

Networking Lab 8 HTTP

A simple web page

1. Open your web browser and Wireshark
2. Start a capture in Wireshark
3. Browse to this site: <http://www.ismycomputeron.com/> (the site was chosen for its simplicity.)
4. Stop the capture
5. In the Wireshark display filter, enter `dns`. You should see your host's request for the IP address of www.ismycomputeron.com, and the response. If there are many DNS packets, you can use a more specific filter, `dns contains "ismycomputeron"`. Often there are requests for both A (IPv4) and AAAA (IPv6) addresses. Write down the IPv4 address that is in the response to your DNS query.
6. Clear the Wireshark display filter and look for the web traffic for [ismycomputeron.com](http://www.ismycomputeron.com). If it's hard to find in the other traffic, enter a display filter of `ip.addr==IP address from step 5`.
7. Right-click on the GET request, and select Follow TCP Stream. You should be able to identify the parts of the stream:
 - a. Three way handshake
 - b. GET request
 - c. Server response
 - d. Close connection
8. Examine the HTTP headers in the GET request.

Hand in: What browser and operating system were you using when you browsed to the page (look at the User Agent)?

The Host header should match the site you browsed to. Servers that have multiple sites on the same IP address use this to tell which site you are requesting.
9. Examine the headers in the response. What web server and operating system is the server probably using? Note: headers can be forged.

The Referrer field

1. Do a Google search for `is my computer on fire`.
2. Start a packet capture.
3. On the Google search page, click on the link for ismycomputeronfire.com
4. Stop the capture and locate the TCP stream for the page.
5. Examine the GET request.

Hand in: Is there a Referrer field present? What is its value?

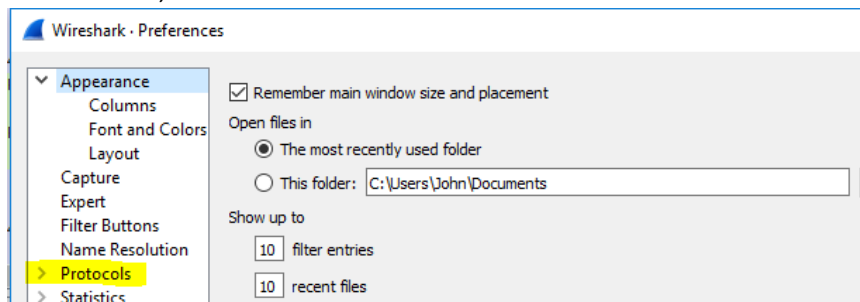
Examining TLS/SSL traffic from your browser

It is often helpful in troubleshooting to look at the network traffic from your browser. However, over half of all web traffic is now encrypted so we need to decrypt it. The Chrome and Firefox browsers can store the encryption keys (pre-master secrets) that the browsers and web servers generate on the fly while creating encrypted connections (HTTPS). Wireshark can use these keys to decrypt the network traffic. It is important to note that we are decrypting traffic from **our own** browser. We cannot decrypt the traffic to someone else's browser unless they give us the keys. We will cover Man In The Middle (MITM) attacks against encryption later in the Cryptography section of our course.

The web page, <https://jimshaver.net/2015/02/11/decrypting-tls-browser-traffic-with-wireshark-the-easy-way/>, shows how to configure Chrome or Firefox to log their session keys by setting the environment variable SSLKEYLOGFILE to the path to a text file. It then shows how to edit Wireshark preferences for SSL so that Wireshark uses the key file. The instructions in the article are detailed and complete with screenshots, so they are not reproduced here.

For practice, we will look at an HTTP connection in Wireshark. This is a connection to the BRCC web site that was extracted from a larger capture, and then saved with File - Export Specified Packets, with the Displayed radio button selected instead of Captured. The session keys were saved using File - Export SSL Session Keys.

- 1) Open the file brcc.edu.pcapng in Wireshark
- 2) In Wireshark, select Edit - Preferences - Protocols

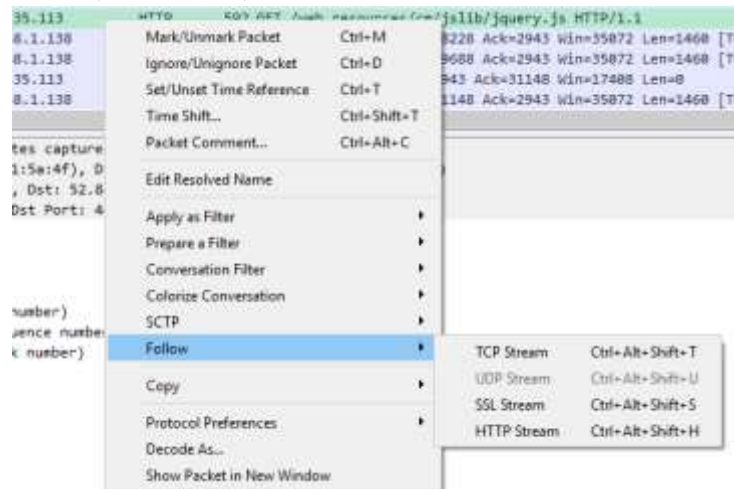


Then scroll down to SSL. Use the Browse button to select the path to the location where you stored brcc.edu.keys.

You should find that Wireshark displays a combination of raw TLS data and decrypted HTTP data.

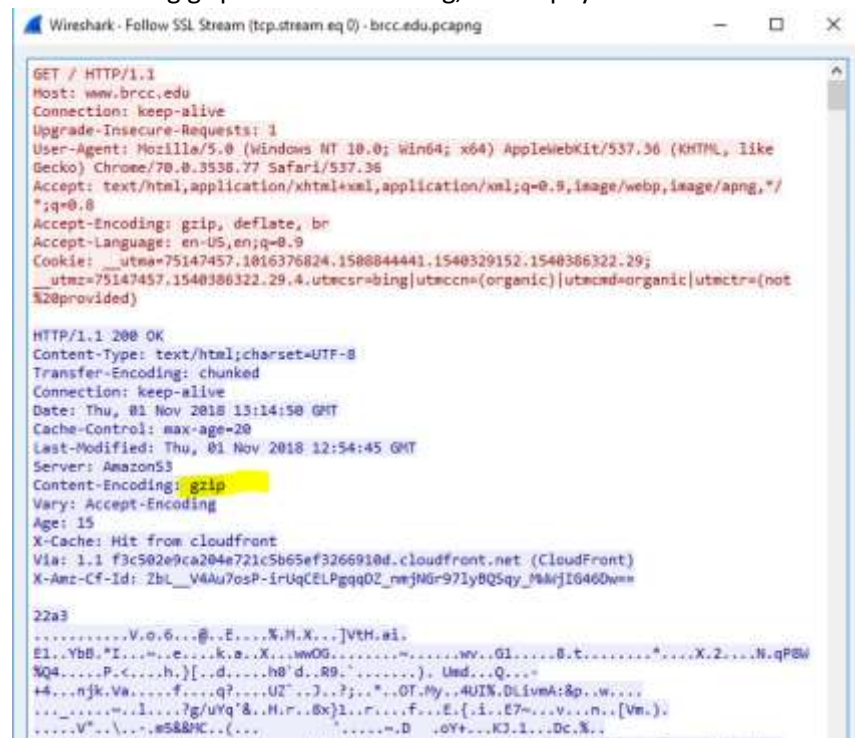
No.	Time	Source	Destination	Protocol	Length	Info
34	0.407988	52.84.35.113	192.168.1.138	TCP	1514	443 → 1484 [ACK] Seq=21725 Ack=1812 Win=32768 Len=1468 [TCP segment of a reassembled PDU]
35	0.409174	192.168.1.138	52.84.35.113	TCP	54	1484 → 443 [ACK] Seq=1812 Ack=20185 Win=17488 Len=0
36	0.438575	52.84.35.113	192.168.1.138	TLSv1.2	818	[SSL segment of a reassembled PDU], Application Data
37	0.412652	192.168.1.138	52.84.35.113	HTTP	647	GET /web_resources/themes/www-brcc-percussion-com/www-brcc-percussion-com.css HTTP/1.1
38	0.511859	52.84.35.113	192.168.1.138	TCP	1514	443 → 1484 [ACK] Seq=23540 Ack=2405 Win=34848 Len=1468 [TCP segment of a reassembled PDU]
39	0.516824	52.84.35.113	192.168.1.138	TCP	1514	443 → 1484 [ACK] Seq=25489 Ack=2405 Win=34848 Len=1468 [TCP segment of a reassembled PDU]
40	0.517856	192.168.1.138	52.84.35.113	TCP	54	1484 → 443 [ACK] Seq=2405 Ack=26869 Win=17488 Len=0
41	0.520639	52.84.35.113	192.168.1.138	TLSv1.2	1379	[SSL segment of a reassembled PDU]
42	0.521994	52.84.35.113	192.168.1.138	HTTP	88	HTTP/1.1 200 OK (text/css)
43	0.521609	192.168.1.138	52.84.35.113	TCP	54	1484 → 443 [ACK] Seq=2405 Ack=28228 Win=18128 Len=0
44	0.538709	192.168.1.138	52.84.35.113	HTTP	502	GET /web_resources/cm/jquery/jquery.js HTTP/1.1
45	0.665814	52.84.35.113	192.168.1.138	TCP	1514	443 → 1484 [ACK] Seq=28228 Ack=2945 Win=33872 Len=1468 [TCP segment of a reassembled PDU]

If you right-click on one of the green HTTP packets you will see that Follow now has three options: TCP Stream, SSL Stream, and HTTP Stream.

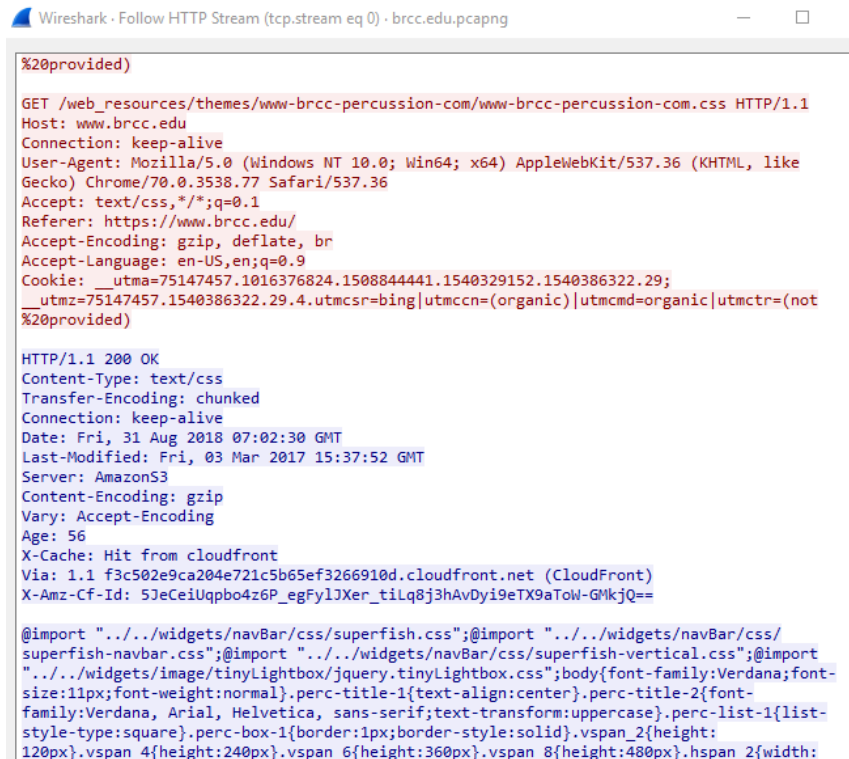


Follow > TCP Stream just gives us the raw encrypted data, so no help there.

Follow > SSL Stream is much better and gives us the decrypted data. Note that in this capture, the web server is using gzip Content-Encoding, so the payload is unreadable. It is compressed with gzip.



Follow > HTTP Stream is also decrypted but goes a step further by expanding the compressed data as well. It changed the order of some parts of the client/server conversation, however.



```
%20provided)

GET /web_resources/themes/www-brcc-percussion-com/www-brcc-percussion-com.css HTTP/1.1
Host: www.brcc.edu
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/70.0.3538.77 Safari/537.36
Accept: text/css,*/*;q=0.1
Referer: https://www.brcc.edu/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: __utma=75147457.1016376824.1508844441.1540329152.1540386322.29;
__utmz=75147457.1540386322.29.4.utmcsr=bing|utmccn=(organic)|utmcmd=organic|utmctr=(not
%20provided)

HTTP/1.1 200 OK
Content-Type: text/css
Transfer-Encoding: chunked
Connection: keep-alive
Date: Fri, 31 Aug 2018 07:02:30 GMT
Last-Modified: Fri, 03 Mar 2017 15:37:52 GMT
Server: AmazonS3
Content-Encoding: gzip
Vary: Accept-Encoding
Age: 56
X-Cache: Hit from cloudfront
Via: 1.1 f3c502e9ca204e721c5b65ef3266910d.cloudfront.net (CloudFront)
X-Amz-Cf-Id: 5JeCeIUqbo4z6P_egFylJXer_tilQ8j3hAvDyi9eTX9aToW-GMkjQ==

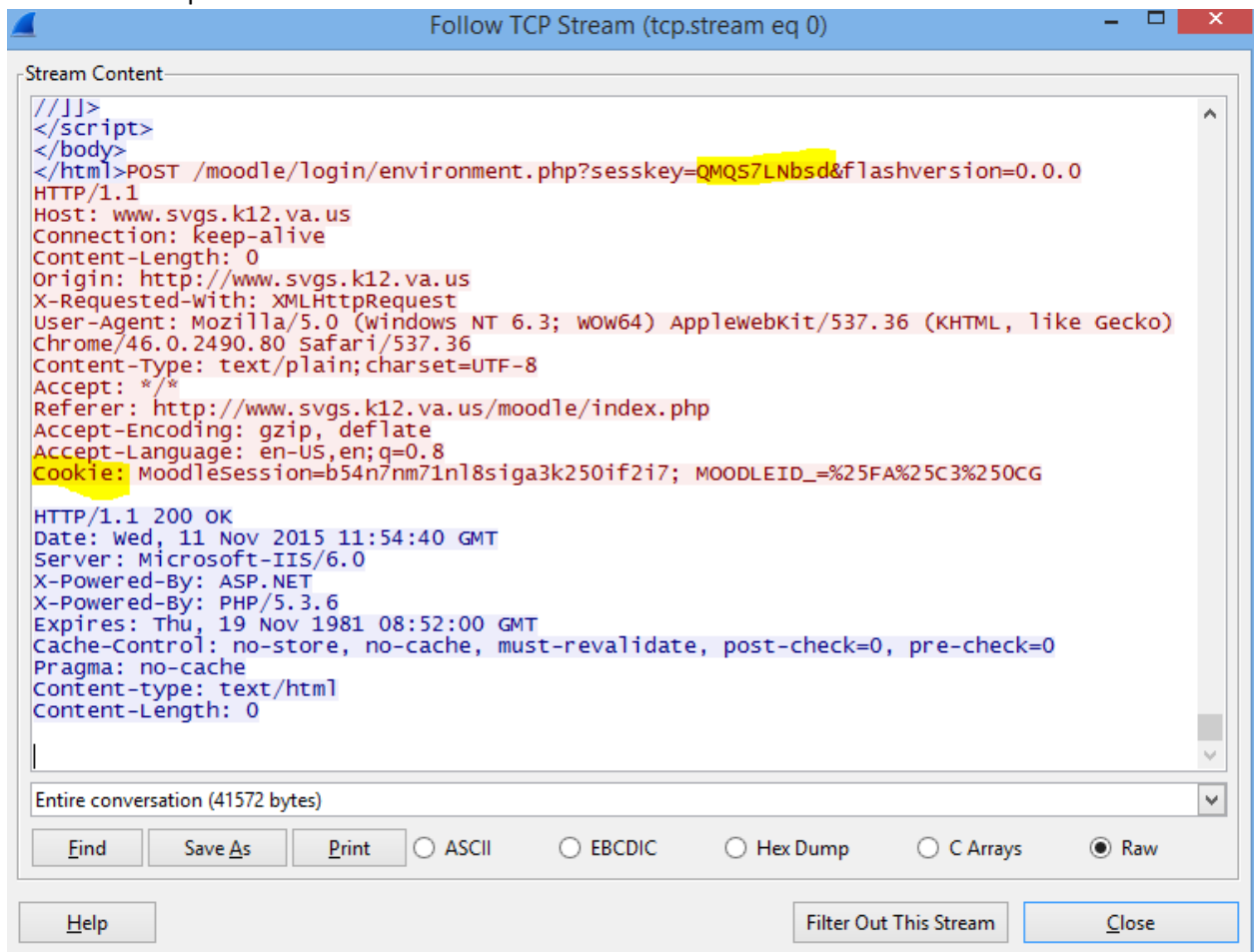
@import ".../widgets/navBar/css/superfish.css";@import ".../widgets/navBar/css/
superfish-navbar.css";@import ".../widgets/navBar/css/superfish-vertical.css";@import
".../widgets/image/tinyLightbox/jquery.tinyLightbox.css";body{font-family:Verdana;font-
size:11px;font-weight:normal}.perc-title-1{text-align:center}.perc-title-2{font-
family:Verdana, Arial, Helvetica, sans-serif;text-transform:uppercase}.perc-list-1{list-
style-type:square}.perc-box-1{border:1px;border-style:solid}.vspan_2{height:
120px}.vspan_4{height:240px}.vspan_6{height:360px}.vspan_8{height:480px}.hspan_2{width:
```

Hand in: Try decrypting an HTTPS conversation on your own. Remember that you have to have the environment key set to save your keys (<https://jimshaver.net/2015/02/11/decrypting-tls-browser-traffic-with-wireshark-the-easy-way/>) and you have to set your Wireshark preferences for SSL to use your key. Don't forget, and leave your Wireshark pointing at the keys for brcc.edu.pcapng! Hand in a screenshot of your Follow HTTP Stream.

A more complicated site

1. Open a browser and go to the school LMS (Canvas, Blackboard, etc.) site, but don't log in.
2. Start a packet capture
3. Log in and then stop the capture when the page is complete.
4. Find the decrypted traffic and select Follow HTTP Stream.

5. This is an example for Moodle. Note that it has session data in both cookies and POST data.



6. Hand in. Can you find your session cookies for a site that you log in to? If so, submit a screenshot. When you are done, log out of the site so that the cookies are no longer valid.

Recap of items to Hand In

From Simple Web Page, step 8:

What browser and operating system were you using when you browsed to the page (look at the User Agent)?

From The Referer Field, step 5:

Is there a Referer field present? What is its value?

From the end of Examining TLS/SSL traffic from your browser:

Try decrypting an HTTPS conversation on your own. Hand in a screenshot of your Follow HTTP Stream.

From the end of A more complicated site:

Can you find your session cookies for a site that you log in to? If so, submit a screenshot. When you are done, log out of the site so that the cookies are no longer valid.