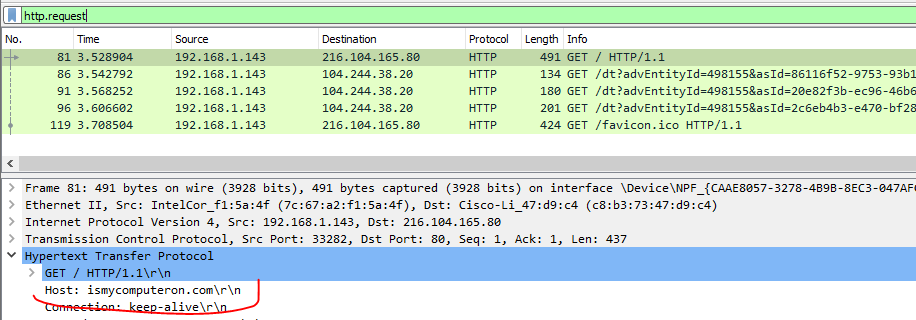
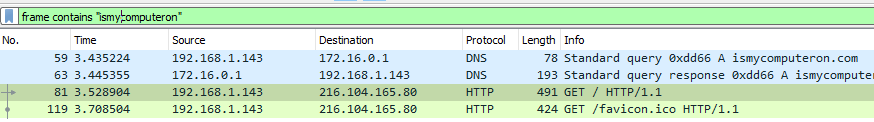
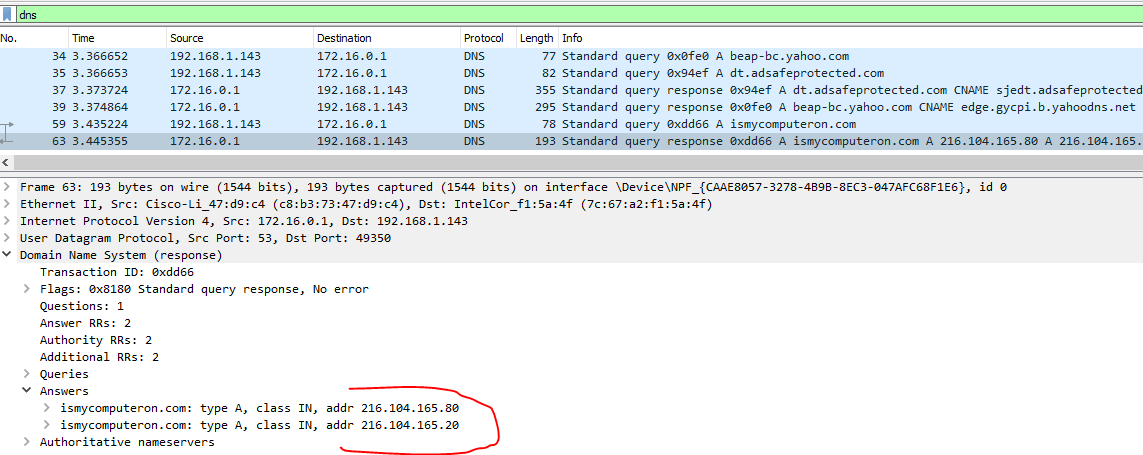
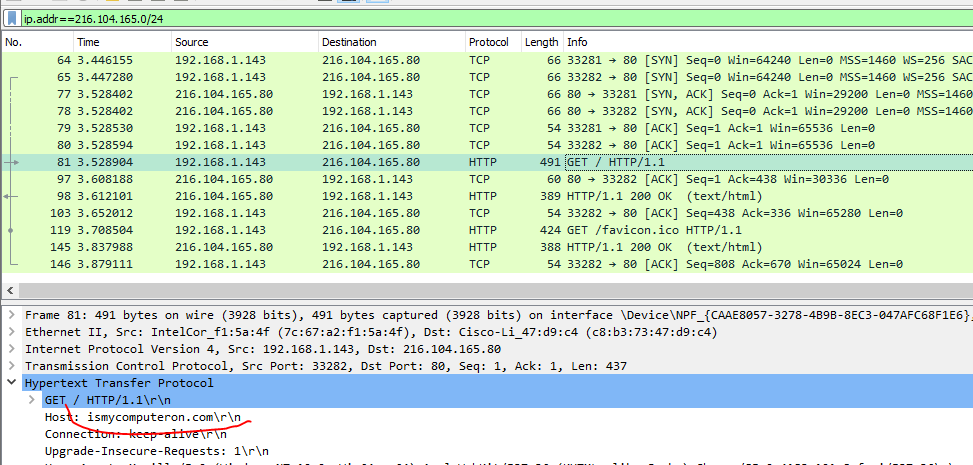
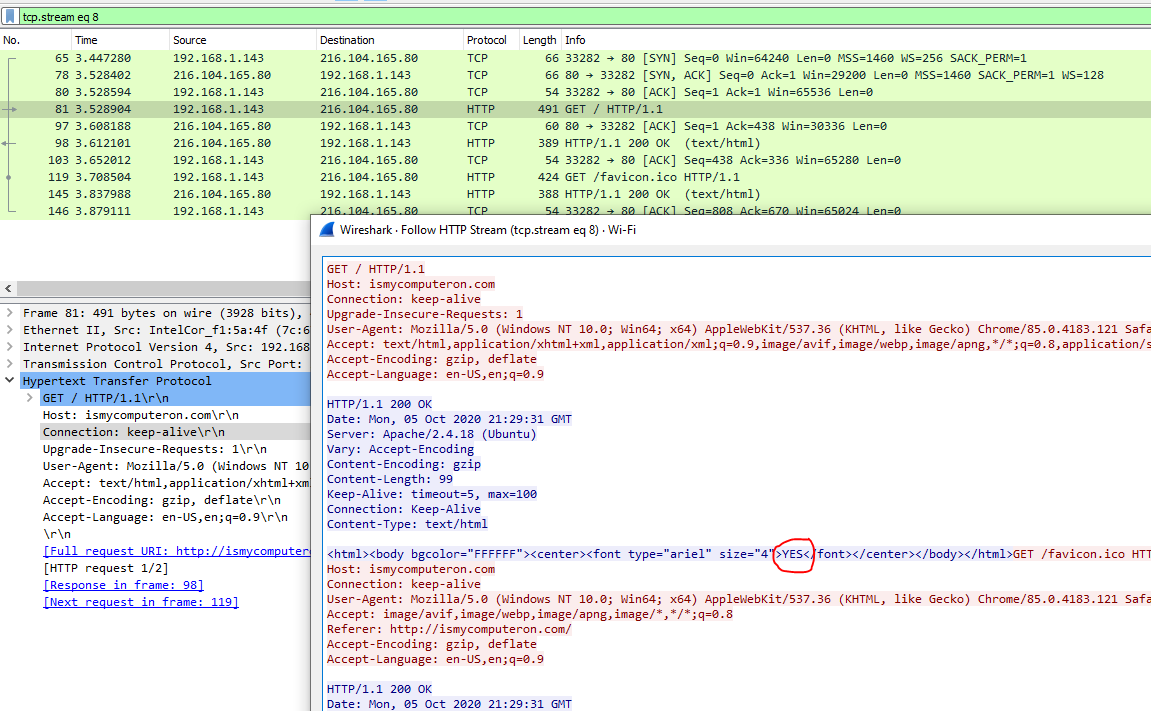
# Networking Lab 8 HTTP

## A simple web page

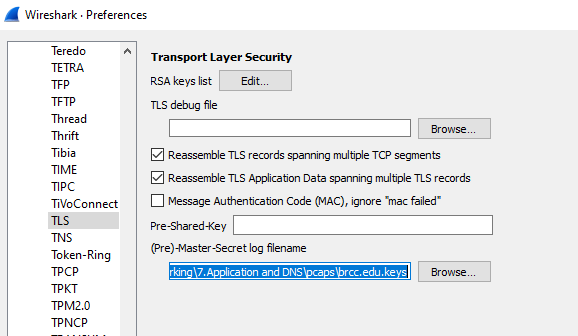
1. Open your web browser and Wireshark
2. Start a capture in Wireshark
3. Browse to this site: <http://www.ismycomputeron.com/> (the site was chosen for its simplicity.)
4. Stop the capture
5. Find the HTTP traffic to the ismycomputeron website in Wireshark. If there is not much traffic in your browser, you may find it by putting http in the display filter window.  
     
     
   If that doesn’t work, try a display filter that will examine every frame to see if it contains the string ismycomputeron.  
   frame contains “ismycomputeron”  
     
     
   Or, you can look for the DNS response to the request for ismycomputeron, find the address of the server, and then enter the IP address into a display filter.  
     
   In this case, we got two answers. You can put them in one at a time to see which one the browser used, or you can put them both in at the same time by using a subnet mask.  
   ip.addr==216.104.165.0/24  
     
   In this case the browser opened two connections (source ports 33281 and 33282), but it is not hard to find the correct connection.  
     
   You may want to try all of the methods above for practice. One of the biggest skills in using Wireshark is being able to filter out the packets you want (needle) from a huge amount of network traffic (haystack).
6. Right-click on the GET request and select Follow HTTP Stream. If the web server is compressing the data it sends (usually with gzip), Follow HTTP Stream will uncompress it. Follow TCP Stream will not.   
     
   You should be able to identify the parts of the stream:
   1. Three-way handshake
   2. GET request
   3. Server response
   4. Close connection
7. Examine the HTTP headers in the GET request.   
   Hand in: Look at the User-Agent header. Does this match with the browser and operating system were you using when you browsed to the page?  
   (If you are confused about why your browser is claiming to have multiple identities in its User-Agent header, read about the curious and funny history of the User-Agent here  
    <https://techcommunity.microsoft.com/t5/discussions/funny-story-of-browser-wars-from-the-beginning-of-the-time-user/m-p/1079832> )  
   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.
8. Examine the headers in the response. What web server and operating system is the server probably using? Note: headers can be forged.

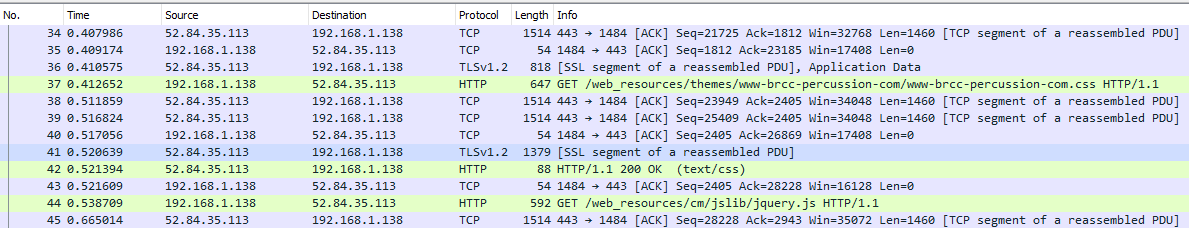
## Examining TLS/SSL traffic from your browser

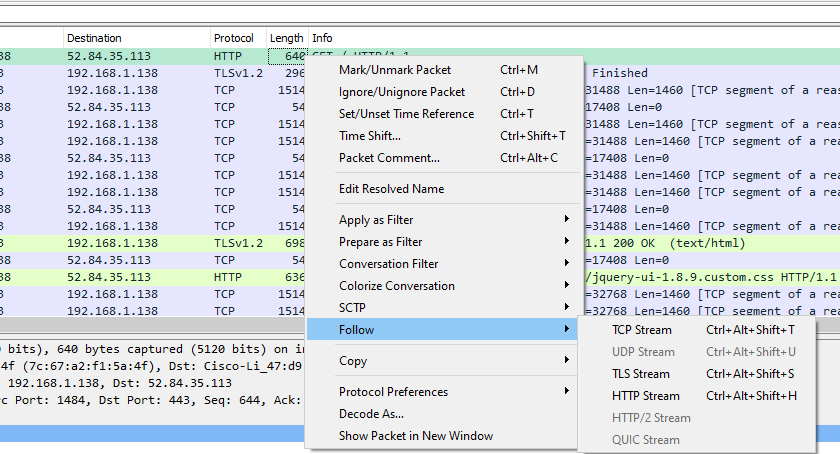
It is often helpful in troubleshooting to look at the network traffic from your browser. However, over half of all web traffic is now encrypted so we need to decrypt it. The Chrome and Firefox browsers can store the encryption keys (pre-master secrets) that the browsers and web servers generate on the 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://support.f5.com/csp/article/K50557518> or <https://knowledgebase.paloaltonetworks.com/KCSArticleDetail?id=kA14u000000HB8gCAG&lang=en_US%E2%80%A9> or <https://www.youtube.com/watch?v=CMbehohHj7c> , 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 TLS so that Wireshark uses the key file. Note: Recent versions of Wireshark have changed the name of the protocol setting from SSL to TLS; use TLS instead of SSL as the article tells you. The instructions in the article are detailed and complete with screenshots, so they are not reproduced here.

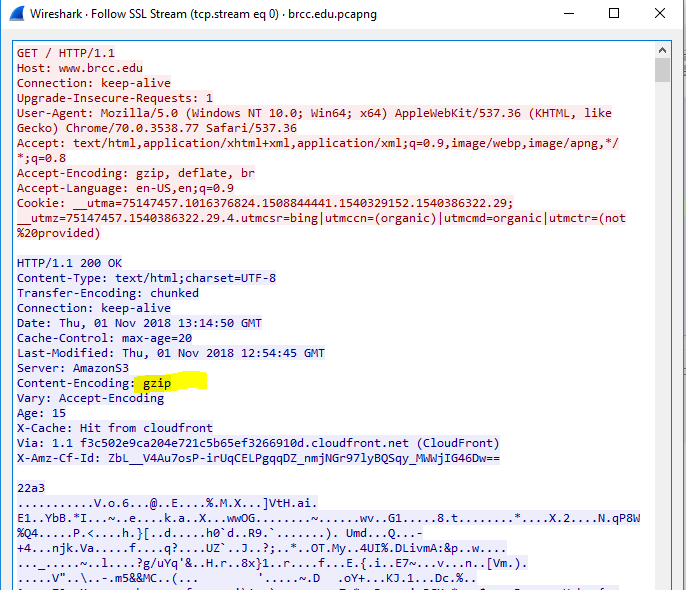
For practice, we will look at an HTTP connection in Wireshark. This is a connection to the BRCC web site that was extracted from a larger capture, and then saved with File - Export Specified Packets, with the Displayed radio button selected instead of Captured. The session keys were saved using File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

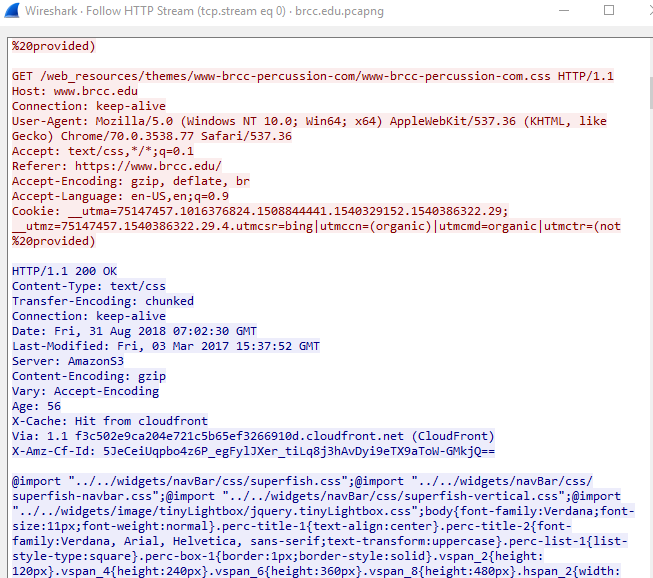
1. Open the file brcc.edu.pcapng in Wireshark
2. In Wireshark, select Edit - Preferences – Protocols, and scroll down to TLS  
     
   Then scroll down to TLS. Use the Browse button to select the path to the location where you stored brcc.edu.keys.

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


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


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

Follow > TLS 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.  


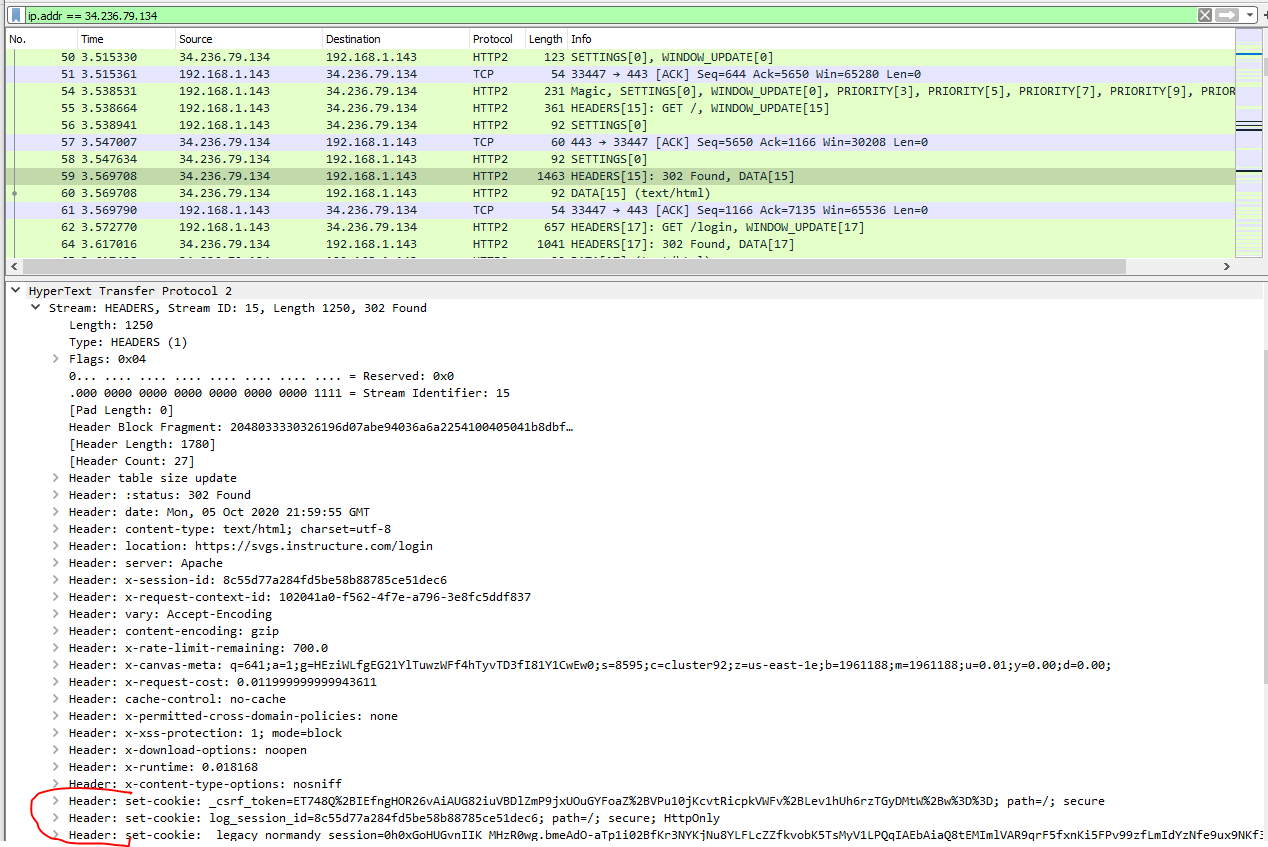
Hand in: Try decrypting an HTTPS conversation on your own. Remember that you have to have the environment key set to save your keys (<https://support.f5.com/csp/article/K50557518> ) and you have to set your Wireshark preferences for TLS to use your key. Don’t forget, and leave your Wireshark pointing at the keys for brcc.edu.pcapng! Hand in a screenshot of your Follow HTTP Stream.

# Optional

## A more complicated site

Real sites (as opposed to “ismycomputeronfire.com”) are harder to examine in Wireshark for several reasons:

1. They use the newer HTTP 2 instead of HTTP 1.1. HTTP 2 is much faster but is harder to dissect and read. For a brief overview of HTTP 2, see one of these articles: <https://www.ssl.com/article/an-introduction-to-http2/> or <https://www.advancedwebranking.com/blog/beginners-guide-to-http2/> .
2. Most sites now use TLS encryption, and HTTP 2 almost always uses TLS. You must configure your browser (Chrome and Firefox can do this, Edge cannot) to save its keys in an environment variable and configure Wireshark to use those keys.
3. Real sites tend to be complicated.

Here is an example from the SVGS Canvas site. It took a few minutes to dig this out of the haystack. The packet I’ve highlighted is one where the server tells the browser to set cookies that will be used for session control.  


If you want to examine a site like this, go for it! Be willing to spend some time puzzling things out and asking your instructor questions.

# Recap of items to Hand In

From Simple Web Page, step 7:  
Look at the User-Agent header. Does this match with the browser and operating system were you using when you browsed to the page?  
(If you are confused about why your browser is claiming to have multiple identities in its User-Agent header, read about the curious and funny history of the User-Agent here  
 <https://techcommunity.microsoft.com/t5/discussions/funny-story-of-browser-wars-from-the-beginning-of-the-time-user/m-p/1079832> )

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 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.