Network Capture Analysis

Dale Stubbs - 14024149

Abstract

For a forensic examiner the ability to comprehend the files associated with packet captures from a suspected network based attack is paramount. The ability to determine malicious traffic from regular traffic can be considered the most important ability. This paper looks into three different network packet capture files. The author looks into these files and attempts to determine if any malicious behaviour is being conducted within the file. Once the malicious behaviour has been detected the author attempts to provide a source of mitigation against these forms of malicious behaviours.

Keywords: PCAP, Network, Malicious Behaviour, Mitigation, Port Scan, Dictionary Password Attack, Brute Force Password Attack.

Contents

List of Tables

List of Images

Introduction

Hash values are generated on each piece of evidence before (to give a 'base' value for comparison purposes) and after (to ensure the data has not been modified) any investigation on the material in question can be carried out. This is done to ensure the validity of the evidence gathered from the material being scrutinised.

This is accomplished by how hash values are generated. Hash values are *'a numeric value of a fixed length that uniquely identifies data.' (Msdn.microsoft.com, 2016).* There are multiple ways to generate a hash value for any material which ensures that the data integrity has not been altered throughout the investigation.

The hash values are generated based on the hexadecimal values of the data provided. This means that a modification to a single byte of data will result in the hash value being generated after the investigation not matching the original hash value, rendering all of the evidence gathered as inadmissible in court.

Pre-Investigation Hash Values

Evidence Name	MD5 Hash Value	SHA1 Hash Value	SHA512 Hash Value
Networkcapture 1.pcap	b834dbf7fad0d1e07 4529b03ad141110	050bb43df2c7e8c03 ab920295370f79651 733288	6526bfd37e7ed6e9643c9a96e49fc 4e561371f292cafd8194ad7acc6ad cabad6f1c9ecf3efc20871dc2e0c3e 8f3d3e5004007203ecb6f5f3b87a6 7384f7a4178
Networkcapture 2.pcap	744d49d048735262 2d624837c2cf589c	b330894d9e5243cb8 06cef971e478233a9 d39b7a	ed66c7b49cde415ad0e4bc7abedf3b a967cdfb39318374700cef73f22e5c 9364dc6e8f6b552d921fb18993a193 27d622abaed421576b263d9938a1a fc36ee313
Networkcapture 3.pcap	e12b2d244d23b988 2301cd4ad52ed1de	b02de56eb4e6aac8a6 944775be95bae259c f5483	b5cb9b447c2713833b617abe07649 b895a027f46fd362e3e92fa3a5737e e2c08802d86cd8e7d1d4431800271 b33cc179f22977144480917cb54f750 6acd896b0

Table 1: Pre Investigation Hash Values

Multiple hash values have been generated to ensure a multiple point validity check after the investigation of the material has been completed.

Frameworks

ACPO Guidelines (Appendix 1). I will be using the ACPO guidelines for the ensuring that the evidence provided is admissible in court. More specifically, I will be using Principle 1, *No action taken by law enforcement agencies, persons employed within those agencies or their agents should change data which may subsequently be relied upon in court. (Williams, 2012)*
This refers to the hashing of the files before and after being worked on to ensure authenticity of the evidence gathered.

Wireshark - I will be using Wireshark to analyse and scrutinise the data provided to attempt to discover any and all malicious/inappropriate behaviour during the provided Network Packet Captures (pcap files).

Tables of Terms

Actor Name	Description
Target	The victim of the attack
Aggressor	The perpetrator of the attack
Intermediary	A middleman to the attack
Spectator	An innocent bystander that has nothing to do with the attack.

Table 2: Actors

Acronyms	Full Term	Definition
TCP	Transmission Control Protocol	A standard the defines how a connection between two nodes
FTP	File Transfer Protocol	A standard internet protocol designed for the transfer of files between computers via the TCP/IP connection.
SYN	Synchronise Flag	A flag of the TCP connection
ACK	Acknowledge Flag	A flag of the TCP connection
RST	Reset Flag	A flag of the TCP connection
FIN	Finish Flag	A flag of the TCP connection
MAC Address	Media Access Control Address	A unique hardware number assigned to the computer
HTTP	Hypertext Transfer Protocol	A protocol for transferring files such as text, images, sound and video files on the World Wide Web.
TYPE I	Type Image	Sets the transfer data type to binary form
PASV	Passive FTP	Sets the FTP to a passive form (appendix 6)

Table 3: Acronyms

Concept Name	Description
Port Scan	A scan of all ports on a computer to determine any access points that may be exploited.
HTTP Request and Response	A transfer of HTTP packets between host and client
Password Attack	An attempt to guess someone's password without authority to do so.

Table 4: Frequently Used Concepts

Capture 1

Capture Overview

This is the first Network Capture provided named 'Networkcapture1.pcap'. My name is Dale Stubbs and my student number is 14024149.

This network capture was completed on 11/02/2010 at 16:20, monitors network traffic for 5 minutes and ceases at 16:25. A total of 48,370 packets were captured in this time.

Table of Actors

Name	IP Address	Description	Mac Address	Role
Jarvis	192.168.56.101	Server: apache/2.2.11	CadmusCo_00:48:4d	Target
Lucille	192.168.56.104	N/A	CadmusCo_1d:1b:ab	Aggressor
Friday	192.168.56.1	N/A	CadmusCo_8e:ca:c2	Spectator

Table 5: Actors within NetworkCatpure1.pcap

Malicious Behaviour

- At approximately 16:20 on the 11/02/2010 the network capture began.
- An initialization of the TCP three-way handshake is started by Friday aimed at Jarvis using the SYN packet.
- Friday then sends out an ARP announcement to ensure that Jarvis is aware of their IP and MAC Address.

2	0.004486	CadmusCo_8e:ca:c2	Broadcast	ARP	42	Who has 192.168.56.1? Tell 192.168.56.101
3	0.004707	CadmusCo_00:48:4d	CadmusCo_8e:ca:c2	ARP	60	192.168.56.1 is at 08:00:27:00:48:4d

Image 1: Friday joins using ARP

- The connection between Friday and Jarvis is then completed (See appendix 1) via port 2812 of Friday and port 80 of Jarvis.

- Friday then begins to request several different items from Jarvis with little success (appendix 2).

6	0.005117	192.168.56.1	192.168.56.101	HTTP	599	GET / HTTP/1.1
7	0.005157	192.168.56.101	192.168.56.1	TCP	54	80→2812 [ACK] Seq=1 Ack=546
8	0.190276	192.168.56.101	192.168.56.1	HTTP	750	HTTP/1.1 200 OK (text/html)
9	0.216901	192.168.56.1	192.168.56.101	HTTP	401	GET /favicon.ico HTTP/1.1
10	0.216950	192.168.56.101	192.168.56.1	TCP	54	80→2812 [ACK] Seq=697 Ack=89
11	0.221121	192.168.56.101	192.168.56.1	HTTP	558	HTTP/1.1 404 Not Found (tex
12	0.389476	192.168.56.1	192.168.56.101	TCP	60	2812→80 [ACK] Seq=893 Ack=12
13	5.354306	192.168.56.1	192.168.56.101	HTTP	614	GET /test.html HTTP/1.1
14	5.355934	192.168.56.101	192.168.56.1	HTTP	605	HTTP/1.1 200 OK (text/html)
15	5.367232	192.168.56.1	192.168.56.101	HTTP	401	GET /favicon.ico HTTP/1.1
16	5.367949	192.168.56.101	192.168.56.1	HTTP	558	HTTP/1.1 404 Not Found (tex

Image 2: Friday and Jarvis HTTP Requests and Responses

- Friday then switches to port 2813 using the three-way handshake (appendix 3) and proceeds to retrieve more items from Jarvis.
- Jarvis then terminates the connections to both ports 2812 and 2813 of Friday using the FIN packet (appendix 2).
- Lucille then joins the group via IGMPv3 (appendix 4) protocol and begins a series of MDNS queries (appendix 5).

49	44.403567	192.168.56.104	224.0.0.22	IGMPv3	60	Membership Report / Join group
50	44.606083	192.168.56.104	224.0.0.251	MDNS	258	Standard query 0x0000 ANY ubunt
51	44.633415	192.168.56.104	224.0.0.251	MDNS	163	Standard query response 0x0000
52	44.666868	192.168.56.104	224.0.0.251	MDNS	200	Standard query 0x0000 ANY b.a.b
53	44.853359	192.168.56.104	224.0.0.251	MDNS	258	Standard query 0x0000 ANY ubunt
54	44.917544	192.168.56.104	224.0.0.251	MDNS	200	Standard query 0x0000 ANY b.a.b
55	45.104985	192.168.56.104	224.0.0.251	MDNS	258	Standard query 0x0000 ANY ubunt
56	45.170102	192.168.56.104	224.0.0.251	MDNS	200	Standard query 0x0000 ANY b.a.b
57	45.305299	192.168.56.104	224.0.0.251	MDNS	240	Standard query response 0x0000

Image 3: Lucille MDNS Queries

- Lucille then sends out an ARP packet to request information about Jarvis' IP Address.
- Lucille then pings Jarvis and then sends out and ARP packet to tell Jarvis their IP and MAC Addresses
- Friday reconnects to Jarvis and requests some more HTTP Get Requests (appendix 6).
- Lucille then resends the ARP packet for Jarvis' information.
- Lucille then begins a Sequential Port Scan on Jarvis. Packet 94 to packet 48,261 is the complete run down of the port scan. This theory is backed up by the groups of SYN requests in between single RST, ACK replies at different stages of the scan.

47809	170.925706	192.168.56.104	192.168.56.101	TCP	60	59290→64849 [SYN] Seq=0
47810	170.927864	192.168.56.104	192.168.56.101	TCP	60	59290→64850 [SYN] Seq=0

Image 4: Multiple TCP SYN Requests Together

48259	170.993626	192.168.56.101	192.168.56.104	TCP	54	65532→59290 [RST, ACK]
48260	170.993636	192.168.56.101	192.168.56.104	TCP	54	65533→59290 [RST, ACK]
48261	170.993643	192.168.56.101	192.168.56.104	TCP	54	65534→59290 [RST, ACK]

Image 5: Multiple TCP RST, ACK Responses Together

- At packet 48,321 Lucille leaves the group whilst Friday and Jarvis hand packets back and forth regarding HTTP requests.

48321	215.260058	192.168.56.104	224.0.0.22	IGMPv3	60	Membership Report / Leav
48322	223.394232	192.168.56.1	192.168.56.101	TCP	62	2817→80 [SYN] Seq=0 Win=
48323	223.394272	192.168.56.101	192.168.56.1	TCP	62	80→2817 [SYN, ACK] Seq=(
48324	223.394770	192.168.56.1	192.168.56.101	TCP	60	2817→80 [ACK] Seq=1 Ack=
48325	223.394815	192.168.56.1	192.168.56.101	HTTP	559	GET /test.html HTTP/1.1
48326	223.394834	192.168.56.101	192.168.56.1	TCP	54	80→2817 [ACK] Seq=1 Ack=
48327	223.395980	192.168.56.101	192.168.56.1	HTTP	606	HTTP/1.1 200 OK (text/h
48328	223.554377	192.168.56.1	192.168.56.101	TCP	60	2817→80 [ACK] Seq=506 Ac
48329	225.123935	192.168.56.1	192.168.56.101	HTTP	572	GET /Network%20Attacks.h
48330	225.124624	192.168.56.101	192.168.56.1	HTTP	608	HTTP/1.1 200 OK (text/h
48331	225.258547	192.168.56.1	192.168.56.101	TCP	60	2817→80 [ACK] Seq=1024 A
48332	227.013210	192.168.56.1	192.168.56.101	HTTP	560	GET /Lena.html HTTP/1.1
48333	227.013851	192.168.56.101	192.168.56.1	HTTP	642	HTTP/1.1 200 OK (text/h

Image 6: Lucille leaves after completing the port scan

- Network capture concludes at 16:25 of the same date.

Mitigation

Port scan on networks can easily be flagged up by using *'Network security applications [can be] configured to alert administrators if they detect connection requests across a broad range of ports from a single host.' (Lifewire, 2016)* This would enable anyone being the subject of a port scan to easily detect the port scan and can take the necessary steps to stop the attacker in their tracks.

Another way of mitigating an attack of this calibre would be to block the IP address that is guilty of port scanning you. This can be done via the router itself. You would need to have administrative privileges on the router in order to accomplish this. If you have administrator privileges then the use of the iptables rule is how to accomplish blocking ip addresses from the server. This program allows the user to manually add malicious IP addresses to a list of blocked IP addresses denying them contact in any future connection attempts. There is also a software available to complete this for you, Snort. Snort is an open-source network intrusion detection and prevention system. Using either of these aforementioned approaches would effectively mitigate against this form of attack.

Capture 2

Capture Overview

This is the second Network Capture provided named ‘Networkcapture2.pcap’. The examiners name is Dale Stubbs and my student number is 14024149.

This network capture was completed on 15/02/2010 at 15:38, monitors network traffic for one hour and 40 minutes and ceases at 17:18. A total of 14,240 packets were captured in this time.

Table of Actors

Name	IP Address	Description	Mac Address	Role
Jarvis	192.168.56.101	Server: apache/2.2.11	CadmusCo_00:48:4d	Intermediary
Gideon	192.168.56.102		CadmusCo_1d:1b:ab	Target
Friday	192.168.56.1		CadmusCo_8e:ca:c2	Aggressor

Table 6: Actors within NetworkCapture2.pcap

Malicious Behaviour

- At approximately 15:38 on 15/02/2010 the network capture begins and we are greeted with Gideon logging onto Jarvis using the user name ‘ftpuser’ and password ‘cmpsem055’ and begin to set a transfer mode using the commands ‘TYPE I’ and ‘PASV’ (see ‘Table 3: Acronyms’) and then prints the current working directory ‘PWD’ followed by listing all of the contents of the directory ‘LIST’. This exchange happens between packets 1 to 44.

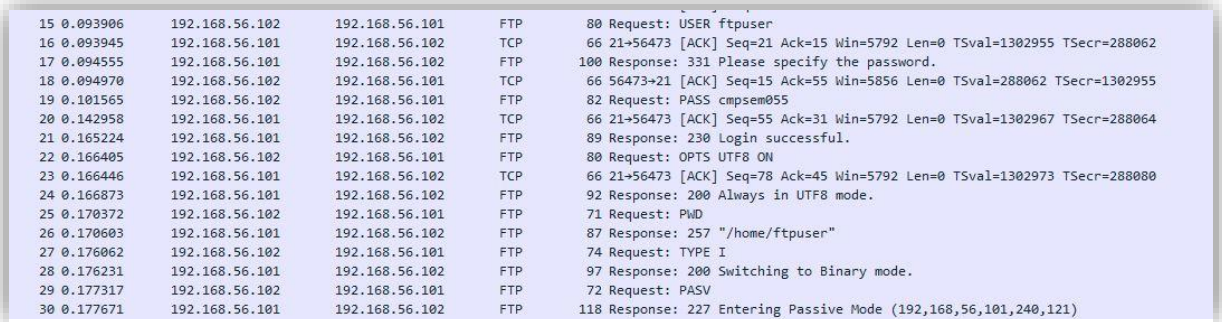


Image 7: Gideon connects to Jarvis

- Following this exchange Gideon then logs onto Jarvis again and proceeds to store a document called Confidential Information.doc in the ‘/home/ftpuser’ directory. This occurs between packets 55 and 63

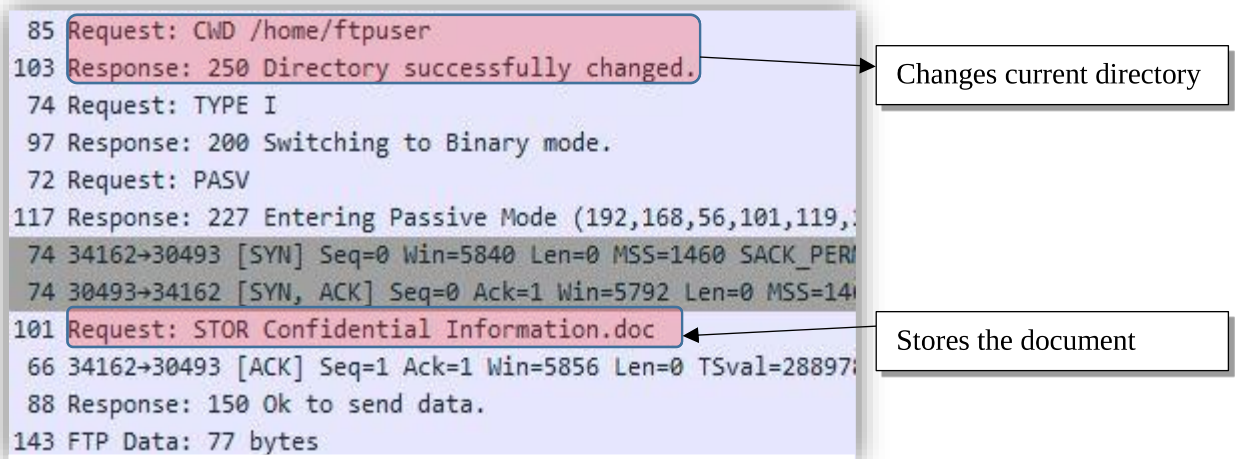


Image 8: Gideon stores a file

- Friday then connects to Jarvis at packet 87 on port 21 (appendix 7). Friday then issues an ARP command to request the MAC Address of Jarvis.

87 19.324364	192.168.56.1	192.168.56.101	TCP	62 1888→21 [SYN] Seq=0 Win=64512 Len=0 MSS=1460 SACK_PERM=1
88 19.327867	CadmusCo_8e:ca:c2	Broadcast	ARP	42 Who has 192.168.56.1? Tell 192.168.56.101
89 19.328258	CadmusCo_00:48:4d	CadmusCo_8e:ca:c2	ARP	60 192.168.56.1 is at 08:00:27:00:48:4d
90 19.328296	192.168.56.101	192.168.56.1	TCP	62 21→1888 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1

Image 9: Friday's ARP Request

- Friday then begins a password attack in an attempt to gain access to the files stored within Jarvis. The password attack used in this scenario is called a 'Dictionary Password Attack' (appendix 8). The first attempt to access the files is at packet 105. Friday uses the user name 'ftpuser' and uses the password 'aaa'.

102 19.345613	192.168.56.1	192.168.56.101	FTP	68 Request: USER ftpuser
103 19.345645	192.168.56.101	192.168.56.1	TCP	54 21→1889 [ACK] Seq=21 Ack=15 Win=5840 Len=0
104 19.346340	192.168.56.101	192.168.56.1	FTP	88 Response: 331 Please specify the password.
105 19.346620	192.168.56.1	192.168.56.101	FTP	64 Request: PASS aaa
106 19.384418	192.168.56.101	192.168.56.1	TCP	54 21→1889 [ACK] Seq=55 Ack=25 Win=5840 Len=0
107 22.252294	192.168.56.101	192.168.56.1	FTP	76 Response: 530 Login incorrect.
108 22.253347	192.168.56.1	192.168.56.101	TCP	60 1889→21 [FIN, ACK] Seq=25 Ack=77 Win=64436 Len=0
109 22.253494	192.168.56.101	192.168.56.1	FTP	64 Response: 500 OOPS:

Image 10: Start of the password attack

- After each unsuccessful password attempt, Friday switches the port that they are transmitting the requests from.

112 22.254145	192.168.56.1	192.168.56.101	TCP	62 1890→21 [SYN] Seq=0 Win=64512 Len=0 MSS=1460 SACK_PERM=1
113 22.254163	192.168.56.101	192.168.56.1	TCP	62 21→1890 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
114 22.254174	192.168.56.1	192.168.56.101	TCP	60 1889→21 [RST, ACK] Seq=26 Ack=87 Win=0 Len=0
115 22.254183	192.168.56.1	192.168.56.101	TCP	60 1889→21 [RST] Seq=26 Win=0 Len=0
116 22.254187	192.168.56.1	192.168.56.101	TCP	60 1889→21 [RST] Seq=26 Win=0 Len=0
117 22.254589	192.168.56.1	192.168.56.101	TCP	60 1890→21 [ACK] Seq=1 Ack=1 Win=64512 Len=0

Image 11: Port switch

- During the attack Gideon and Jarvis continue to transfer packets for HTTP requests as well as FTP requests showing that Gideon has continue to go about their business unaware of the attack by Friday.

197 36.899069	192.168.56.102	192.168.56.101	HTTP	601 GET /Network%20Attacks.html HTTP/1.1
198 36.913578	192.168.56.101	192.168.56.102	HTTP	620 HTTP/1.1 200 OK (text/html)
199 36.914246	192.168.56.102	192.168.56.101	TCP	66 42644→80 [ACK] Seq=1084 Ack=1107 Win=8

Image 12: Gideon and Jarvis traffic

350 62.191259	192.168.56.1	192.168.56.101	TCP	60 1903→21 [ACK] Seq=1 Ack=1 Win=64512 Len=0
351 62.198266	192.168.56.101	192.168.56.1	FTP	74 Response: 220 (vsFTPd 2.0.7)
352 62.199197	192.168.56.1	192.168.56.101	FTP	68 Request: USER ftpuser
353 62.199238	192.168.56.101	192.168.56.1	TCP	54 21→1903 [ACK] Seq=21 Ack=15 Win=5840 Len=0
354 62.199873	192.168.56.101	192.168.56.1	FTP	88 Response: 331 Please specify the password.
355 62.200519	192.168.56.1	192.168.56.101	FTP	67 Request: PASS albert
356 62.242029	192.168.56.101	192.168.56.1	TCP	54 21→1903 [ACK] Seq=55 Ack=28 Win=5840 Len=0
357 62.531657	192.168.56.102	192.168.56.101	FTP	77 Request: CWD /home
358 62.532111	192.168.56.101	192.168.56.102	FTP	103 Response: 250 Directory successfully changed.
359 62.545463	192.168.56.102	192.168.56.101	FTP	71 Request: PWD
360 62.545677	192.168.56.101	192.168.56.102	FTP	79 Response: 257 "/home"
361 62.553994	192.168.56.102	192.168.56.101	FTP	72 Request: PASV

Image 13: Gideon, Jarvis and Friday traffic

- At packet 14,143 Friday successfully logs on using the user name ‘ftpuser’ with the password ‘cmpsem055’ and successfully transfers the document ‘Confidential Information.doc’ to their own machine meaning the confidentiality of the contained information has now been destroyed at packet 14223.

14138	2545.980326	192.168.56.1	192.168.56.101	TCP	60 2764→21 [ACK] Seq=1 Ack=1 Win=64512 Len=0
14139	2545.987719	192.168.56.101	192.168.56.1	FTP	74 Response: 220 (vsFTPd 2.0.7)
14140	2545.988351	192.168.56.1	192.168.56.101	FTP	68 Request: USER ftpuser
14141	2545.988376	192.168.56.101	192.168.56.1	TCP	54 21→2764 [ACK] Seq=21 Ack=15 Win=5840 Len=0
14142	2545.988650	192.168.56.101	192.168.56.1	FTP	88 Response: 331 Please specify the password.
14143	2545.989198	192.168.56.1	192.168.56.101	FTP	70 Request: PASS cmpsem055
14144	2546.029578	192.168.56.101	192.168.56.1	TCP	54 21→2764 [ACK] Seq=55 Ack=31 Win=5840 Len=0
14145	2546.053891	192.168.56.101	192.168.56.1	FTP	77 Response: 230 Login successful.
14146	2546.055648	192.168.56.1	192.168.56.101	TCP	60 2764→21 [FIN, ACK] Seq=31 Ack=78 Win=64435 Len=0
14147	2546.055896	192.168.56.101	192.168.56.1	FTP	64 Response: 500 OOPS:
14148	2546.055992	192.168.56.101	192.168.56.1	FTP	84 Response: vsf_sysutil_recv_peek: no data

Image 14: Friday is successful at cracking the password

14213	6019.886675	192.168.56.101	192.168.56.1	TCP	54 21→2841 [ACK] Seq=78 Ack=45 Win=5840 Len=0
14214	6019.887210	192.168.56.101	192.168.56.1	FTP	80 Response: 200 Always in UTF8 mode.
14215	6019.888317	192.168.56.1	192.168.56.101	FTP	73 Request: CWD /home/ftpuser
14216	6019.888557	192.168.56.101	192.168.56.1	FTP	91 Response: 250 Directory successfully changed.
14217	6019.994394	192.168.56.1	192.168.56.101	TCP	60 2841→21 [ACK] Seq=64 Ack=141 Win=64372 Len=0
14218	6021.016593	192.168.56.1	192.168.56.101	FTP	62 Request: TYPE I
14219	6021.017135	192.168.56.101	192.168.56.1	FTP	85 Response: 200 Switching to Binary mode.
14220	6021.017808	192.168.56.1	192.168.56.101	FTP	60 Request: PASV
14221	6021.018343	192.168.56.101	192.168.56.1	FTP	105 Response: 227 Entering Passive Mode (192,168,56,101,61,163)
14222	6021.019739	192.168.56.1	192.168.56.101	FTP	89 Request: RETR Confidential Information.doc
14223	6021.020012	192.168.56.1	192.168.56.101	TCP	66 2842→15779 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=128 SACK_PERM=1
14224	6021.020034	192.168.56.101	192.168.56.1	TCP	66 15779→2842 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=32
14225	6021.020242	192.168.56.1	192.168.56.101	TCP	60 2842→15779 [ACK] Seq=1 Ack=1 Win=4194304 Len=0
14226	6021.020662	192.168.56.101	192.168.56.1	FTP	140 Response: 150 Opening BINARY mode data connection for Confidential Information.doc (77 bytes).
14227	6021.020696	192.168.56.101	192.168.56.1	FTP-DA…	131 FTP Data: 77 bytes

Image 15: Friday transfers the confidential file

- Friday then terminates the connection to Jarvis at packets 14,238, 14,239 and 14,240.

14238	6025.325873	192.168.56.1	192.168.56.101	TCP	60 2839→21 [RST, ACK] Seq=71 Ack=279 Win=0 Len=0
14239	6025.325890	192.168.56.1	192.168.56.101	TCP	60 2839→21 [RST] Seq=71 Win=0 Len=0
14240	6025.325895	192.168.56.1	192.168.56.101	TCP	60 2839→21 [RST] Seq=71 Win=0 Len=0

Image 16: Friday disconnects

- Packet capture terminates at 17:18.

Mitigation

There are several steps that can be taken to mitigate attempted password attacks on a network. They are all based on the network administrators’ perspective. Firstly, enforce a ‘lockout’ procedure when someone attempts to log in incorrectly after a set number of attempts, secondly, attempt to enforce the use much stronger passwords. For example, a phrase could be used as opposed to a word. Something along the lines of ‘mywifeisbeautiful’. *"It would take a computer about 898 THOUSAND YEARS to crack your password" (Collider, 2016).* Thirdly, not allow password reuse. Ensuring that the same password is never used more than once, fourthly, not allowing clear text storage and instead using a form of password salting and hashing (appendix 9), and finally never allow default passwords to be used after user creation. Some people never change from the default password, whether this is through laziness or simply not knowing how to change the password. This creates a very easy point of failure within the network as anyone can easily acquire a list of default passwords from the internet and use each of these very easily. The focus of mitigating password attacks is to make the cracking of the password more costly than what the attacker would gain from cracking the password.

Capture 3

Capture Overview

This is the third and final Network Capture provided named 'Networkcapture3.pcap'. The examiners name is Dale Stubbs and my student number is 14024149.

This network capture was completed on 8/03/2010 at 12:21, monitors network traffic for 5 minutes and ceases at 12:46. A total of 81,189 packets were captured in this time.

Table of Actors

Name	IP Address	Description	Mac Address	Role
Jarvis	192.168.56.101	Server: apache/2.2.11	CadmusCo_00:48:4d	Intermediary
Gideon	192.168.56.102		CadmusCo_1d:1b:ab	Target
Friday	192.168.56.1	N/A	CadmusCo_8e:ca:c2	Aggressor

Table 7: Actors within NetworkCapture3.pcap

Malicious Behaviour

- At approximately 12:21 on 8/03/2010 Gideon and Jarvis were exchanging HTTP request and response packets from packets 1 to 29.

1	0.000000	192.168.56.101	192.168.56.102	TCP	74	55037→80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=948392 TSecr=0 WS=32
2	0.000084	192.168.56.102	192.168.56.101	TCP	74	80→55037 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=947196 TSecr=
3	0.000486	192.168.56.101	192.168.56.102	TCP	66	55037→80 [ACK] Seq=1 Ack=1 Win=5856 Len=0 TSval=948393 TSecr=947196
4	0.002770	192.168.56.101	192.168.56.102	HTTP	615	GET /Lena.html HTTP/1.1
5	0.002809	192.168.56.102	192.168.56.101	TCP	66	80→55037 [ACK] Seq=1 Ack=550 Win=6912 Len=0 TSval=947197 TSecr=948393
6	0.005485	192.168.56.102	192.168.56.101	HTTP	655	HTTP/1.1 200 OK (text/html)
7	0.006322	192.168.56.101	192.168.56.102	TCP	66	55037→80 [ACK] Seq=550 Ack=590 Win=7040 Len=0 TSval=948394 TSecr=947197
8	0.024334	192.168.56.101	192.168.56.102	HTTP	599	GET /Pictures/jpg4.jpg HTTP/1.1
9	0.024781	192.168.56.102	192.168.56.101	HTTP	254	HTTP/1.1 304 Not Modified
10	0.065153	192.168.56.101	192.168.56.102	TCP	66	55037→80 [ACK] Seq=1083 Ack=778 Win=8224 Len=0 TSval=948409 TSecr=947202
11	6.149066	192.168.56.101	192.168.56.102	HTTP	617	GET /Ducati.html HTTP/1.1
12	6.149809	192.168.56.102	192.168.56.101	HTTP	660	HTTP/1.1 200 OK (text/html)
13	6.150682	192.168.56.101	192.168.56.102	TCP	66	55037→80 [ACK] Seq=1634 Ack=1372 Win=9408 Len=0 TSval=949931 TSecr=948733
14	6.167723	192.168.56.101	192.168.56.102	HTTP	600	GET /Pictures/916.jpg HTTP/1.1
15	6.168033	192.168.56.102	192.168.56.101	HTTP	254	HTTP/1.1 304 Not Modified
16	6.205650	192.168.56.101	192.168.56.102	TCP	66	55037→80 [ACK] Seq=2168 Ack=1560 Win=10592 Len=0 TSval=949945 TSecr=948738
17	8.565045	192.168.56.101	192.168.56.102	HTTP	619	GET /Dolphins.html HTTP/1.1
18	8.566000	192.168.56.102	192.168.56.101	HTTP	665	HTTP/1.1 200 OK (text/html)
19	8.566664	192.168.56.101	192.168.56.102	TCP	66	55037→80 [ACK] Seq=2721 Ack=2159 Win=11776 Len=0 TSval=950535 TSecr=949337
20	8.584338	192.168.56.101	192.168.56.102	HTTP	605	GET /Pictures/jpg10.jpg HTTP/1.1
21	8.584753	192.168.56.102	192.168.56.101	HTTP	255	HTTP/1.1 304 Not Modified

Image 17: Gideon and Jarvis' HTTP and TCP traffic

- Friday then connects directly to Gideon at packet 23. Friday sends a HTTP request for the Checksums.html page at packet 26.

23	16.199137	192.168.56.1	192.168.56.102	TCP	62	4664→80 [SYN] Seq=0 Win=64512 Len=0 MSS=1460 SACK_PERM=1
24	16.199184	192.168.56.102	192.168.56.1	TCP	62	80→4664 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
25	16.199668	192.168.56.1	192.168.56.102	TCP	60	4664→80 [ACK] Seq=1 Ack=1 Win=64512 Len=0
26	16.199679	192.168.56.1	192.168.56.102	HTTP	591	GET /Checksums.html HTTP/1.1
27	16.199697	192.168.56.102	192.168.56.1	TCP	54	80→4664 [ACK] Seq=1 Ack=538 Win=6444 Len=0
28	16.212145	192.168.56.102	192.168.56.1	HTTP	1370	HTTP/1.1 200 OK (text/html)
29	16.375097	192.168.56.1	192.168.56.102	TCP	60	4664→80 [ACK] Seq=538 Ack=1317 Win=63196 Len=0
30	21.112674	CadmusCo_00:48:4d	Broadcast	ARP	60	Who has 192.168.56.102? Tell 192.168.56.1
31	21.112718	CadmusCo_8e:ca:c2	CadmusCo_00:48:4d	ARP	42	192.168.56.102 is at 08:00:27:8e:ca:c2

Image 18: Friday connects to Gideon

- Friday begins a port scan on Gideon at packets 32 to 88. A successful connection is found at packet 95 on port 21.

32	21.124433	192.168.56.1	192.168.56.102	TCP	60	34227→554 [SYN] Seq=0 Win=2048 Len=0 MSS=1460
33	21.124467	192.168.56.102	192.168.56.1	TCP	54	554→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
34	21.124478	192.168.56.1	192.168.56.102	TCP	60	34227→21 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
35	21.124498	192.168.56.102	192.168.56.1	TCP	58	21→34227 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
36	21.124512	192.168.56.1	192.168.56.102	TCP	60	34227→8888 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37	21.124520	192.168.56.102	192.168.56.1	TCP	54	8888→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
38	21.124528	192.168.56.1	192.168.56.102	TCP	60	34227→110 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
39	21.124535	192.168.56.102	192.168.56.1	TCP	54	110→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40	21.124554	192.168.56.1	192.168.56.102	TCP	60	34227→22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
41	21.124560	192.168.56.102	192.168.56.1	TCP	54	22→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
42	21.124568	192.168.56.1	192.168.56.102	TCP	60	34227→199 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
43	21.124574	192.168.56.102	192.168.56.1	TCP	54	199→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
44	21.124583	192.168.56.1	192.168.56.102	TCP	60	34227→113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
45	21.124926	192.168.56.102	192.168.56.1	TCP	54	113→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
46	21.124940	192.168.56.1	192.168.56.102	TCP	60	34227→25 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
47	21.124946	192.168.56.102	192.168.56.1	TCP	54	25→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
48	21.124953	192.168.56.1	192.168.56.102	TCP	60	34227→445 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
49	21.124958	192.168.56.102	192.168.56.1	TCP	54	445→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
50	21.124995	192.168.56.1	192.168.56.102	TCP	60	34227→587 [SYN] Seq=0 Win=2048 Len=0 MSS=1460
51	21.125002	192.168.56.102	192.168.56.1	TCP	54	587→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
52	21.125023	192.168.56.1	192.168.56.102	TCP	60	34227→21 [RST] Seq=1 Win=0 Len=0
53	21.128025	192.168.56.1	192.168.56.102	TCP	60	34227→1025 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
54	21.128037	192.168.56.102	192.168.56.1	TCP	54	1025→34227 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
55	21.128050	192.168.56.1	192.168.56.102	TCP	60	34227→143 [SYN] Seq=0 Win=2048 Len=0 MSS=1460

Image 19: Port scan on Gideon by Friday

95	30.759227	192.168.56.1	192.168.56.102	TCP	62	4677→21 [SYN] Seq=0 Win=64512 Len=0 MSS=1460 SACK_PERM=1
96	30.759273	192.168.56.102	192.168.56.1	TCP	62	21→4677 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
97	30.759474	192.168.56.1	192.168.56.102	TCP	60	4677→21 [ACK] Seq=1 Ack=1 Win=64512 Len=0

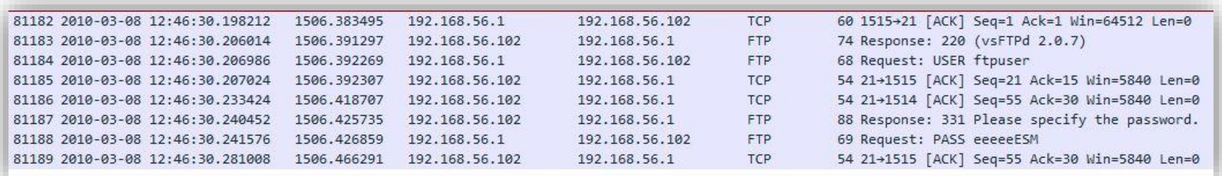
Image 20: Friday finds open port

- Friday then begin a ‘Brute Force Password Attack’ (appendix 10) from packets 130 until the end of the capture.

134	30.779351	192.168.56.102	192.168.56.1	FTP	74	Response: 220 (vsFTPd 2.0.7)
135	30.780730	192.168.56.1	192.168.56.102	FTP	68	Request: USER ftpuser
136	30.780870	192.168.56.102	192.168.56.1	TCP	54	21→4678 [ACK] Seq=21 Ack=15 Win=5840 Len=0
137	30.781195	192.168.56.102	192.168.56.1	FTP	88	Response: 331 Please specify the password.
138	30.782484	192.168.56.1	192.168.56.102	FTP	69	Request: PASS eeeeeeee
139	30.787912	192.168.56.102	192.168.56.1	FTP	74	Response: 220 (vsFTPd 2.0.7)
140	30.788917	192.168.56.1	192.168.56.102	FTP	68	Request: USER ftpuser
141	30.788938	192.168.56.102	192.168.56.1	TCP	54	21→4679 [ACK] Seq=21 Ack=15 Win=5840 Len=0
142	30.789340	192.168.56.102	192.168.56.1	FTP	88	Response: 331 Please specify the password.
143	30.790475	192.168.56.1	192.168.56.102	FTP	69	Request: PASS eeeeeeei
144	30.796524	192.168.56.102	192.168.56.1	FTP	74	Response: 220 (vsFTPd 2.0.7)
145	30.797589	192.168.56.1	192.168.56.102	FTP	68	Request: USER ftpuser
146	30.797615	192.168.56.102	192.168.56.1	TCP	54	21→4680 [ACK] Seq=21 Ack=15 Win=5840 Len=0
147	30.822926	192.168.56.102	192.168.56.1	TCP	54	21→4678 [ACK] Seq=55 Ack=30 Win=5840 Len=0
148	30.823893	192.168.56.102	192.168.56.1	FTP	88	Response: 331 Please specify the password.
149	30.825571	192.168.56.1	192.168.56.102	FTP	69	Request: PASS eeeeeeeo
150	30.830297	192.168.56.102	192.168.56.1	TCP	54	21→4679 [ACK] Seq=55 Ack=30 Win=5840 Len=0
151	30.846694	192.168.56.102	192.168.56.1	FTP	74	Response: 220 (vsFTPd 2.0.7)
152	30.847108	192.168.56.1	192.168.56.102	FTP	68	Request: USER ftpuser
153	30.847148	192.168.56.102	192.168.56.1	TCP	54	21→4681 [ACK] Seq=21 Ack=15 Win=5840 Len=0
154	30.847654	192.168.56.102	192.168.56.1	FTP	88	Response: 331 Please specify the password.
155	30.848740	192.168.56.1	192.168.56.102	FTP	69	Request: PASS eeeeeeeA
156	30.854684	192.168.56.102	192.168.56.1	FTP	74	Response: 220 (vsFTPd 2.0.7)
157	30.856934	192.168.56.1	192.168.56.102	FTP	68	Request: USER ftpuser
158	30.856951	192.168.56.102	192.168.56.1	TCP	54	21→4682 [ACK] Seq=21 Ack=15 Win=5840 Len=0
159	30.863013	192.168.56.102	192.168.56.1	TCP	54	21→4680 [ACK] Seq=55 Ack=30 Win=5840 Len=0
160	30.886292	192.168.56.102	192.168.56.1	TCP	54	21→4681 [ACK] Seq=55 Ack=30 Win=5840 Len=0
161	30.888643	192.168.56.102	192.168.56.1	FTP	74	Response: 220 (vsFTPd 2.0.7)
162	30.889114	192.168.56.1	192.168.56.102	FTP	68	Request: USER ftpuser
163	30.889134	192.168.56.102	192.168.56.1	TCP	54	21→4683 [ACK] Seq=21 Ack=15 Win=5840 Len=0
164	30.896124	192.168.56.102	192.168.56.1	FTP	88	Response: 331 Please specify the password.
165	30.896675	192.168.56.1	192.168.56.102	FTP	69	Request: PASS eeeeeeeI
166	30.896711	192.168.56.102	192.168.56.1	TCP	54	21→4682 [ACK] Seq=55 Ack=30 Win=5840 Len=0
167	30.924964	192.168.56.102	192.168.56.1	FTP	74	Response: 220 (vsFTPd 2.0.7)

Image 21: Start of password attack

- Friday fails to crack the password during this attempted password attack before the capture is terminated. This is unusual as Friday is attempting to crack the password for the user name ‘ftpuser’ using a Brute Force attack after successfully cracking the password for this user using a Dictionary attack less than one month earlier.



81182	2010-03-08 12:46:30.198212	1506.383495	192.168.56.1	192.168.56.102	TCP	60	1515→21 [ACK] Seq=1 Ack=1 Win=64512 Len=0
81183	2010-03-08 12:46:30.206014	1506.391297	192.168.56.102	192.168.56.1	FTP	74	Response: 220 (vsFTPd 2.0.7)
81184	2010-03-08 12:46:30.206986	1506.392269	192.168.56.1	192.168.56.102	FTP	68	Request: USER ftpuser
81185	2010-03-08 12:46:30.207024	1506.392307	192.168.56.102	192.168.56.1	TCP	54	21→1515 [ACK] Seq=21 Ack=15 Win=5840 Len=0
81186	2010-03-08 12:46:30.233424	1506.418707	192.168.56.102	192.168.56.1	TCP	54	21→1514 [ACK] Seq=55 Ack=30 Win=5840 Len=0
81187	2010-03-08 12:46:30.240452	1506.425735	192.168.56.102	192.168.56.1	FTP	88	Response: 331 Please specify the password.
81188	2010-03-08 12:46:30.241576	1506.426859	192.168.56.1	192.168.56.102	FTP	69	Request: PASS eeeeeESM
81189	2010-03-08 12:46:30.281008	1506.466291	192.168.56.102	192.168.56.1	TCP	54	21→1515 [ACK] Seq=55 Ack=30 Win=5840 Len=0

Image 22: Password not cracked

Mitigation

In this network capture, there are two things to mitigate against. These are a port scan and a Brute Force password attack.

As covered in capture one of this assignment, port scans can be mitigated by using a network monitor set to alert the administrator for multiple connection requests from and to a specific machine.

Finally, any form of password attack can be mitigated by using the steps in the previous capture. The key to successfully mitigating a Brute Force type attack is to use longer passwords so the attacker must spend more time cycling through all of the possible combinations. As such, it would make their efforts much less valuable than the contents they are trying to access. A key point here is that all passwords can be cracked using a Brute Force attack so changing the password at regular intervals would greatly decrease the possibility of the attacker gaining the password.

Post Investigation Hash Values

Evidence Name	MD5 Hash Value	SHA1 Hash Value	SHA512 Hash Value
Networkcapture1.pcap	b834dbf7fad0d1e074529b03ad141110	050bb43df2c7e8c03ab920295370f79651733288	6526bfd37e7ed6e9643c9a96e49fc4e561371f292cafd8194ad7acc6adcabad6f1c9ecf3efc20871dc2e0c3e8f3d3e5004007203ecb6f5f3b87a67384f7a4178
Networkcapture2.pcap	744d49d0487352622d624837c2cf589c	b330894d9e5243cb806cef971e478233a9d39b7a	ed66c7b49cde415ad0e4bc7abedf3ba967cdfb39318374700cef73f22e5c9364dc6e8f6b552d921fb18993a19327d622abaed421576b263d9938a1afc36ee313
Networkcapture3.pcap	e12b2d244d23b9882301cd4ad52ed1de	b02de56eb4e6aac8a6944775be95bae259cf5483	b5cb9b447c2713833b617abe07649b895a027f46fd362e3e92fa3a5737ee2c08802d86cd8e7d1d4431800271b33cc179f22977144480917cb54f7506acd896b0

Table 8: Post Investigation Hash Values

Appendices

Appendix 1

ACPO Guidelines

"2. SECTION 2 – THE PRINCIPLES OF DIGITAL EVIDENCE

2.1 PRINCIPLES

***2.1.1 Principle 1**: No action taken by law enforcement agencies, persons employed within those agencies or their agents should change data which may subsequently be relied upon in court.*

***2.1.2 Principle 2**: In circumstances where a person finds it necessary to access original data, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions.*

***2.1.3 Principle 3**: An audit trail or other record of all processes applied to digital evidence should be created and preserved. An independent third party should be able to examine those processes and achieve the same result.*

***2.1.4 Principle 4**: The person in charge of the investigation has overall responsibility for ensuring that the law and these principles are adhered to." (Williams, 2012)*

Appendix 2

HTTP Page Requests

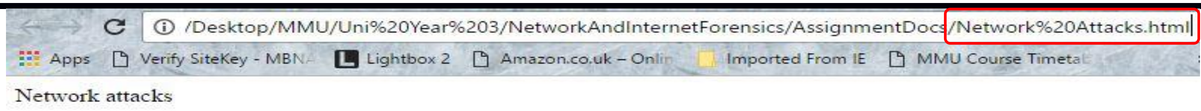
HTML Address	Contents of HTML Page
192.168.56.102/test.html	 file:///C:/Users/dastu_000/Desktop/MMU/Uni%20Year%203/NetworkAndInternetForensics/ExtractedObjects/test.html this is a test page
192.168.56.102/favicon.ico	Unable to display contents
192.168.56.102/Ducati.html	Unable to display contents
192.168.56.102/Dolphins.html	Unable to display contents
192.168.56.102/Checksums.html	 file:///C:/Users/dastu_000/Desktop/MMU/Uni%20Year%203/NetworkAndInternetForensics/ExtractedObjects/Checksums.html 4A89C678708ACCB6C5F0C88AD982CB1A717559536285156EEAF118E3CD78B5F2 6283F0B370504025848CC1177D7AE1733C7E4199A896AC8E5D69A98CD7DDCDDA E1E7B0DC917AB629EAAEF9AC5A650CE8E511DB87A914E24349C8F670986805A1 4327A8024BE399C944D0E1B229558FD29580BF6986352EC23F377A368CC5F79B 1A5D040BE4D0193FE8F6CDBECC0F4E38DD6ED3AC8E40497A07C49D975FF4C698 62A66A3E21BF2A91ADB5D3F3E61F57501A0A4B82189E1EAB3D06D21DA1A891C5 751D610E00B86E760530546D60C91EC01FA61E01AD5D0D1E24753A8B1CC72F25 86512CAD76131783F5DAE4346DDC3FB39F6F7C0F74B3039BFF70CA4015ADE034 AD9C159FB6286DCB1321D267EA4B218DBE41E9B74B28D93D1C9A9315EDB870A6 74A9ABF6B9298CAEA5BBBCF6FEC4CBBFA0D855BC7F417E8605050C948951C687 396B820F6136631A5E3FC966316738BF2E86163E6B1F58C4BF87117880FB46DF E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855 7168B0529FEFD9B9A0A491472AE02F2B1C4254081E93A38426FC5F19834A688F 03422544256768BE16DC4549DCFD3F7B2EC9503A9E6160A9508E669DF20E2CAF 18C64B430B1E0F6360B3AAA4DCD10160BAA4FC4E7557711BE3EC4060FE1AD1E8 AF33B55582C21EFAA594FA3FFADF30B7EB5B9AF1F548BB8ECCF5E64B26E81994 8317FC3994CD50ED7F35933611A41E30D62B85CF259E185C9BB4A05833FB3C08 2FDA11B4F44C9346A7F0EF922743E2664C0E4B0CCE86B5AA5FF0F71B3F111DC2 C04843949035B9641270A7650DDCBA03BE3872E190C315FEBB0AB0584B25C930
192.168.56.102/Network%20Attacks.html	 /Desktop/MMU/Uni%20Year%203/NetworkAndInternetForensics/AssignmentDocs/Network%20Attacks.html Apps Verify SiteKey - MBNA Lightbox 2 Amazon.co.uk – Onlin Imported From IE MMU Course Timetab Network attacks
192.168.56.102/Lena.html	Unable to display contents
192.168.56.102/HTML.html	 file:///C:/Users/dastu_000/Desktop/MMU/Uni%20Year%203/NetworkAndInternetForensics/ExtractedObjects/HTML.html This is a HTML webpage.

Table 9: HTTP Requests Made in the Captures

Appendix 3

TCP Three-Way Handshake

"The TCP three-way handshake in Transmission Control Protocol (also called the TCP-handshake; three message handshake and/or SYN-SYN-ACK) is the method used by TCP set up a TCP/IP connection over an Internet Protocol based network. TCP's three way handshaking technique is often referred to as "SYN-SYN-ACK" (or more accurately SYN, SYN-ACK, ACK) because there are three messages transmitted by TCP to negotiate and start a TCP session between two computers." (Inetdaemon.com, 2016)

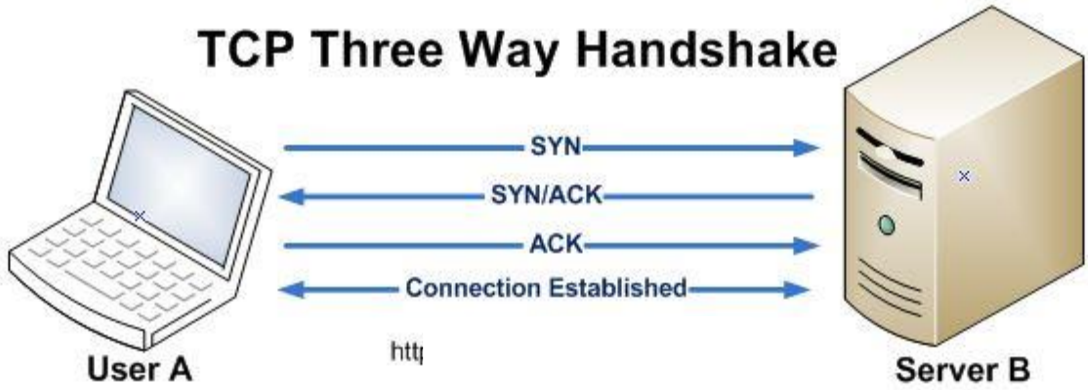


Image 23: TCP 3-way handshake

Appendix 4

IGMPv3

"Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) are the Multicast Group Membership Discovery (MGMD) protocols. They are essentially the same protocol, with IGMP used for IPv4 multicast groups and MLD used for IPv6 multicast groups. These protocols are used between end systems (often desktops) and the multicast router to request data for a given multicast group." (Metaswitch.com, 2016)

Appendix 5

MDNS

"Multicast DNS is a joint effort by participants of the IETF Zero Configuration Networking (zeroconf) and DNS Extensions (dnsext) working groups. The requirements are driven by the Zeroconf working group; the implementation details are a chartered work item for the DNSEXT group. Most of the people working on mDNS are active participants of both working groups. While the requirements for Zeroconf name resolution could be met by designing an entirely new protocol, it is better to provide this functionality by making minimal changes to the current standard DNS protocol. This saves application programmers from having to learn new APIs, and saves application programmers from having to write application code two different ways — one way for large configured networks and a different way for small Zeroconf networks. It means that most current applications need no changes at all to work correctly using mDNS in a Zeroconf network. It also means that engineers do not have to learn an entirely new protocol, and current network packet capture tools can already decode and display DNS packets, so they do not have to be updated to understand new packet formats." (Cheshire, 2016)

Appendix 6

HTTP

"HTTP allows for communication between a variety of hosts and clients, and supports a mixture of network configurations. To make this possible, it assumes very little about a particular system, and does not keep state between different message exchanges. This makes HTTP a stateless protocol. The communication usually takes place over TCP/IP, but any reliable transport can be used. The default port for TCP/IP is 80, but other ports can also be used. Custom headers can also be created and sent by the client. Communication between a host and a client occurs, via a request/response pair. The client

initiates an HTTP request message, which is serviced through a HTTP response message in return…………

………… URLs reveal the identity of the particular host with which we want to communicate, but the action that should be performed on the host is specified via HTTP verbs. Of course, there are several actions that a client would like the host to perform. HTTP has formalized on a few that capture the essentials that are universally applicable for all kinds of applications.
These request verbs are:
GET: fetch an existing resource. The URL contains all the necessary information the server needs to locate and return the resource.
POST: create a new resource. POST requests usually carry a payload that specifies the data for the new resource.
PUT: update an existing resource. The payload may contain the updated data for the resource.
DELETE: delete an existing resource." (Code Envato Tuts+, 2016)

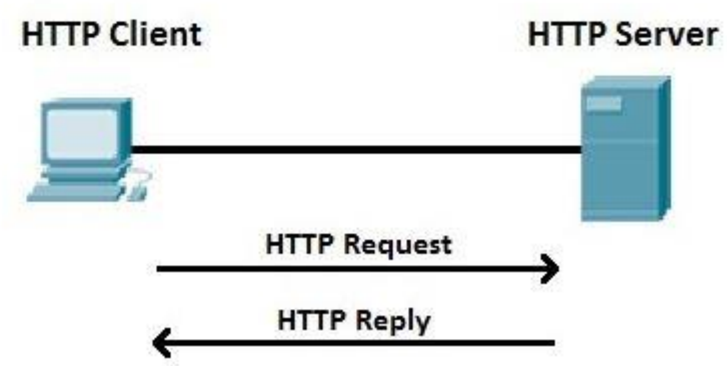


Image 24: HTTP Request and Response

Appendix 7

<u>Port 21 & Passive FTP</u>

The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among hosts, and 4) to transfer data reliably and efficiently. FTP, though usable directly by a user at a terminal, is designed mainly for use by programs. (Postel and Reynolds, 1985)

Appendix 8

<u>Dictionary Password Attack</u>

"A dictionary attack is a method of breaking into a password-protected computer or server by systematically entering every word in a dictionary as a password." (SearchSecurity, 2016)
The dictionary that is used can be obtained from anywhere and many of these dictionaries are readily available online for anyone to acquire.

Appendix 9

Password Hashing

“The general workflow for account registration and authentication in a hash-based account system is as follows: The user creates an account. Their password is hashed and stored in the database. At no point is the plain-text (unencrypted) password ever written to the hard drive. When the user attempts to login, the hash of the password they entered is checked against the hash of their real password (retrieved from the database). If the hashes match, the user is granted access. If not, the user is told they entered invalid login credentials. Steps 3 and 4 repeat every time someone tries to login to their account. In step 4, never tell the user if it was the username or password they got wrong. Always display a generic message like "Invalid username or password." This prevents attackers from enumerating valid usernames without knowing their passwords.” (Hornby, 2016)

Appendix 10

Brute Force Password Attack

“Just as a criminal might break into, or "crack" a safe by trying many possible combinations, a brute force cracking application proceeds through all possible combinations of legal characters in sequence. Brute force is considered to be an infallible, although time-consuming, approach.”(SearchSecurity, 2016)

Appendix 11

Python Script for Pcap Visualisation

```
#################################################
#         Dale Stubbs - 14024149                        #
#  This script is designed to read in the                #
#  Network Captures from the coursework.                 #
#  It gathers the information from the                   #
#  capture and plots the information on                  #
#  a graph using MatPlotLib.                             #
#  Last Edited - 21/11/2016                              #
#################################################
import dpkt
from dpkt.tcp import TCP
import sys
import matplotlib.pyplot as plt
import socket
#  Plots a graph based on the port number and time stamp.
def plot_graph1(ports, packets, in_ip):
	f = plt.figure(1)
	plt.plot(packets, ports, 'b-')
	plt.title("14024149 \n Port Scan")
	plt.ylabel('Port Number')
	plt.xlabel('Time Stamp of Packet (Aggressor IP: ' + in_ip + ')')
	f.show()
#  Plots a graph based on the incorrect password attempts and time stamp.
def plot_graph2(attempts, packets, in_ip):
	g = plt.figure(2)
	plt.plot(packets, attempts, 'b-')
	plt.title("14024149\n Password Attack")
	plt.ylabel('Number of Incorrect Password Attempts')
	plt.xlabel('Time Stamp of Each Attempt (Aggressor IP: ' + in_ip + ')')
	g.show()

def main():
	x = 0
	while x != 1:
		if len(sys.argv) < 3:
			print 'Usage: python NetCapAnalysis.py [Capture File] [IP of
Interest/Aggressor IP]'
			sys.exit()
		else:
			capture = sys.argv[1]
			in_ip = sys.argv[2]
			x = 1
	#  Reads the PCAP file in using the 'read binary' command
	f = open(capture, 'rb')
	pcap = dpkt.pcap.Reader(f)
	count = 1
	ports = []
	packets1 = []
	packets2 = []
	attempts = []
	#  Loops through each packet in the capture
```

```
        for ts, buf in pcap:
                eth = dpkt.ethernet.Ethernet(buf)
                ip = eth.data
                tcp = ip.data
                #  Filters the traffic based on the IP address
                #print 'Works'
                if type(ip.data) == TCP:
                        source_ip = socket.inet_ntoa(ip.src)
                        if source_ip == in_ip:
                                #  Appends the Port Number and time stamp to the
                                #  appropriate lists
                                packets1.append(ts)
                                ports.append(tcp.dport)
                                #  Filters traffic again based on the '530' response code
                                #  being present in the TCP Data.
                                #  530 is the code for 'Incorrect Login'

                                if 'PASS' in str(tcp.data):
                                        packets2.append(ts)
                                        attempts.append(count)
                                        count += 1
        #  Check to see if the attack is a port scan or password attack.
        with open('tcp.dport.txt', 'w') as q:
                for line in ports:
                        q.write(str(line) + '\n')
        q.close()
        if '3' in capture:
                plot_graph1(ports, packets1, in_ip)
                plot_graph2(attempts, packets2, in_ip)
                raw_input()
        else:
                if len(attempts) < 10:
                        plot_graph1(ports, packets1, in_ip)
                        raw_input()
                else:
                        plot_graph2(attempts, packets2, in_ip)
                        raw_input()
if __name__ == '__main__':
  main()
```

Appendix 12

Images Created Using the Above Script

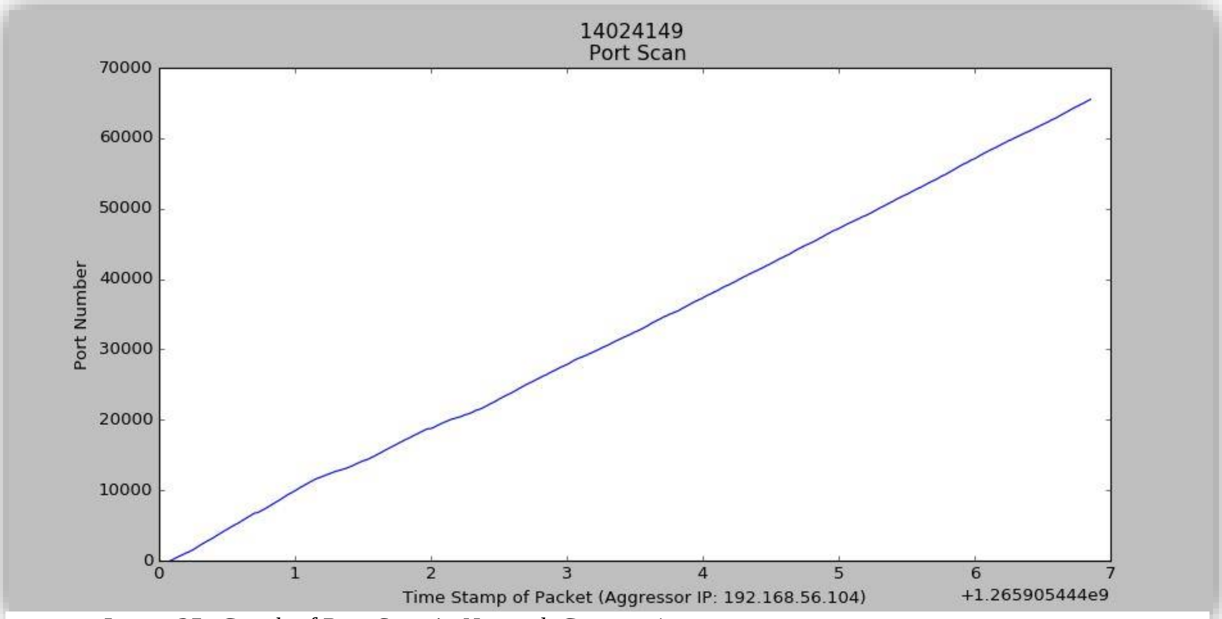


Image 25: Graph of Port Scan in Network Capture 1

The graph above show the port numbers and the timestamp of each packet within the NetworkCapture1 pcap file. The surge from port 1 to port 65,534 shows a sequential increase of the port numbers in a very short space of time leading to the conclusion that a sequential port scan being completed within the pcap file.

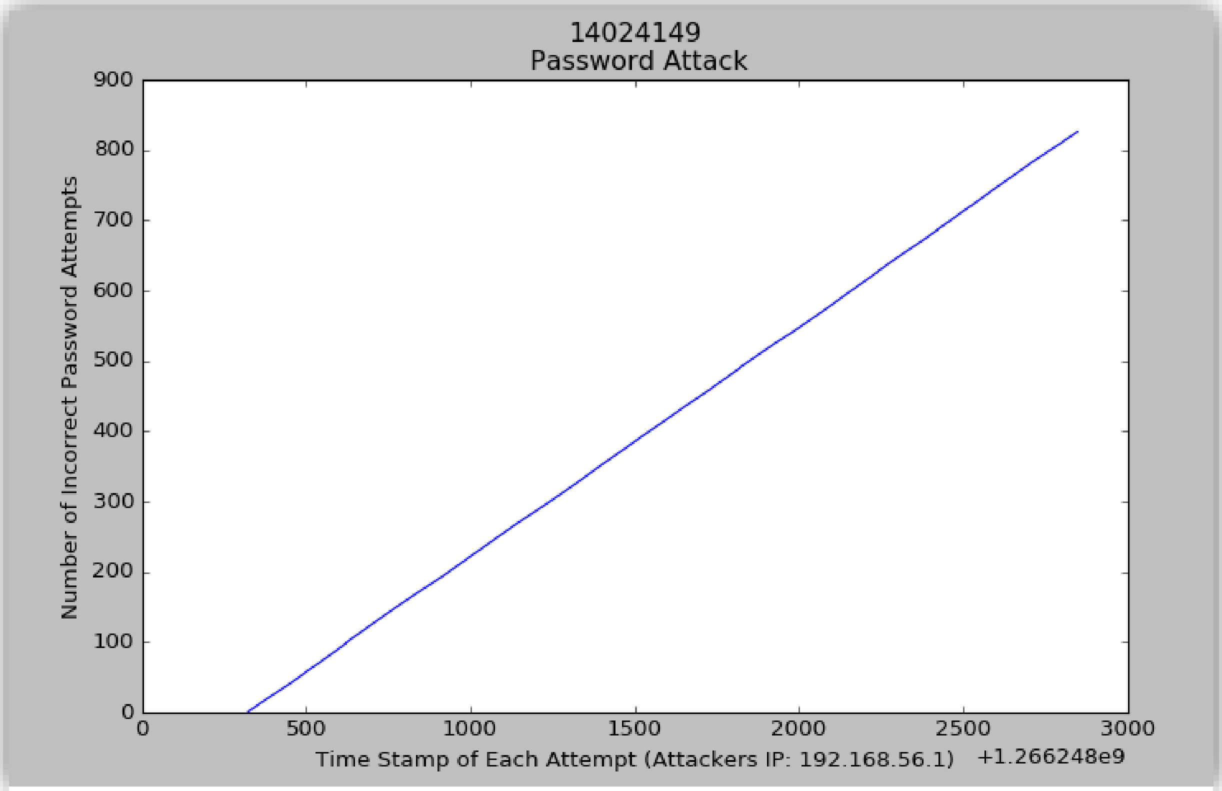


Image 26: Graph of Password Attack in Network Capture 2

The graph above shows the number of attempts made to input a password showing a password attack within the NetworkCapture2 pcap file. Having almost 900 incorrect password entries within the file in such a small amount of time would suggest that a password attack is occurring.

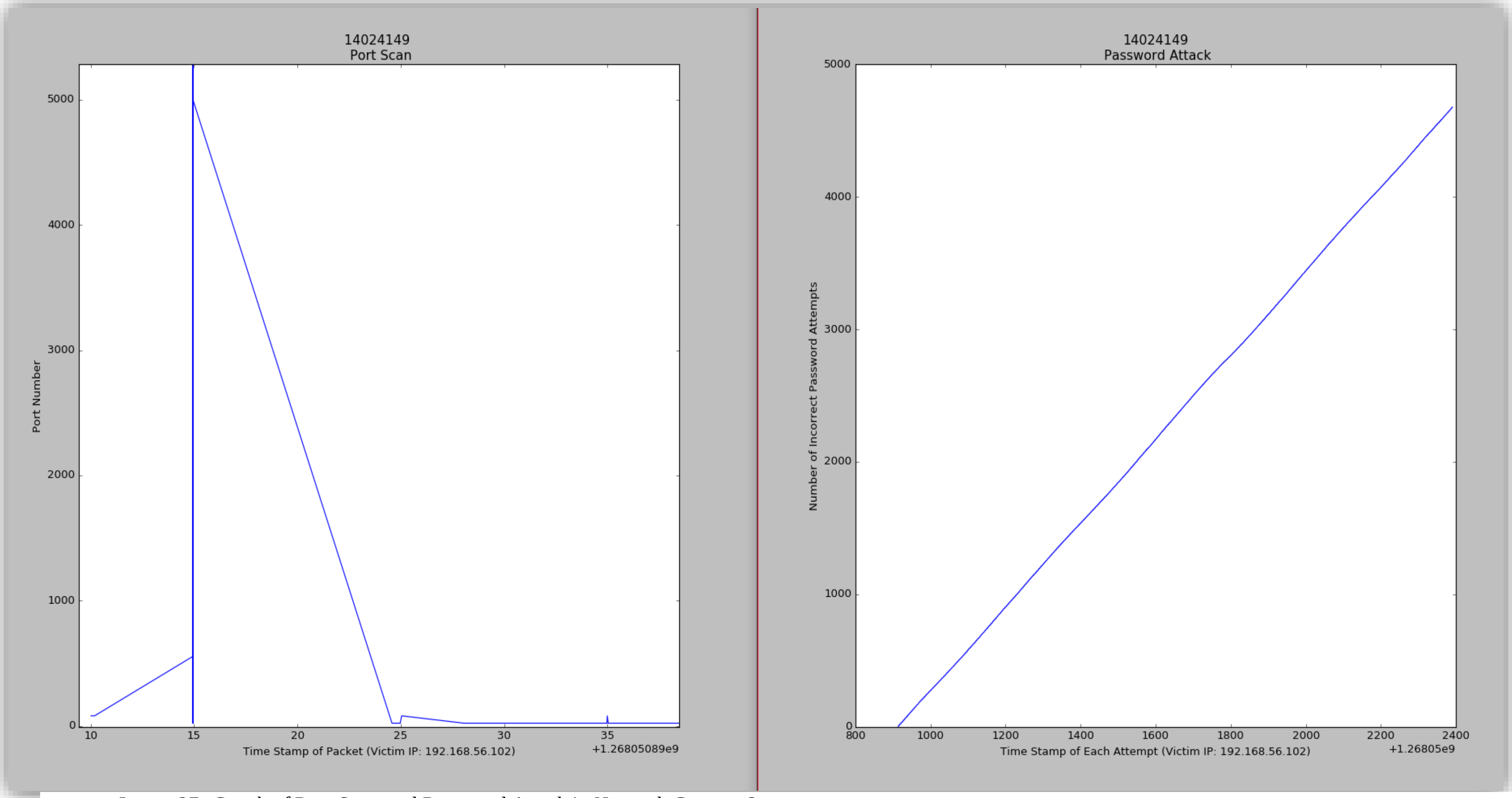


Image 27: Graph of Port Scan and Password Attack in Network Capture 3

Although the graph on the left doesn't look like a typical port scan, due the micro timings between the timestamps the significant changes are not visible however, this is confirmed as a port scan due to the port numbers being accessed in such a small time scale. The graph on the right show the number of attempts made to input a password showing a password attack within the NetworkCapture3 pcap file.

References

Msdn.microsoft.com. (2016). Ensuring Data Integrity with Hash Codes. [online] Available at: https://msdn.microsoft.com/en-us/library/f9ax34y5(v=vs.110).aspx [Accessed 5 Nov. 2016].

Williams, J. (2012). ACPO Good Practice Guide. 5th ed. [ebook] Metropolitan Police Service, p.6. Available at: http://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf [Accessed 5 Nov. 2016].

Lifewire. (2016). *Wondering How Port Scanning Works? Here's the Answer*. [online] Available at: https://www.lifewire.com/introduction-to-port-scanning-2486802 [Accessed 30 Oct. 2016].

Inetdaemon.com. (2016). TCP 3-Way Handshake (SYN,SYN-ACK,ACK) - InetDaemon's IT Tutorials. [online] Available at: http://www.inetdaemon.com/tutorials/internet/tcp/3-way_handshake.shtml [Accessed 12 Dec. 2016].

Metaswitch.com. (2016). *What is Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)?*. [online] Available at: http://www.metaswitch.com/resources/what-is-internet-group-management-protocol-igmp-multicast-listener-discovery-mld [Accessed 30 Oct. 2016].

Cheshire, S. (2016). Multicast DNS. [online] Multicastdns.org. Available at: http://www.multicastdns.org/ [Accessed 30 Oct. 2016].

Code Envato Tuts+. (2016). HTTP: The Protocol Every Web Developer Must Know - Part 1. [online] Available at: https://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177 [Accessed 30 Oct. 2016].

Collider, S. (2016). How Secure Is My Password?. [online] Howsecureismypassword.net. Available at: https://howsecureismypassword.net/ [Accessed 5 Nov. 2016].

Postel, J. and Reynolds, J. (1985). FILE TRANSFER PROTOCOL (FTP). 1st ed. [ebook] Available at: https://tools.ietf.org/pdf/rfc959.pdf [Accessed 12 Dec. 2016].

SearchSecurity. (2016). What is dictionary attack? - Definition from WhatIs.com. [online] Available at: http://searchsecurity.techtarget.com/definition/dictionary-attack [Accessed 5 Nov. 2016].

Hornby, T. (2016). Salted Password Hashing - Doing it Right - CodeProject. [online] Codeproject.com. Available at: http://www.codeproject.com/Articles/704865/Salted-Password-Hashing-Doing-it-Right [Accessed 5 Nov. 2016].

SearchSecurity. (2016). What is brute force cracking? - Definition from WhatIs.com. [online] Available at: http://searchsecurity.techtarget.com/definition/brute-force-cracking [Accessed 7 Nov. 2016].