Q.1

Facebook

I tried to access both HTTP and HTTPS versions of the Facebook website (www.facebook.com) and screenshots of the headers returned in the response of the curl request sent are given below.

The **HTTPS** response is given below.

```
Ehsuns-MBP:~ ehsunmohammedhanif$ curl -v https://www.facebook.com
*   Trying 2a03:2880:f11a:83:face:b00c:0:25de...
* TCP_NODELAY set
* Connected to www.facebook.com (2a03:2880:f11a:83:face:b00c:0:25de) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/cert.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-ECDSA-AES128-GCM-SHA256
* ALPN, server accepted to use h2
* Server certificate:
*  subject: C=US; ST=California; L=Menlo Park; O=Facebook, Inc.; CN=*.facebook.com
*  start date: Sep 22 00:00:00 2019 GMT
*  expire date: Dec 20 12:00:00 2019 GMT
*  subjectAltName: host "www.facebook.com" matched cert's "*.facebook.com"
*  issuer: C=US; O=DigiCert Inc; OU=www.digicert.com; CN=DigiCert SHA2 High Assurance Server CA
*  SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x7ff1ec005400)
> GET / HTTP/2
> Host: www.facebook.com
> User-Agent: curl/7.64.1
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS == 100)!
< HTTP/2 200
< set-cookie: fr=1FJa0vroMnWy4tFUv..BdzFZ2.FA.AAA.0.0.BdzFZ2.AWWw90tP; expires=Tue, 11-Feb-2020 19:16:05 GMT; Max-Age=7775999; path=/; domain=.facebook.com; secure; httponly
< set-cookie: sb=dlbMXaxu5eBPBv2cEBtv-06i; expires=Fri, 12-Nov-2021 19:16:06 GMT; Max-Age=63072000; path=/; domain=.facebook.com; secure; httponly
< cache-control: private, no-cache, no-store, must-revalidate
```

< pragma: no-cache
< strict-transport-security: max-age=15552000; preload
< vary: Accept-Encoding
< x-content-type-options: nosniff
< x-frame-options: DENY
< x-xss-protection: 0
< expires: Sat, 01 Jan 2000 00:00:00 GMT
< content-type: text/html; charset="utf-8"
< x-fb-debug:
Ik4B9aJ1XnzlDtT5UdOILdZwI0mrgAwd2Dx4Gt2fH5fdUBbW+54NkQ8VKLmzzKp19hy8yzyqoIorRhmVIa2b
9Q==
< date: Wed, 13 Nov 2019 19:16:06 GMT
< alt-svc: h3-23=":443"; ma=3600

The **HTTP** response is given below.

Ehsuns-MBP:~ ehsunmohammedhanif$ curl -v http://www.facebook.com
* Trying 2a03:2880:f11a:83:face:b00c:0:25de...
* TCP_NODELAY set
* Connected to www.facebook.com (2a03:2880:f11a:83:face:b00c:0:25de) port 80 (#0)
> GET / HTTP/1.1
> Host: www.facebook.com
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 302 Found
< Location: https://www.facebook.com/
< Content-Type: text/html; charset="utf-8"
< X-FB-Debug:
J9WAdg3W8vtDQxXkLBjwBbOzALpE/INjbP2vvntgB+csEhlinocW5ZMQdunTqt4PWaEmKtOjrl27EXvWkxal
kA==
< Date: Wed, 13 Nov 2019 19:27:35 GMT
< Alt-Svc: h3-23=":443"; ma=3600
< Connection: keep-alive
< Content-Length: 0
<
* Connection #0 to host www.facebook.com left intact
* Closing connection 0

The first reason why there is a difference between the HTTPS and HTTP header returned by Facebook is because in HTTPS the TLS is first set up and in HTTP there is no TLS. Facebook returns a HTTP/1.1 302 response when we try to access it with HTTP and this is because it is trying to redirect the website and that is what the HTTP 302 response code signifies. On the other hand, the headers in the HTTPS version contains a set-cookie header which includes a httponly flag which prevents client-side scripting. The HTTPS version also contains x-xss-protection header, the value of which set to 0 which means that XSS filtering is disabled for the website. Moreover, there is x-fb-debug header in both the HTTPS and HTTP version because this helps the website to debug the website when someone fills in a bug report. Another important header in the HTTPS response is the strict-transport-security header which enables the Facebook website to tell the browser that the website should only be accessed via HTTPS and the max-age flag here tells the browser to remember that the browser should remember for up to 15552000 seconds that the website should be accessible with HTTPS. Moreover, the server establishes the connection via port 80 for HTTP and via port 443 for HTTPS. One difference between the HTTPS version and the HTTP

version is the x-content-type-options header with the nosniff flag. This option helps the server to indicate that the MIME types shown in the content type header (text/html; charset="utf-8") are not to be changed and be followed. The content-length header doesn't exist in the

CS Birmingham

I tried to access both HTTP and HTTPS versions of the UoB Computer Science website (www.cs.bham.ac.uk) and screenshots of the headers returned in the response of the curl request sent are given below.

The **HTTPS** response is given below.

```
Ehsuns-MBP:~ ehsunmohammedhanif$ curl -v https://www.cs.bham.ac.uk
*   Trying 147.188.192.42...
* TCP_NODELAY set
* Connected to www.cs.bham.ac.uk (147.188.192.42) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/cert.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: C=GB; ST=West Midlands; L=BIRMINGHAM; O=University of Birmingham; OU=School of
Computer Science; CN=www-lb-2.cs.bham.ac.uk
*  start date: Feb  5 14:51:14 2018 GMT
*  expire date: Feb  5 15:01:00 2021 GMT
*  subjectAltName: host "www.cs.bham.ac.uk" matched cert's "www.cs.bham.ac.uk"
*  issuer: C=BM; O=QuoVadis Limited; CN=QuoVadis Global SSL ICA G3
*  SSL certificate verify ok.
> GET / HTTP/1.1
> Host: www.cs.bham.ac.uk
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Wed, 13 Nov 2019 19:37:48 GMT
< Server: Apache
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/html; charset=utf-8
```

The **HTTP** response is given below.

```
Ehsuns-MBP:~ ehsunmohammedhanif$ curl -v http://www.cs.bham.ac.uk
*   Trying 147.188.192.42...
* TCP_NODELAY set
* Connected to www.cs.bham.ac.uk (147.188.192.42) port 80 (#0)
> GET / HTTP/1.1
> Host: www.cs.bham.ac.uk
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Wed, 13 Nov 2019 19:38:55 GMT
< Server: Apache
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/html; charset=utf-8
```

The University of Birmingham Computer Science website returned almost the same headers for both the HTTPS and HTTP versions. The only difference here is that in the HTTPS response, the TLS is first set up. Moreover, for the HTTP version, the server establishes the connection via port 80 and via port 443 for the HTTPS version. Both the HTTPS response and the HTTP response contain a header called Transfer-Encoding which has a value called chunked which specifies that the data is sent in a succession of chunks. Here the server also tells what server technology it is using which is Apache.

Dr Ian Batten

The **HTTPS** version is given below.

```
Ehsuns-MBP:~ ehsunmohammedhanif$ curl -v https://www.batten.eu.org
*   Trying 2a04:92c7:e:5f4::b375...
* TCP_NODELAY set
* Connected to www.batten.eu.org (2a04:92c7:e:5f4::b375) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/cert.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-ECDSA-AES128-GCM-SHA256
* ALPN, server accepted to use h2
* Server certificate:
```

* subject: CN=*.batten.eu.org
* start date: Nov 4 14:21:48 2019 GMT
* expire date: Feb 2 14:21:48 2020 GMT
* subjectAltName: host "www.batten.eu.org" matched cert's "*.batten.eu.org"
* issuer: C=US; O=Let's Encrypt; CN=Let's Encrypt Authority X3
* SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x7f9b1f005400)
> GET / HTTP/2
> Host: www.batten.eu.org
> User-Agent: curl/7.64.1
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
< HTTP/2 200
< server: nginx
< date: Wed, 13 Nov 2019 20:37:12 GMT
< content-type: text/html
< content-length: 714
< last-modified: Mon, 03 Jun 2019 17:57:09 GMT
< etag: "5cf55f75-2ca"
< expires: Mon, 03 Jun 2019 18:27:09 GMT
< cache-control: no-cache
< content-security-policy: report-uri https://batten.report-uri.com/r/d/csp/enforce; upgrade-insecure-requests; default-src 'self'; plugin-types application/pdf; frame-ancestors 'self' ; img-src 'self' https:; style-src 'self' https://www.batten.eu.org; font-src 'self' data:
< strict-transport-security: max-age=31536000; includeSubDomains; preload
< expect-ct: enforce,max-age=31536000,report-uri=https://batten.report-uri.com/r/d/ct/enforce;
< x-xss-protection: 1; mode=block; report=https://batten.report-uri.com/r/d/xss/enforce
< x-frame-options: SAMEORIGIN
< x-content-type-options: nosniff
< referrer-policy: no-referrer-when-downgrade
< accept-ranges: bytes

The **HTTP** version is given below.

```
Ehsuns-MBP:~ ehsunmohammedhanif$ curl -v http://www.batten.eu.org
*   Trying 2a04:92c7:e:5f4::b375...
* TCP_NODELAY set
* Connected to www.batten.eu.org (2a04:92c7:e:5f4::b375) port 80 (#0)
> GET / HTTP/1.1
> Host: www.batten.eu.org
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Server: nginx
< Date: Wed, 13 Nov 2019 20:39:06 GMT
< Content-Type: text/html
< Content-Length: 162
< Connection: keep-alive
< Location: https://www.batten.eu.org/
< Expires: Wed, 13 Nov 2019 21:09:06 GMT
< Cache-Control: max-age=1800
< X-XSS-Protection: 1; mode=block; report=https://batten.report-uri.com/r/d/xss/enforce
< X-Frame-Options: SAMEORIGIN
< X-Content-Type-Options: nosniff
< Referrer-Policy: no-referrer-when-downgrade
```

Dr. Ian Batten's website has returned some very different headers for the HTTP as compared to the HTTPS version. The first thing is that when one tries to access it without HTTPS, it responds with 301 code meaning that it has been moved permanently to the HTTPS version and is trying to redirect the website to the HTTPS version. In the HTTPS response, the TLS is first set up. The HTTPS version uses HTTP/2 as compared to the HTTP/1.1 for the HTTP version. The HTTPS version specifies that the content-length header has a value of 714 bytes for the body as compared to the 162 bytes in the HTTP header. Moreover, the last-modified header is missing in the HTTP version. The HTTPS version has a header called etag which acts as an identifier for a resource with a specific version. The cache-control header is present in the HTTPS version which is also in the HTTP version. In the HTTPS version, the cache-control header with the no-cache flag helps to force the request to be submitted to the origin server by the cache. Moreover, the HTTP response contains the cache-control header with the max-age flag with the value 1800 that helps to keep a resource fresh for a maximum amount of 1800 seconds

the HTTPS version also contains the str-transport-security header to force the browser to accept only HTTPS requests. The HTTPS response contains a expect-ct header which helps to enforce policies for Certificate Transparency and considers the host as a known Expect-CT host for a maximum time of 31536000 seconds.

BBC News

The **HTTPS** version is given below.

```
Ehsuns-MBP:~ ehsunmohammedhanif$ curl -v https://www.bbc.co.uk
*   Trying 212.58.244.66...
* TCP_NODELAY set
* Connected to www.bbc.co.uk (212.58.244.66) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/cert.pem
  CApath: none
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-ECDSA-AES128-GCM-SHA256
* ALPN, server accepted to use h2
* Server certificate:
*  subject: C=GB; ST=London; L=London; O=British Broadcasting Corporation; CN=*.bbc.co.uk
*  start date: Jan 21 13:51:05 2019 GMT
*  expire date: Jan 22 13:51:05 2020 GMT
*  subjectAltName: host "www.bbc.co.uk" matched cert's "*.bbc.co.uk"
*  issuer: C=BE; O=GlobalSign nv-sa; CN=GlobalSign Organization Validation CA - SHA256 - G2
*  SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x7f950d008600)
> GET / HTTP/2
> Host: www.bbc.co.uk
> User-Agent: curl/7.64.1
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS == 200)!
< HTTP/2 200
< x-cache-action: HIT
< x-cache-hits: 712
< vary: Accept-Encoding, X-CDN, X-BBC-Edge-Scheme
< x-cache-age: 106
< cache-control: private, max-age=0, must-revalidate
< content-type: text/html; charset=utf-8
< date: Wed, 13 Nov 2019 20:40:30 GMT
< etag: W/"50178-PK/xYTqSzCai7WI5QhTqG54JWj0"
```

< set-cookie: ckns_mvt=c344b050-137d-4d57-887c-01265b0e8153; Domain=.bbc.co.uk; Path=/; Expires=Thu, 12 Nov 2020 20:38:44 GMT; HttpOnly; Secure
< x-frame-options: SAMEORIGIN
< content-length: 328056

The **HTTP** version is given below.

Ehsuns-MBP:~ ehsunmohammedhanif$ curl -v http://www.bbc.co.uk
*   Trying 212.58.244.66...
* TCP_NODELAY set
* Connected to www.bbc.co.uk (212.58.244.66) port 80 (#0)
> GET / HTTP/1.1
> Host: www.bbc.co.uk
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Server: nginx
< X-BBC-No-Scheme-Rewrite: 1
< X-Cache-Action: HIT
< X-Cache-Hits: 5864
< Vary: X-BBC-Edge-Scheme
< Cache-Control: public, max-age=3600
< X-Cache-Age: 2242
< Content-Type: text/html
< Date: Wed, 13 Nov 2019 20:41:18 GMT
< Location: https://www.bbc.co.uk/
< Content-Length: 162
< Connection: Keep-Alive

The BBC News website has some similarities and differences between the responses it gave for its HTTPS and the HTTP version. The HTTPS connection is established by the server through port 443 while the HTTP connection is established by the server through port 80. The HTTPS provides us information over HTTP/2 whereas the HTTP version provides information over HTTP/1.1. The HTTP version provides us with the server technology behind the server which is nginx unlike the HTTPS version. The HTTP version response gives a status code 301 which means it is trying to redirect the client to the HTTPS version. The BBC News website also contains an etag which means called etag which acts as an identifier for a resource with a specific version.

Conclusion

Here, we have compared headers for four different websites (Facebook, BBC News, CS UoB website and Dr. Ian Batten's website) but it was between the websites own HTTPS and HTTP versions. Let us zoom out to compare the responses between the different websites. Apart from the University of Birmingham Computer Science website, all of the websites gave different headers for their HTTP and the HTTPS version. Moreover, The BBC News website has some radically different headers like x-cache-hits and x-cache-action. The majority of the websites didn't provide much different headers.

Q.2

Dr Ian Batten - First Page

Ehsuns-MBP:~ ehsunmohammedhanif$ openssl s_client -connect batten.eu.org:443
GET /~igb/index.html HTTP/1.1
Host:www.batten.eu.org

This is how I constructed the request for fetching a page from Dr. Ian Batten's website which is secure. And this is why I had to use the openssl command instead of telnet. The full response that the server gave back to me is contained in the file Q.2 Batten 1. The response given to me by the server started with information about the HTTPS certificate. It contained headers similar to the ones we have seen in Question 1 after which it gave all the html content that made up the page. The lines shown in bold are the headers I wrote to get the index.html page from Dr. Ian Battens website.

Dr Ian Batten - Second Page

Ehsuns-MBP:~ ehsunmohammedhanif$ openssl s_client -connect batten.eu.org:443
GET /~igb/cs.html HTTP/1.1
Host: batten.eu.org

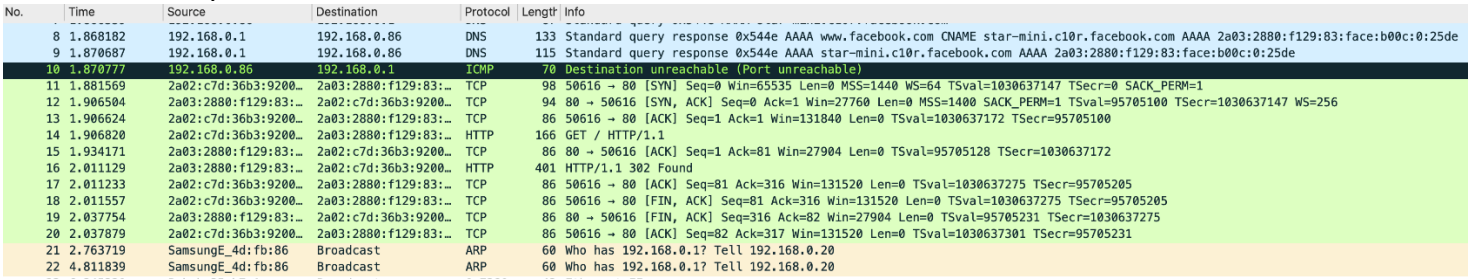
This is how I constructed the request for fetching a page from Dr. Ian Batten's website which is secure. And this is why I had to use the openssl command instead of telnet. The full response that the server gave back to me is contained in the file Q.2 Batten 2. The response given to me by the server started with information about the HTTPS certificate. It contained headers similar to the ones we have seen in Question 1 after which it gave all the html content that made up the page. The lines shown in bold are the headers I wrote to get the cs.html page from Dr. Ian Battens website.

Q.3

N.B. Please zoom in to view pictures clearly. I have also attached the pictures as a separate file on Canvas.

Facebook

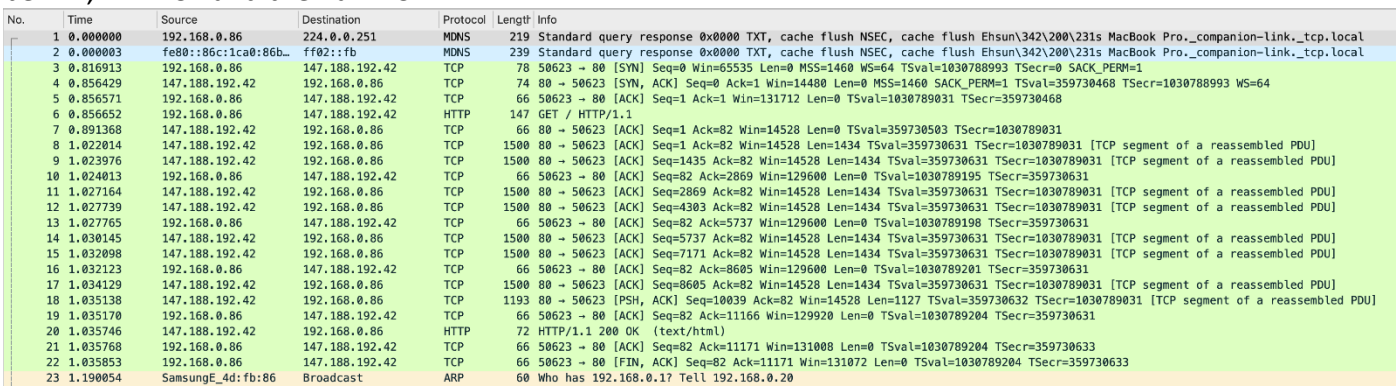
The Facebook TCP packets start with a SYN, SYN ACK and then an ACK which is normal. Then there are other packets after which the connection finishes with FIN ACK first sent by the client to the server and then one sent by the server to the client after which the client sends an ACK.



No.	Time	Source	Destination	Protocol	Length	Info
8	1.868182	192.168.0.1	192.168.0.86	DNS	133	Standard query response 0x544e AAAA www.facebook.com CNAME star-mini.c10r.facebook.com AAAA 2a03:2880:f129:83:face:b00c:0:25de
9	1.870687	192.168.0.1	192.168.0.86	DNS	115	Standard query response 0x544e AAAA star-mini.c10r.facebook.com AAAA 2a03:2880:f129:83:face:b00c:0:25de
10	1.870777	192.168.0.86	192.168.0.1	ICMP	70	Destination unreachable (Port unreachable)
11	1.881569	2a02:c7d:36b3:9200…	2a03:2880:f129:83:…	TCP	98	50616 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=64 TSval=1030637147 TSecr=0 SACK_PERM=1
12	1.906504	2a03:2880:f129:83:…	2a02:c7d:36b3:9200…	TCP	94	80 → 50616 [SYN, ACK] Seq=0 Ack=1 Win=27760 Len=0 MSS=1400 SACK_PERM=1 TSval=95705100 TSecr=1030637147 WS=256
13	1.906624	2a02:c7d:36b3:9200…	2a03:2880:f129:83:…	TCP	86	50616 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0 TSval=1030637172 TSecr=95705100
14	1.906820	2a02:c7d:36b3:9200…	2a03:2880:f129:83:…	HTTP	166	GET / HTTP/1.1
15	1.934171	2a03:2880:f129:83:…	2a02:c7d:36b3:9200…	TCP	86	80 → 50616 [ACK] Seq=1 Ack=81 Win=27904 Len=0 TSval=95705128 TSecr=1030637172
16	2.011129	2a03:2880:f129:83:…	2a02:c7d:36b3:9200…	HTTP	401	HTTP/1.1 302 Found
17	2.011233	2a02:c7d:36b3:9200…	2a03:2880:f129:83:…	TCP	86	50616 → 80 [ACK] Seq=81 Ack=316 Win=131520 Len=0 TSval=1030637275 TSecr=95705205
18	2.011557	2a02:c7d:36b3:9200…	2a03:2880:f129:83:…	TCP	86	50616 → 80 [FIN, ACK] Seq=81 Ack=316 Win=131520 Len=0 TSval=1030637275 TSecr=95705205
19	2.037754	2a03:2880:f129:83:…	2a02:c7d:36b3:9200…	TCP	86	80 → 50616 [FIN, ACK] Seq=316 Ack=82 Win=27904 Len=0 TSval=95705231 TSecr=1030637275
20	2.037879	2a02:c7d:36b3:9200…	2a03:2880:f129:83:…	TCP	86	50616 → 80 [ACK] Seq=82 Ack=317 Win=131520 Len=0 TSval=1030637301 TSecr=95705231
21	2.763719	SamsungE_4d:fb:86	Broadcast	ARP	60	Who has 192.168.0.1? Tell 192.168.0.20
22	4.811839	SamsungE_4d:fb:86	Broadcast	ARP	60	Who has 192.168.0.1? Tell 192.168.0.20

University of Birmingham Computer Science

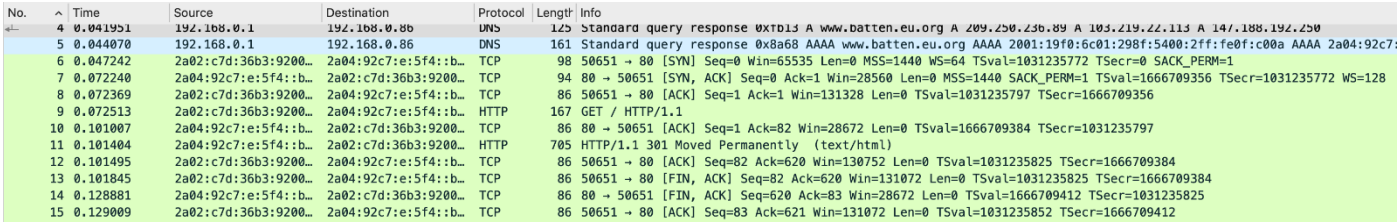
The UoB CS website TCP packets start with a SYN, SYN ACK and then an ACK which is normal. Then there some other packets after which the connection finishes with FIN ACK. The only things unusual here is that there is only one final FIN ACK which is sent by the client to the server because the usual sequence should be FIN, FIN ACK and then an ACK.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.86	224.0.0.251	MDNS	219	Standard query response 0x0000 TXT, cache flush NSEC, cache flush Ehsun\342\200\231s MacBook Pro._companion-link._tcp.local
2	0.000003	fe80::86c:1ca0:86b…	ff02::fb	MDNS	239	Standard query response 0x0000 TXT, cache flush NSEC, cache flush Ehsun\342\200\231s MacBook Pro._companion-link._tcp.local
3	0.816913	192.168.0.86	147.188.192.42	TCP	78	50623 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1030788993 TSecr=0 SACK_PERM=1
4	0.856429	147.188.192.42	192.168.0.86	TCP	74	80 → 50623 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=359730468 TSecr=1030788993 WS=64
5	0.856571	192.168.0.86	147.188.192.42	TCP	66	50623 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=1030789031 TSecr=359730468
6	0.856652	192.168.0.86	147.188.192.42	HTTP	147	GET / HTTP/1.1
7	0.891368	147.188.192.42	192.168.0.86	TCP	66	80 → 50623 [ACK] Seq=1 Ack=82 Win=14528 Len=0 TSval=359730503 TSecr=1030789031
8	1.022014	147.188.192.42	192.168.0.86	TCP	1500	80 → 50623 [ACK] Seq=1 Ack=82 Win=14528 Len=1434 TSval=359730631 TSecr=1030789031 [TCP segment of a reassembled PDU]
9	1.023976	147.188.192.42	192.168.0.86	TCP	1500	80 → 50623 [ACK] Seq=1435 Ack=82 Win=14528 Len=1434 TSval=359730631 TSecr=1030789031 [TCP segment of a reassembled PDU]
10	1.024013	192.168.0.86	147.188.192.42	TCP	66	50623 → 80 [ACK] Seq=82 Ack=2869 Win=129600 Len=0 TSval=1030789195 TSecr=359730631
11	1.027164	147.188.192.42	192.168.0.86	TCP	1500	80 → 50623 [ACK] Seq=2869 Ack=82 Win=14528 Len=1434 TSval=359730631 TSecr=1030789031 [TCP segment of a reassembled PDU]
12	1.027739	147.188.192.42	192.168.0.86	TCP	1500	80 → 50623 [ACK] Seq=4303 Ack=82 Win=14528 Len=1434 TSval=359730631 TSecr=1030789031 [TCP segment of a reassembled PDU]
13	1.027765	192.168.0.86	147.188.192.42	TCP	66	50623 → 80 [ACK] Seq=82 Ack=5737 Win=129600 Len=0 TSval=1030789198 TSecr=359730631
14	1.030145	147.188.192.42	192.168.0.86	TCP	1500	80 → 50623 [ACK] Seq=5737 Ack=82 Win=14528 Len=1434 TSval=359730631 TSecr=1030789031 [TCP segment of a reassembled PDU]
15	1.032098	147.188.192.42	192.168.0.86	TCP	1500	80 → 50623 [ACK] Seq=7171 Ack=82 Win=14528 Len=1434 TSval=359730631 TSecr=1030789031 [TCP segment of a reassembled PDU]
16	1.032123	192.168.0.86	147.188.192.42	TCP	66	50623 → 80 [ACK] Seq=82 Ack=8605 Win=129600 Len=0 TSval=1030789201 TSecr=359730631
17	1.034129	147.188.192.42	192.168.0.86	TCP	1500	80 → 50623 [ACK] Seq=8605 Ack=82 Win=14528 Len=1434 TSval=359730631 TSecr=1030789031 [TCP segment of a reassembled PDU]
18	1.035138	147.188.192.42	192.168.0.86	TCP	1193	80 → 50623 [PSH, ACK] Seq=10039 Ack=82 Win=14528 Len=1127 TSval=359730632 TSecr=1030789031 [TCP segment of a reassembled PDU]
19	1.035170	192.168.0.86	147.188.192.42	TCP	66	50623 → 80 [ACK] Seq=82 Ack=11166 Win=129920 Len=0 TSval=1030789204 TSecr=359730631
20	1.035746	147.188.192.42	192.168.0.86	HTTP	72	HTTP/1.1 200 OK (text/html)
21	1.035768	192.168.0.86	147.188.192.42	TCP	66	50623 → 80 [ACK] Seq=82 Ack=11171 Win=131008 Len=0 TSval=1030789204 TSecr=359730633
22	1.035853	192.168.0.86	147.188.192.42	TCP	66	50623 → 80 [FIN, ACK] Seq=82 Ack=11171 Win=131072 Len=0 TSval=1030789204 TSecr=359730633
23	1.190054	SamsungE_4d:fb:86	Broadcast	ARP	60	Who has 192.168.0.1? Tell 192.168.0.20

Dr. Ian Batten

The website of Dr. Ian Batten started with a SYN, SYN ACK and then an ACK packet which is normal. Then there are some other packets after which the connection finishes with FIN ACK. The only unusual thing here is that there are two FIN ACK packets and then an ACK packet in the end. This is unlike Facebook and UoB CS website.



No.	Time	Source	Destination	Protocol	Length	Info
4	0.041951	192.168.0.1	192.168.0.86	DNS	125	Standard query response 0xfb13 A www.batten.eu.org A 209.250.236.89 A 103.219.22.113 A 147.188.192.250
5	0.044070	192.168.0.1	192.168.0.86	DNS	161	Standard query response 0x8a68 AAAA www.batten.eu.org AAAA 2001:19f0:6c01:298f:5400:2ff:fe0f:c00a AAAA 2a04:92c7:
6	0.047242	2a02:c7d:36b3:9200…	2a04:92c7:e:5f4::b…	TCP	98	50651 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=64 TSval=1031235772 TSecr=0 SACK_PERM=1
7	0.072240	2a04:92c7:e:5f4::b…	2a02:c7d:36b3:9200…	TCP	94	80 → 50651 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MSS=1440 SACK_PERM=1 TSval=1666709356 TSecr=1031235772 WS=128
8	0.072369	2a02:c7d:36b3:9200…	2a04:92c7:e:5f4::b…	TCP	86	50651 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=1031235797 TSecr=1666709356
9	0.072513	2a02:c7d:36b3:9200…	2a04:92c7:e:5f4::b…	HTTP	167	GET / HTTP/1.1
10	0.101007	2a04:92c7:e:5f4::b…	2a02:c7d:36b3:9200…	TCP	86	80 → 50651 [ACK] Seq=1 Ack=82 Win=28672 Len=0 TSval=1666709384 TSecr=1031235797
11	0.101404	2a04:92c7:e:5f4::b…	2a02:c7d:36b3:9200…	HTTP	705	HTTP/1.1 301 Moved Permanently (text/html)
12	0.101495	2a02:c7d:36b3:9200…	2a04:92c7:e:5f4::b…	TCP	86	50651 → 80 [ACK] Seq=82 Ack=620 Win=130752 Len=0 TSval=1031235825 TSecr=1666709384
13	0.101845	2a02:c7d:36b3:9200…	2a04:92c7:e:5f4::b…	TCP	86	50651 → 80 [FIN, ACK] Seq=82 Ack=620 Win=131072 Len=0 TSval=1031235825 TSecr=1666709384
14	0.128881	2a04:92c7:e:5f4::b…	2a02:c7d:36b3:9200…	TCP	86	80 → 50651 [FIN, ACK] Seq=620 Ack=83 Win=28672 Len=0 TSval=1666709412 TSecr=1031235825
15	0.129009	2a02:c7d:36b3:9200…	2a04:92c7:e:5f4::b…	TCP	86	50651 → 80 [ACK] Seq=83 Ack=621 Win=131072 Len=0 TSval=1031235852 TSecr=1666709412

BBC News

The BBC News website also starts with a standard SYN, SYN ACK, and then an ACK packet sequence. The ending is also unusual similar to Dr. Ian's website because this website also ends with two FIN ACK packets

after which there is one final ACK.

No.	Time	Source	Destination	Protocol	Length	Info
5	0.035097	192.168.0.1	192.168.0.86	DNS	73	Standard query response 0x75f8 AAAA www.bbc.co.uk
6	0.059205	192.168.0.1	192.168.0.86	DNS	136	Standard query response 0x75f8 AAAA www.bbc.net.uk SOA 212.58.230.200
7	0.111145	192.168.0.86	212.58.244.26	TCP	78	50657 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=1031411279 TSecr=0 SACK_PERM=1
8	0.136211	212.58.244.26	192.168.0.86	TCP	74	80 → 50657 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=1446 SACK_PERM=1 TSval=25444182 TSecr=1031411279 WS=128
9	0.136321	192.168.0.86	212.58.244.26	TCP	66	50657 → 80 [ACK] Seq=1 Ack=1 Win=131904 Len=0 TSval=1031411304 TSecr=25444182
10	0.163464	212.58.244.26	192.168.0.86	TCP	66	[TCP Window Update] 80 → 50657 [ACK] Seq=1 Ack=1 Win=28928 Len=0 TSval=25444182 TSecr=1031411304
11	0.163526	192.168.0.86	212.58.244.26	HTTP	143	GET / HTTP/1.1
12	0.188139	212.58.244.26	192.168.0.86	TCP	72	80 → 50657 [ACK] Seq=1 Ack=78 Win=29056 Len=0 TSval=25444209 TSecr=1031411331
13	0.189175	212.58.244.26	192.168.0.86	HTTP	566	HTTP/1.1 301 Moved Permanently (text/html)
14	0.189269	192.168.0.86	212.58.244.26	TCP	66	50657 → 80 [ACK] Seq=78 Ack=501 Win=131392 Len=0 TSval=1031411356 TSecr=25444209
15	0.189648	192.168.0.86	212.58.244.26	TCP	66	50657 → 80 [FIN, ACK] Seq=78 Ack=501 Win=131392 Len=0 TSval=1031411356 TSecr=25444209
16	0.214953	212.58.244.26	192.168.0.86	TCP	66	80 → 50657 [FIN, ACK] Seq=501 Ack=79 Win=29056 Len=0 TSval=25444236 TSecr=1031411356
17	0.215070	192.168.0.86	212.58.244.26	TCP	66	50657 → 80 [ACK] Seq=79 Ack=502 Win=131392 Len=0 TSval=1031411380 TSecr=25444236

Conclusion

One potential problem can be that of a DoS attack on the websites because there are almost ten packets used in every connection that we make. So, if done at a very large scale, it can cause the website to crash due to the high volume of requests.