CA169 Networks Assignment Two

Answer Sheets

STUDENT NAME:	Andrew Finn
STUDENT NUMBER:	[redacted]
PROJECT NUMBER:	2
MODULE CODE:	CA169
DEGREE: {CA|EC|CPSSD|ECSA]	CA
LECTURER:	Brian Stone

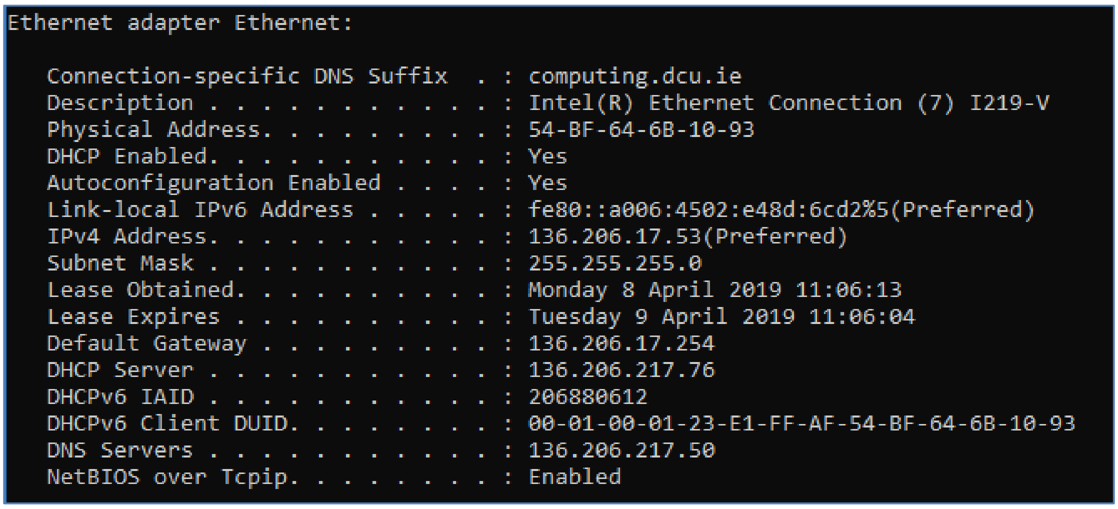
Declaration

In submitting this project, I declare that the project material, which I now submit, is my own work. Any assistance received by way of borrowing from the work of others has been cited and acknowledged within the work. I make this declaration in the knowledge that a breach of the rules pertaining to project submission may carry serious consequences.

Part 1: DHCP traffic

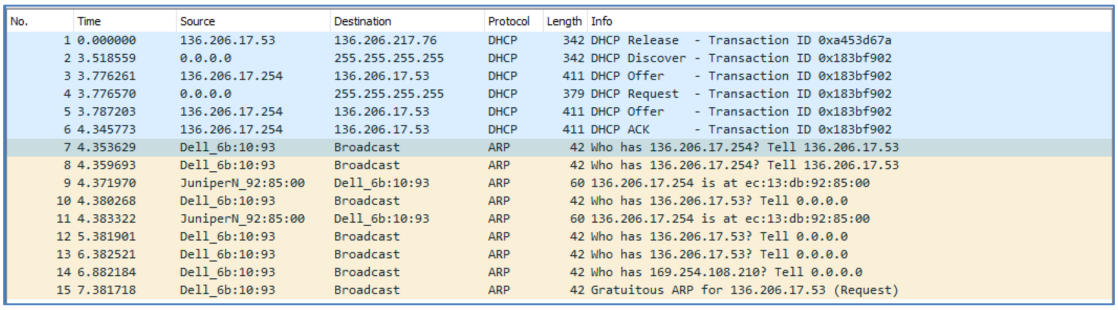
Your IP & MAC address for this experiment (use ipconfig)

136.206.17.53	*54-BF-64-6B-10-93*



```
Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : computing.dcu.ie
   Description . . . . . . . . . . . : Intel(R) Ethernet Connection (7) I219-V
   Physical Address. . . . . . . . . : 54-BF-64-6B-10-93
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::a006:4502:e48d:6cd2%5(Preferred)
   IPv4 Address. . . . . . . . . . . : 136.206.17.53(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Monday 8 April 2019 11:06:13
   Lease Expires . . . . . . . . . . : Tuesday 9 April 2019 11:06:04
   Default Gateway . . . . . . . . . : 136.206.17.254
   DHCP Server . . . . . . . . . . . : 136.206.217.76
   DHCPv6 IAID . . . . . . . . . . . : 206880612
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-23-E1-FF-AF-54-BF-64-6B-10-93
   DNS Servers . . . . . . . . . . . : 136.206.217.50
   NetBIOS over Tcpip. . . . . . . . : Enabled
```



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	136.206.17.53	136.206.217.76	DHCP	342	DHCP Release - Transaction ID 0xa453d67a
2	3.518559	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x183bf902
3	3.776261	136.206.17.254	136.206.17.53	DHCP	411	DHCP Offer - Transaction ID 0x183bf902
4	3.776570	0.0.0.0	255.255.255.255	DHCP	379	DHCP Request - Transaction ID 0x183bf902
5	3.787203	136.206.17.254	136.206.17.53	DHCP	411	DHCP Offer - Transaction ID 0x183bf902
6	4.345773	136.206.17.254	136.206.17.53	DHCP	411	DHCP ACK - Transaction ID 0x183bf902
7	4.353629	Dell_6b:10:93	Broadcast	ARP	42	Who has 136.206.17.254? Tell 136.206.17.53
8	4.359693	Dell_6b:10:93	Broadcast	ARP	42	Who has 136.206.17.254? Tell 136.206.17.53
9	4.371970	JuniperN_92:85:00	Dell_6b:10:93	ARP	60	136.206.17.254 is at ec:13:db:92:85:00
10	4.380268	Dell_6b:10:93	Broadcast	ARP	42	Who has 136.206.17.53? Tell 0.0.0.0
11	4.383322	JuniperN_92:85:00	Dell_6b:10:93	ARP	60	136.206.17.254 is at ec:13:db:92:85:00
12	5.381901	Dell_6b:10:93	Broadcast	ARP	42	Who has 136.206.17.53? Tell 0.0.0.0
13	6.382521	Dell_6b:10:93	Broadcast	ARP	42	Who has 136.206.17.53? Tell 0.0.0.0
14	6.882184	Dell_6b:10:93	Broadcast	ARP	42	Who has 169.254.108.210? Tell 0.0.0.0
15	7.381718	Dell_6b:10:93	Broadcast	ARP	42	Gratuitous ARP for 136.206.17.53 (Request)

Packet numbers relevant to the DHCP interaction:

a. **DHCP DISCOVER –** Packet 2
b. **DHCP OFFER –** Packet 3,5
c. **DHCP Request –** Packet 4
d. **DHCP Acknowledgement –** Packet 6
e. **DHCP Release (if you release using `ipconfig /release`) –** Packet 1
f. **All ARP packets used -** < 8 Packets

Function of each packet

a. **DHCP DISCOVER –** This is the initial DHCP packet being sent from the client across the entire network. The packet contains details such as its hostname and MAC address. The goal of this packet is to find the DHCP server is order to obtain an IP address.

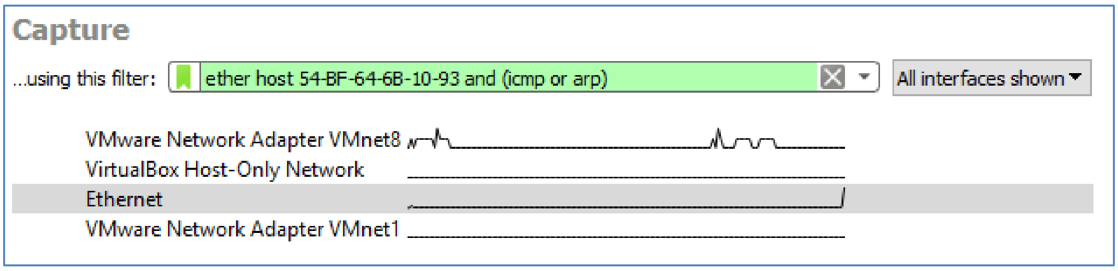
b. **DHCP OFFER –** This is the reply to the Discover request. This reply is sent from the DHCP server offering the client an IP address
c. **DHCP Request –** This packet is sent from the client in response to the Offer. This is the client accepting the IP address.
d. **DHCP Acknowledgement –** These packets are send from the DHCP server they are acknowledgements of the clients requests and specify the length in seconds of the lease.
e. **DHCP Release (if you release using `ipconfig /release`) –** This packet is sent from the client it is a packet that notifies the server that the IP address is no longer associated with the client. This is so the server can assign to IP address to other clients.
f. **ARP –** These packets are ARP both request and replies from several clients across the network. They are broadcast across the network they contain the senders IP and MAC and the message. The message is questioning all clients by asking what MAC address has been assigned a given IP address. There are several types of ARP packets such as the one in Packet 15 which is broadcasting the clients new IP/MAC address mapping rather than having other clients having to request it.

Part 2: `ping` *traffic*

Your IP & MAC address for this experiment (use ipconfig)

136.206.17.53	54-BF-64-6B-10-93

Screen capture of Wireshark filter utilised.

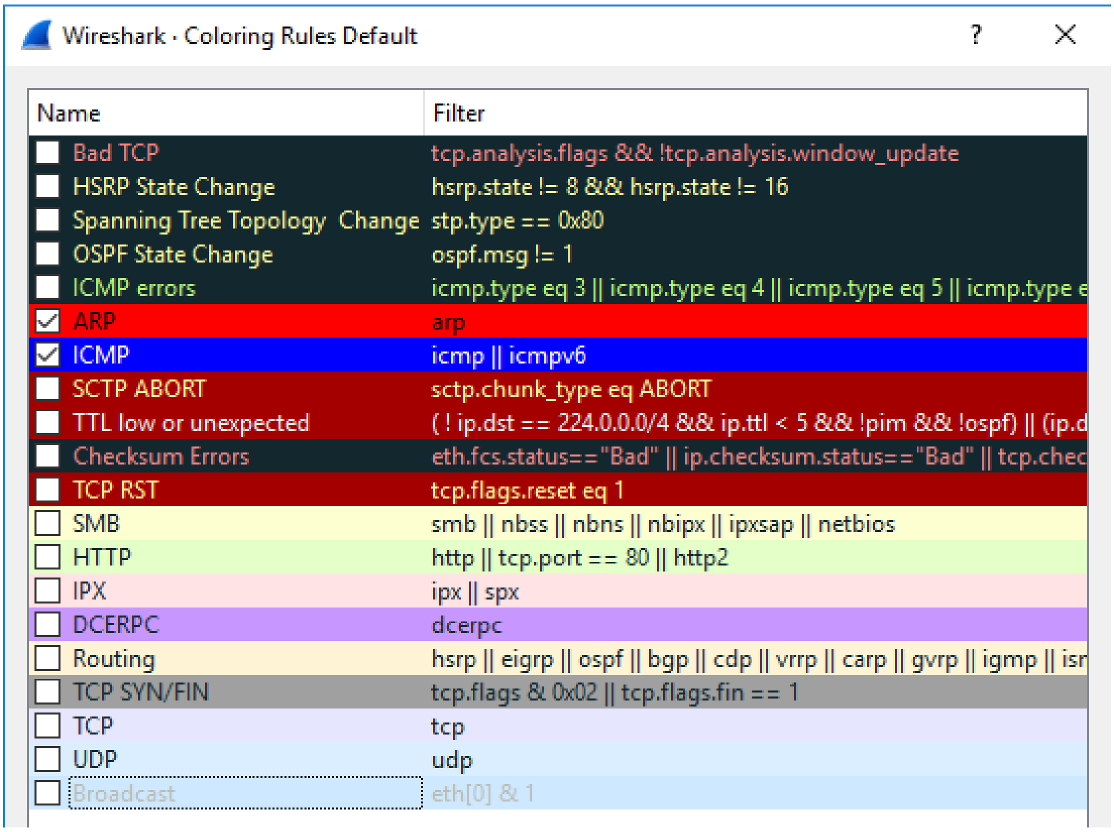


Capture

...using this filter: ether host 54-BF-64-6B-10-93 and (icmp or arp)

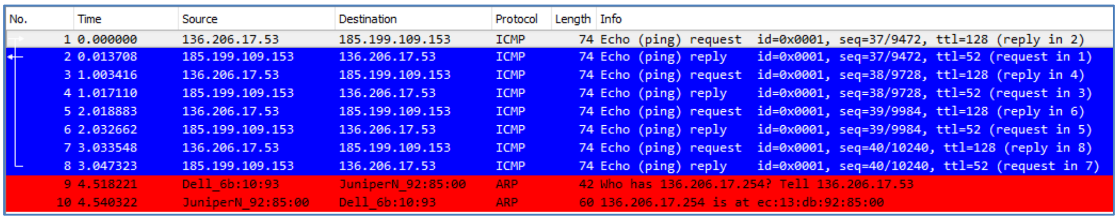
VMware Network Adapter VMnet8
VirtualBox Host-Only Network
Ethernet
VMware Network Adapter VMnet1

Screen capture of Wireshark colouring rules applied



Name	Filter
□ Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update
□ HSRP State Change	hsrp.state != 8 && hsrp.state != 16
□ Spanning Tree Topology Change	stp.type == 0x80
□ OSPF State Change	ospf.msg != 1
□ ICMP errors	icmp.type eq 3 || icmp.type eq 4 || icmp.type eq 5 || icmp.type e
☑ ARP	arp
☑ ICMP	icmp || icmpv6
□ SCTP ABORT	sctp.chunk_type eq ABORT
□ TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) || (ip.d
□ Checksum Errors	eth.fcs.status=="Bad" || ip.checksum.status=="Bad" || tcp.chec
□ TCP RST	tcp.flags.reset eq 1
□ SMB	smb || nbss || nbns || nbipx || ipxsap || netbios
□ HTTP	http || tcp.port == 80 || http2
□ IPX	ipx || spx
□ DCERPC	dcerpc
□ Routing	hsrp || eigrp || ospf || bgp || cdp || vrrp || carp || gvrp || igmp || isr
□ TCP SYN/FIN	tcp.flags & 0x02 || tcp.flags.fin == 1
□ TCP	tcp
□ UDP	udp
□ Broadcast	eth[0] & 1

Screen capture of Wireshark packet trace showing all relevant ping generated traffic, including ARP and ICMP traffic.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	136.206.17.53	185.199.109.153	ICMP	74	Echo (ping) request id=0x0001, seq=37/9472, ttl=128 (reply in 2)
2	0.013708	185.199.109.153	136.206.17.53	ICMP	74	Echo (ping) reply id=0x0001, seq=37/9472, ttl=52 (request in 1)
3	1.003416	136.206.17.53	185.199.109.153	ICMP	74	Echo (ping) request id=0x0001, seq=38/9728, ttl=128 (reply in 4)
4	1.017110	185.199.109.153	136.206.17.53	ICMP	74	Echo (ping) reply id=0x0001, seq=38/9728, ttl=52 (request in 3)
5	2.018883	136.206.17.53	185.199.109.153	ICMP	74	Echo (ping) request id=0x0001, seq=39/9984, ttl=128 (reply in 6)
6	2.032662	185.199.109.153	136.206.17.53	ICMP	74	Echo (ping) reply id=0x0001, seq=39/9984, ttl=52 (request in 5)
7	3.033548	136.206.17.53	185.199.109.153	ICMP	74	Echo (ping) request id=0x0001, seq=40/10240, ttl=128 (reply in 8)
8	3.047323	185.199.109.153	136.206.17.53	ICMP	74	Echo (ping) reply id=0x0001, seq=40/10240, ttl=52 (request in 7)
9	4.518221	Dell_6b:10:93	JuniperN_92:85:00	ARP	42	Who has 136.206.17.254? Tell 136.206.17.53
10	4.540322	JuniperN_92:85:00	Dell_6b:10:93	ARP	60	136.206.17.254 is at ec:13:db:92:85:00

.

Packet 1:
This is the first IMCP Echo request. It was sent after I pinged afinn.me (personal website). Its payload is 32 bytes and the packet it 74 bytes long in length. This IMCP packet is a method to ensure both clients are able to communicate and understand each other. The packet contains the next destination address for the packet (router). The packet also contains a time to live counter or TTL. Each time the packet reaches a router this counter is reduced by one. If this counter reaches 0 this will indicate the packet did not reach its target. This prevents and infinite loop if a target cannot be found. This process is the same for packets 1,3,5 and 7.

Packet 2:
This is the IMCP echo reply. This was send from the host server of afinn.me (GitHub). It was send in reply of Packet 1. This packet confirms to the client that the server received the IMCP request and confirms to the client that it is able to communicate using IMCP. This process is the same for packets 2,5,6 and 8

Packet 9:
This is a ARP request of length 42 (bytes). This request is broadcast by the client across the network. It is a request to see what is the Physical/MAC address of 136.206.17.254. It contains the clients IP and MAC address and the address it wishes to communicate with.

Packet 10:
This is the ARP reply is response to Packet 9 send by the router. This allows the client and router to communicate with each other as they now know each other MAC and IP address as well as a protocol to which they can communicate. This packet contains the routers IP and MAC address.

Part 3:

Your IP & MAC address for this experiment (use ipconfig)

136.206.17.53	54-BF-64-6B-10-93

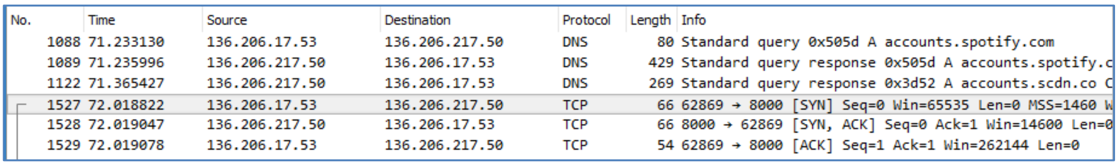
Filter to show only traffic concerning the test machine

Filter	tcp.stream eq 39 or dns contains “spotify”

Explain how you found the start of the interaction between your PC and the website.

I found the interaction between my PC and the website (Spotify) by flushing my DNS this removed the DNS cache to ensure I made a query to the DNS server. I then loaded www.spotify.com and used the follow option on a TCP that was part of a the 3 way handshake to identify eq 39. I then used the above filter to filter out packets that were not related to Spotify.

Wireshark window showing the start of the interaction (should show ARP, DNS and TCP 3-way handshake)



No.	Time	Source	Destination	Protocol	Length	Info
1088	71.233130	136.206.17.53	136.206.217.50	DNS	80	Standard query 0x505d A accounts.spotify.com
1089	71.235996	136.206.217.50	136.206.17.53	DNS	429	Standard query response 0x505d A accounts.spotify.c
1122	71.365427	136.206.217.50	136.206.17.53	DNS	269	Standard query response 0x3d52 A accounts.scdn.co C
1527	72.018822	136.206.17.53	136.206.217.50	TCP	66	62869 → 8000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 W
1528	72.019047	136.206.217.50	136.206.17.53	TCP	66	8000 → 62869 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
1529	72.019078	136.206.17.53	136.206.217.50	TCP	54	62869 → 8000 [ACK] Seq=1 Ack=1 Win=262144 Len=0

Write down the numbers of the packets with the 3-way handshake. Explain what is happening with these 3 packets.

Packet 1527 [SYN]: The SYN packet is sent by the client to the server. This packet is a message to the server to check if it is open for new connections. It also contains a sequence number in this case 0.

Packet 1528 [SYN, ATK]: This is the server response to Packet 1527. It acts as a sort of authorization telling the client that it is open for new connections and that it is free to try to connect. It also has a ACK number which is 1.

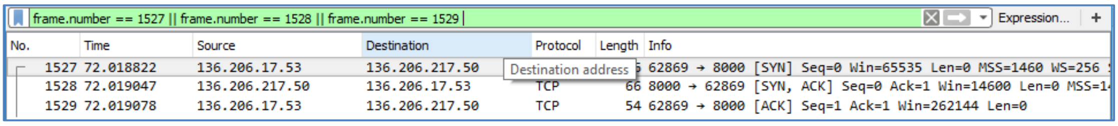
Packet 1529 [ACK]: When the client received Packet 1528 it responded with this ACK packet. This established a connection. This then iterates on the ACK variable by 1.

The ACK and SEQ variable are a method of verification to ensure each message is sent and received as the response should be +1 of the message sent/revived.

Write down a filter to show only these three-way-handshake packets

Filter	frame.number == 1527 || frame.number == 1528 || frame.number == 1529

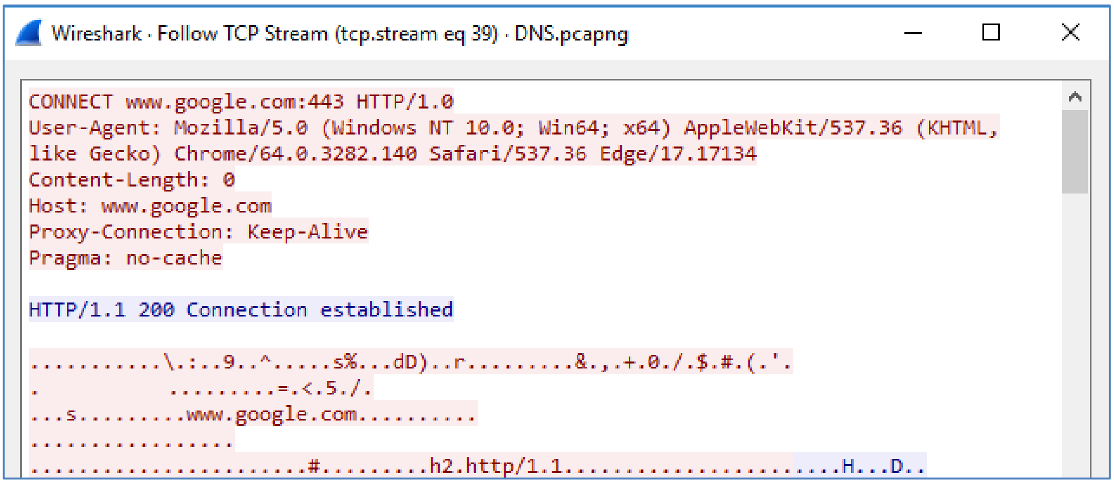
Wireshark window for the 3-way-handshake



```
frame.number == 1527 || frame.number == 1528 || frame.number == 1529

No.    Time       Source          Destination      Protocol  Length  Info
  1527 72.018822  136.206.17.53   136.206.217.50   Destination address 5 62869 → 8000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 
  1528 72.019047  136.206.217.50  136.206.17.53    TCP       66 8000 → 62869 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1
  1529 72.019078  136.206.17.53   136.206.217.50   TCP       54 62869 → 8000 [ACK] Seq=1 Ack=1 Win=262144 Len=0
```

Show the <u>Follow TCP Stream</u> window here.



Wireshark · Follow TCP Stream (tcp.stream eq 39) · DNS.pcapng

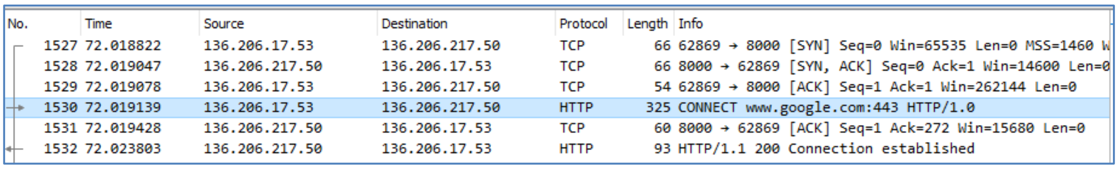
```
CONNECT www.google.com:443 HTTP/1.0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134
Content-Length: 0
Host: www.google.com
Proxy-Connection: Keep-Alive
Pragma: no-cache

HTTP/1.1 200 Connection established

...........\.:..9..^.....s%...dD)..r.........&.,.+.0./.$.#.(.'.
.          .........=.<.5./.
...s.........www.google.com..........
.................
.......................#.........h2.http/1.1......................H...D..
```

Your notes on…

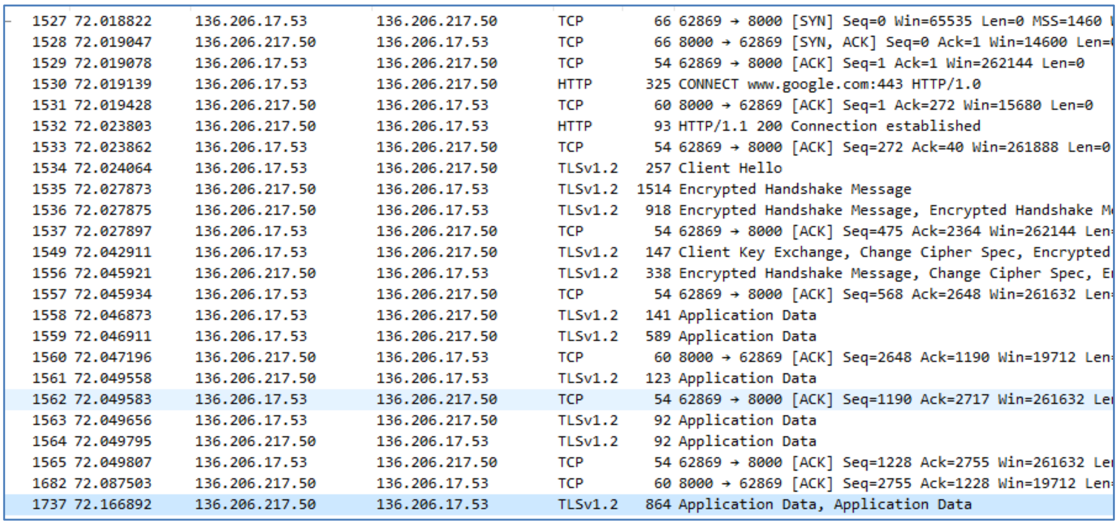
a. The GET requests made



```
No.    Time       Source          Destination      Protocol  Length  Info
  1527 72.018822  136.206.17.53   136.206.217.50   TCP        66 62869 → 8000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 W
  1528 72.019047  136.206.217.50  136.206.17.53    TCP        66 8000 → 62869 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
  1529 72.019078  136.206.17.53   136.206.217.50   TCP        54 62869 → 8000 [ACK] Seq=1 Ack=1 Win=262144 Len=0
  1530 72.019139  136.206.17.53   136.206.217.50   HTTP      325 CONNECT www.google.com:443 HTTP/1.0
  1531 72.019428  136.206.217.50  136.206.17.53    TCP        60 8000 → 62869 [ACK] Seq=1 Ack=272 Win=15680 Len=0
  1532 72.023803  136.206.217.50  136.206.17.53    HTTP       93 HTTP/1.1 200 Connection established
```

As all websites should Spotify.com uses HTTPS encryption and as such we can’t sniff for a “GET” request as it is encrypted. We however can see a “CONNECT” request which is verified at Packet 1532.

b. The responses from the server



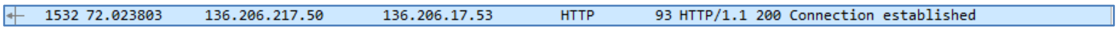
```
1527 72.018822    136.206.17.53     136.206.217.50    TCP       66 62869 → 8000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
1528 72.019047    136.206.217.50    136.206.17.53     TCP       66 8000 → 62869 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=
1529 72.019078    136.206.17.53     136.206.217.50    TCP       54 62869 → 8000 [ACK] Seq=1 Ack=1 Win=262144 Len=0
1530 72.019139    136.206.17.53     136.206.217.50    HTTP     325 CONNECT www.google.com:443 HTTP/1.0
1531 72.019428    136.206.217.50    136.206.17.53     TCP       60 8000 → 62869 [ACK] Seq=1 Ack=272 Win=15680 Len=0
1532 72.023803    136.206.217.50    136.206.17.53     HTTP      93 HTTP/1.1 200 Connection established
1533 72.023862    136.206.17.53     136.206.217.50    TCP       54 62869 → 8000 [ACK] Seq=272 Ack=40 Win=261888 Len=0
1534 72.024064    136.206.17.53     136.206.217.50    TLSv1.2  257 Client Hello
1535 72.027873    136.206.217.50    136.206.17.53     TLSv1.2 1514 Encrypted Handshake Message
1536 72.027875    136.206.217.50    136.206.17.53     TLSv1.2  918 Encrypted Handshake Message, Encrypted Handshake M
1537 72.027897    136.206.17.53     136.206.217.50    TCP       54 62869 → 8000 [ACK] Seq=475 Ack=2364 Win=262144 Len
1549 72.042911    136.206.17.53     136.206.217.50    TLSv1.2  147 Client Key Exchange, Change Cipher Spec, Encrypted
1556 72.045921    136.206.217.50    136.206.17.53     TLSv1.2  338 Encrypted Handshake Message, Change Cipher Spec, E
1557 72.045934    136.206.17.53     136.206.217.50    TCP       54 62869 → 8000 [ACK] Seq=568 Ack=2648 Win=261632 Len
1558 72.046873    136.206.17.53     136.206.217.50    TLSv1.2  141 Application Data
1559 72.046911    136.206.17.53     136.206.217.50    TLSv1.2  589 Application Data
1560 72.047196    136.206.217.50    136.206.17.53     TCP       60 8000 → 62869 [ACK] Seq=2648 Ack=1190 Win=19712 Len
1561 72.049558    136.206.217.50    136.206.17.53     TLSv1.2  123 Application Data
1562 72.049583    136.206.17.53     136.206.217.50    TCP       54 62869 → 8000 [ACK] Seq=1190 Ack=2717 Win=261632 Le
1563 72.049656    136.206.17.53     136.206.217.50    TLSv1.2   92 Application Data
1564 72.049795    136.206.217.50    136.206.17.53     TLSv1.2   92 Application Data
1565 72.049807    136.206.17.53     136.206.217.50    TCP       54 62869 → 8000 [ACK] Seq=1228 Ack=2755 Win=261632 Le
1682 72.087503    136.206.217.50    136.206.17.53     TCP       60 8000 → 62869 [ACK] Seq=2755 Ack=1228 Win=19712 Len
1737 72.166892    136.206.217.50    136.206.17.53     TLSv1.2  864 Application Data, Application Data
```

Packet 1532: Initial response from the server “HTTP/1.1 200 Connection established”. This is telling the client that a secure connection has been made.

Packet 1535 and 1536: As mentioned earlier Spotify uses the HTTPs standard which uses TSL encryption. These packets are a verification that this has been established correctly. The client is encrypting a predefined known message and the server is decrypting. This is to ensure TSL has been established correctly.

Packets > 1536: These are encrypted “application data” between the server and client.

c. The HTTP response codes used in the interaction and what they mean (look them up yourself on the Web)



```
1532 72.023803    136.206.217.50    136.206.17.53     HTTP      93 HTTP/1.1 200 Connection established
```

As explained earlier ‘HTTP/1.1 200 Connection established”. This is telling the client that a secure connection has been made.’ This was the only response code received.’