	www.sans.org/SEC580	
GAWN	SEC617	Wireless Penetration Testing and Ethical Hacking	www.sans.org/SEC617	
	SEC642	Advanced Web App Penetration Testing, Ethical Hacking, and Exploitation Techniques	www.sans.org/SEC642	ONDEMAND
GXPN	SEC660	Advanced Penetration Testing, Exploit Writing, and Ethical Hacking	www.sans.org/SEC660	ONDEMAND
	SEC760	Advanced Exploit Development for Penetration Testers	www.sans.org/SEC760	

Online Training 4-MONTHS ACCESS to Content → ONDEMAND

GIAC CERTIFICATIONS
www.giac.org

GCIH	Certified Incident Handler
GWAPT	Web Application Penetration Tester
GPEN	Penetration Tester
GPYC	Python Coder
GMOB	Mobile Device Security Analyst
GAWN	Assessing and Auditing Wireless Networks
GXPN	Exploit Researcher & Adv. Penetration Tester

Learn more about SANS PENETRATION TESTING and ETHICAL HACKING courses at
www.sans.org/roadmap

PEN TEST BLOGS, CHEAT SHEETS, DOWNLOADS, RESOURCES:
https://pen-testing.sans.org

RULES OF ENGAGEMENT & SCOPING

Rules of Engagement

- ☐ Penetration testing team contact information
- ☐ Target organization contact information
- ☐ "Daily debriefing" frequency
- ☐ "Daily debriefing" time/location
- ☐ Start date of penetration test
- ☐ End date of penetration test
- ☐ Times when the testing occurs

- ☐ Will test be announced to target personnel?
- ☐ Will target organization shun IP addresses of attack systems?
- ☐ Does target organization's network have automatic shunning capabilities that might disrupt access in unforeseen ways (i.e., create a denial-of-service condition), and if so, what steps will be taken to mitigate the risk?
- ☐ Would the shunning of attack systems conclude the test, and if not, what steps will be taken to continue if systems get shunned and what approval (if any) will be required?
- ☐ What are the IP addresses of penetration testing team's attack systems?
- ☐ Is this a "black box" test?
- ☐ What is the policy regarding viewing data (including potentially sensitive/confidential data) on compromised hosts?
- ☐ Will target personnel observe the testing team?

Scoping

- ☐ What are the target organization's biggest security concerns? *(Examples include disclosure of sensitive information, interruption of production processing, embarrassment due to website defacement, etc.)*
- ☐ What specific hosts, network address ranges, or applications should be tested?
- ☐ What specific hosts, network address ranges, or applications should explicitly NOT be tested?
- ☐ List any third parties that own systems or networks that are in scope as well as which systems they own (written permission must have been obtained in advance by the target organization).
- ☐ Will the test be performed against a live production environment or a test environment?

- ☐ Which of the following testing techniques will the penetration test include:
  - ☐ Ping sweep of network ranges?
  - ☐ Port scan of target hosts?
  - ☐ Vulnerability scan of targets?
  - ☐ Penetration into targets?
  - ☐ Application-level manipulation?
  - ☐ Client-side reverse engineering?
  - ☐ Physical penetration attempts?
  - ☐ Social engineering of people?
  - ☐ Other?

- ☐ Will penetration test include internal network testing?
  - ☐ If so, how will access be obtained?
- ☐ Are client/end-user systems included in scope?
  - ☐ If so, how many client systems will be targeted?
- ☐ Is social engineering allowed?
  - ☐ If so, how may it be used?
- ☐ Are denial-of-service attacks allowed?
  - ☐ Are dangerous checks/exploits allowed?

NMAP

Base Syntax

```
# nmap [ScanType] [Options] {targets}
```

Target Specification

IPv4 address: **192.168.1.1**
IPv6 address: **AABB:CCDD::FF%eth0**
Host name: **www.target.tgt**
IP address range: **192.168.0-255.0-255**
CIDR block: **192.168.0.0/16**
Use file with lists of targets: **-iL <filename>**

Target Ports

No port range specified scans 1,000 most popular ports
-F Scan 100 most popular ports
-p<port1>-<port2> Port range
-p<port1>,<port2>,... Port List
-pU:53,U:110,T20-445 Mix TCP and UDP
-r Scan linearly (do not randomize ports)
--top-ports <n> Scan n most popular ports
-p-65535 Leaving off initial port makes Nmap scan start at port 1
-p0- Leaving off end port makes Nmap scan up to port 65535
-p- Leaving off start and end port makes Nmap scan ports 1-65535

Probing Options

-Pn Don't probe (assume all hosts are up)
-PB Default probe (TCP 80, 445 & ICMP)
-PS<portlist>
Check whether targets are up by probing TCP ports
-PE Use ICMP Echo Request
-PP Use ICMP Timestamp Request
-PM Use ICMP Netmask Request

Scan Types

-sn Probe only (host discovery, not port scan)
-sS SYN Scan
-sT TCP Connect Scan
-sU UDP Scan
-sV Version Scan
-O OS Detection
--scanflags Set custom list of TCP using URGACKPSHRSTSYNFIN in any order

Fine-Grained Timing Options

--min-hostgroup/max-hostgroup <size>
Parallel host scan group sizes

--min-parallelism/max-parallelism <numprobes>
Probe parallelization

--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>
Specifies probe round trip time.

--max-retries <tries>
Caps number of port scan probe retransmissions.

--host-timeout <time>
Give up on target after this long

--scan-delay/--max-scan-delay <time>
Adjust delay between probes

--min-rate <number>
Send packets no slower than <number> per second

--max-rate <number>
Send packets no faster than <number> per second

Aggregate Timing Options

-T0 Paranoid: Very slow, used for IDS evasion
-T1 Sneaky: Quite slow, used for IDS evasion
-T2 Polite: Slows down to consume less bandwidth, runs ~10 times slower than default
-T3 Normal: Default, a dynamic timing model based on target responsiveness
-T4 Aggressive: Assumes a fast and reliable network and may overwhelm targets
-T5 Insane: Very aggressive; will likely overwhelm targets or miss open ports

Scripting Engine

-sC Run default scripts
--script=<ScriptName>|<ScriptCategory>|<ScriptDir>...
Run individual or groups of scripts
--script-args=<Name1=Value1,...>
Use the list of script arguments
--script-updatedb
Update script database

Output Formats

-oN Standard Nmap output
-oG Greppable format
-oX XML format
-oA <basename> Generate Nmap, Greppable, and XML output files using basename for files

Misc Options

-n Disable reverse IPaddress lookups
-6 Use IPv6 only
-A Use several features, including OS Detection, Version Detection, Script Scanning (default), and traceroute
--reason Display reason Nmap thinks port is open, closed, or filtered

POWERSHELL

Syntax

Cmdlets are small scripts that follow a dash-separated verb-noun convention such as "Get-Process".

SIMILAR VERBS WITH DIFFERENT ACTIONS:

- New- Creates a new resource
- Set- Modifies an existing resource
- Get- Retrieves an existing resource
- Read- Gets information from a source, such as a file
- Find- Used to look for an object
- Search- Used to create a reference to a resource
- Start- (asynchronous) begin an operation, such as starting a process
- Invoke- (synchronous) perform an operation such as running a command

PARAMETERS:
Each verb-noun named cmdlet may have many parameters to control cmdlet functionality.

OBJECTS:
The output of most cmdlets are objects that can be passed to other cmdlets and further acted upon. This becomes important in pipelining cmdlets.

5 PowerShell Essentials

CONCEPT	WHAT'S IT DO?	A HANDY ALIAS
PS C:\> Get-Help [cmdlet] -examples	Shows help & examples	PS C:\> **Help [cmdlet] -examples**
PS C:\> Get-Command	Shows a list of commands	PS C:\> **gcm *[string]***
PS C:\> Get-Member	Shows properties & methods	PS C:\> [cmdlet] | gm
PS C:\> ForEach-Object { $_ }	Takes each item on pipeline and handles it as $_	PS C:\> **[cmdlet] | % { [cmdlet] $_ }**
PS C:\> Select-String	Searches for strings in files or output, like grep	PS C:\> **sls –path [file] –pattern [string]**

Finding Cmdlets

To get a list of all available cmdlets:
```
PS C:\> Get-Command
```
Get-Command supports filtering. To filter cmdlets on the verb set:
```
PS C:\> Get-Command Set* or
PS C:\> Get-Command –Verb Set
```
Or on the noun "Process":
```
PS C:\> Get-Command *Process or
PS C:\> Get-Command –Noun process
```

Pipelining, Loops, and Variables

Piping cmdlet output to another cmdlet:
```
PS C:\> Get-Process | Format-List –property name
```
ForEach-Object in the pipeline (alias %):
```
PS C:\> ls *.txt | ForEach-Object {cat $_}
```
Where-Object condition (alias where or ?):
```
PS C:\> Get-Process | Where-Object {$_.name –eq "notepad"}
```
Generating ranges of numbers and looping:
```
PS C:\> 1..10
PS C:\> 1..10 | % {echo "Hello!"}
```
Creating and listing variables:
```
PS C:\> $tmol = 42
PS C:\> ls variable:
```
Examples of passing cmdlet output down pipeline:
```
PS C:\> dir | group extension | sort
PS C:\> Get-Service dhcp | Stop-Service -PassThru | Set-Service -StartupType Disabled
```

Getting Help

To get help with help:
```
PS C:\> Get-Help
```
To read cmdlet self documentation:
```
PS C:\> Get-Help <cmdlet>
```
Detailed help:
```
PS C:\> Get-Help <cmdlet> -detailed
```
Usage examples:
```
PS C:\> Get-Help <cmdlet> -examples
```
Full (everything) help:
```
PS C:\> Get-Help <cmdlet> -full
```
Online help (if available):
```
PS C:\> Get-Help <cmdlet> -online
```

Efficient PowerShell

TAB COMPLETION:
```
PS C:\> get-child<TAB>
PS C:\> Get-ChildItem
```
Parameter shortening:
```
PS C:\> ls –recurse is equivalent to:
PS C:\> ls -r
```

Cmdlet Aliases

Aliases provide short references to long commands.
To list available aliases (alias alias):
```
PS C:\> Get-Alias
```
To expand an alias into a full name:
```
PS C:\> alias <unknown alias>
PS C:\> alias gcm
```

SCAPY

Scapy Basics

To list supported layers:
```
>>> ls()
```
Some key layers are:
arp, ip, ipv6, tcp, udp, icmp

To view layer fields use ls(layer):
```
>>> ls(IPv6)
>>> ls(TCP)
```
To list available commands:
```
>>> lsc()
```
Some key commands for interacting with packets:
rdpcap, send, sr, sniff, wrpcap

Getting help with commands use help(command):
```
>>> help(rdpcap)
```

Basic Packet Crafting / Viewing

Scapy works with layers. Layers are individual functions linked together with the "/" character to construct packets. To build a basic TCP/IP packet with "data" as the payload:
```
>>> packet = IP(dst="1.2.3.4")/TCP(dport=22)/"data"
```
Note: Scapy allows the user to craft all the way down to the ether() (Data Link) layer, but will use default values for the data link layer if it's omitted when using the send() or sr() functions. To correctly pass traffic, layers should be ordered from lowest to highest from left to right (e.g., ether -> IP -> TCP).

To get a packet summary:
```
>>> packet.summary()
```
To get more packet details:
```
>>> packet.show()
```

Receiving and Analyzing Packets

Received packets can be stored in a variable when using a send/receive function such as sr(), srp(), sr1() sr1p():
```
>>> packet = IP(dst="10.10.10.20")/TCP(dport=(0,1024))
>>> unans, ans = sr(packet)
```
Received 1086 packets, got 1024 answers, remaining 0 packets

"ans" will store the answered packets:
```
>>> ans
<Results: TCP:1024 UDP:0 ICMP:0 Other:0>
```
To see a summary of the responses:
```
>>> ans.summary()
IP / TCP 10.1.1.15:ftp_data > 10.10.10.20:netbios_ssn S ==> IP / TCP 10.10.10.20:netbios_ssn > 10.1.1.15:ftp_data SA / Padding
```
Note: this is the output from port 139 (netbios_ssn). Notice how this port was open and responded with a SYN-ACK.

To view a specific pair of sent/replied packets:
```
>>> ans[15]
```
To view the first packet in the stream:
```
>>> ans[15][0] (this will be packet the Scapy sent)
<IP  frag=0 proto=tcp dst=10.10.10.20 |<TCP dport=netstat flags=S |>>
```
To view the response from the distant end:
```
>>> ans[15][1]
<IP  version=4L ihl=5L tos=0x0 len=40 id=16355 flags=DF frag=0L ttl=128 proto=tcp chksum=0x368c src=10.10.10.20 dst=10.1.1.15 options=[] |<TCP  sport=netstat dport=ftp_data seq=0 ack=1 dataofs=5L reserved=0L flags=RA window=0 chksum=0x2b4c urgptr=0 |<Padding load='\x00\x00\x00\x00\x00\x00' |>>>
```
To view the TCP flags in the response packet:
```
>>> ans[15][1].sprintf("%TCP.flags%")
'RA'
```

Sniffing and pcaps

To sniff using Berkley Packet Filters:
```
>>> packets = sniff(filter="host 1.1.1.1")
```
Sniffing using counts:
```
>>> packets = sniff(count=100)
```
Reading packets from a pcap:
```
>>> packets = rdpcap("filename.pcap")
```
Writing packets to a pcap:
```
>>> wrpcap("filename.pcap", packets)
```

Sending Packets

CREATING AND SENDING A PACKET
```
>>> packet = IP(dst="4.5.6.7")/TCP(dport=80, flags="S")
```
Send a packet, or list of packets without custom ether layer:
```
>>> send(packet)
```
SEND FUNCTION OPTIONS

filter = <Berkley Packet Filter>
retry = <retry count for unanswered packets> timeout = <number of seconds to wait before giving up>
iface = <interface to send and receive>
```
>>> packets = sr(packet, retry=5, timeout=1.5, iface="eth0", filter="host 1.2.3.4 and port 80")
```

METASPLOIT

Post Modules from Meterpreter

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

RUN POST MODULES FROM METERPRETER
```
meterpreter > run post/multi/gather/env
```
RUN POST MODULES ON A BACKGROUNDED SESSION
```
msf > use post/windows/gather/hashdump
msf > show options
msf > set SESSION 1
msf > run
```

Useful Auxiliary Modules

TCP PORT SCANNER:
```
msf > use auxiliary/scanner/portscan/tcp
msf > set RHOSTS 10.10.10.0/24
msf > run
```
DNS ENUMERATION
```
msf > use auxiliary/gather/dns_enum
msf > set DOMAIN target.tgt
msf > run
```
FTP SERVER
```
msf > use auxiliary/server/ftp
msf > set FTPROOT /tmp/ftproot
msf > run
```
PROXY SERVER
Create a socks4 proxy on the local machine that allows external tools to use Metasploit's routing.
```
msf > use auxiliary/server/socks4
msf > run
```

Managing Sessions

MULTIPLE EXPLOITATION:
Run the exploit expecting a single session that is immediately backgrounded:
```
msf > exploit -z
```
Run the exploit in the background, so that msfconsole can still be used while the exploit is running:
```
msf > exploit –j
```
List all current jobs (usually exploit listeners):
```
msf > jobs –l
```
Kill a job:
```
msf > jobs –k [JobID]
```

Metasploit Console Basics (msfconsole)

SEARCH FOR MODULE:
```
msf > search [criteria]
```
SPECIFY AN EXPLOIT TO USE:
```
msf > use exploit/[ExploitPath]
```
SPECIFY A PAYLOAD TO USE:
```
msf > set PAYLOAD [PayloadPath]
```
SHOW OPTIONS FOR THE CURRENT MODULES:
```
msf > show options
```
SET OPTIONS:
```
msf > set [Option] [Value]
```
START EXPLOIT:
```
msf > exploit
```

Metasploit Meterpreter

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

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

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

NETWORK COMMANDS:
ipconfig: Show network interface information
portfwd: Forward packets through TCP session
route: Manage/view the exploited system's routing table

SLINGSHOT LINUX DISTRIBUTION

The Slingshot Linux distribution is used for a variety of different SANS Penetration Testing courses.

Slingshot's tool arsenal has been thoroughly tested to ensure excellent results in course labs and in penetration testing projects.

Slingshot includes the following tools:

- THE METASPLOIT FRAMEWORK
- THE ARMITAGE GUI FOR METASPLOIT
- ETTERCAP MAN IN THE MIDDLE TOOL
- EXIFTOOL FOR METADATA ANALYSIS
- HYDRA PASSWORD GUESSING TOOL
- JOHN THE RIPPER PASSWORD CRACKING TOOL
- LAIR FRAMEWORK PEN TEST COLLABORATION TOOL
- NETCAT GENERAL PURPOSE TCP/UDP TOOL
- NESSUS VULNERABILITY SCANNER
- NIKTO WEB SCANNER
- NMAP PORT SCANNER AND GENERAL PURPOSE PACKET TOOL
- RECON-NG RECONNAISSANCE TOOL
- SCAPY PACKET SUITE
- SOCIAL ENGINEERING TOOLKIT
- TCPDUMP SNIFFER
- WIRESHARK SNIFFER
- VEIL-EVASION ANTI-VIRUS EVASION TOOL
- POWERSHELL EMPIRE POST-EXPLOITATION TOOLKIT
- ZED ATTACK PROXY (ZAP) WEB APPLICATION ATTACK TOOL

01101000 01101111 01101100 01101001 01100100 01100001 01111001 01101000 01100001 01100011 01101011 01100011
01101000 01100001 01101100 01101100 01100101 01101110 01100111 01100101 00101110 01100011 01101111 01101101