Decoding TLS from your browser using Wireshark

Some browsers can log the session keys they create as they visit TLS web sites. At this writing, Chrome and Firefox have this capability and Edge does not. Wireshark can use those session keys to show you decrypted versions of your packet capture. This is nearly the only way to examine HTTP/2 traffic since it is nearly always encrypted.

Note: A common tool used by web penetration testers to monitor and change encrypted traffic from their browser is Burp Suite. You configure your browser proxy all traffic through Burp and configure your browser to trust anything with Burp’s certificate. The community version of Burp Suite is installed on Kali Linux by default. Burp is beyond the scope of this little article, however.

# Configuration

The site <https://redflagsecurity.net/2019/03/10/decrypting-tls-wireshark/> has an excellent description of how to configure your Operating System (OS) and Wireshark to decrypt TLS. In the OS configuration (both Linux and Windows) you create an environment variable called SSLKEYLOG that gives the path to where the keys should be stored. If Chrome or Firefox see that environment variable, they will store the keys there. (Technically, the keys are “pre-master secrets.” You can read more about that here. <https://wiki.wireshark.org/TLS#Using_the_.28Pre.29-Master-Secret>) The Wireshark configuration just requires you to enter the location of your SSLKEYLOG file into your Wireshark preferences.

Note: There is one error in the <https://redflagsecurity.net/2019/03/10/decrypting-tls-wireshark/> article. Earlier versions of Wireshark (before v. 3.0) called the TLS preferences “SSL”, and the screenshot in the article reflects this. Current versions of Wireshark now use the term TLS, so the last configuration screenshot should look like this:

Graphical user interface, text, application

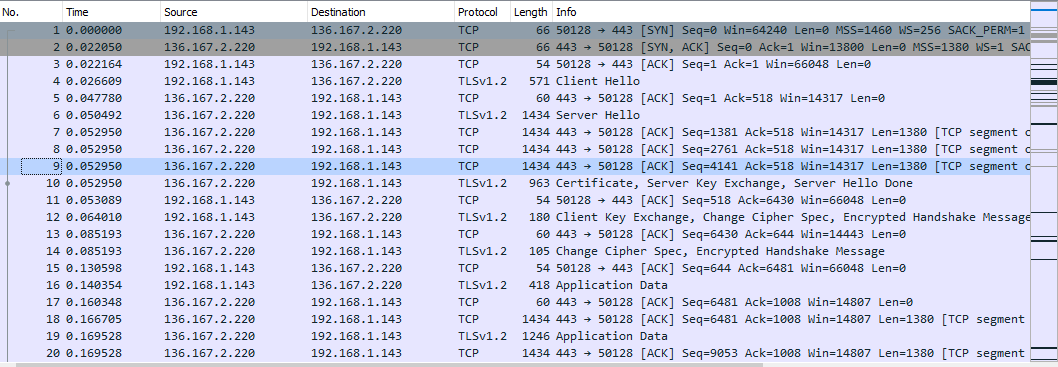
Description automatically generated

With that in mind, configure your OS and Wireshark to decrypt your browser’s TLS sessions.

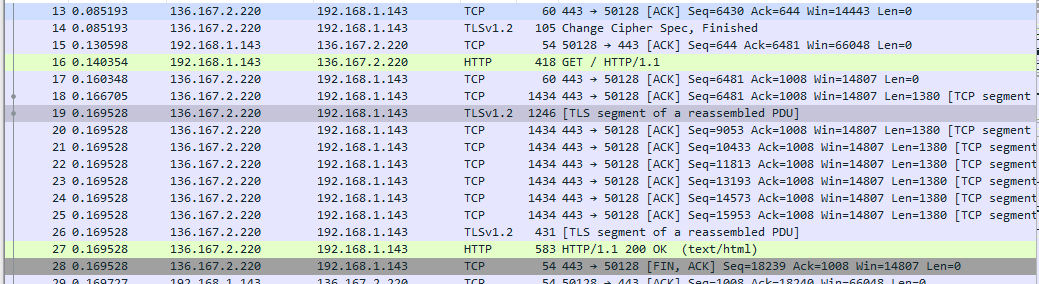
# Examples of Decrypted TLS

## HTTP/1.1 encrypted with TLS 1.2

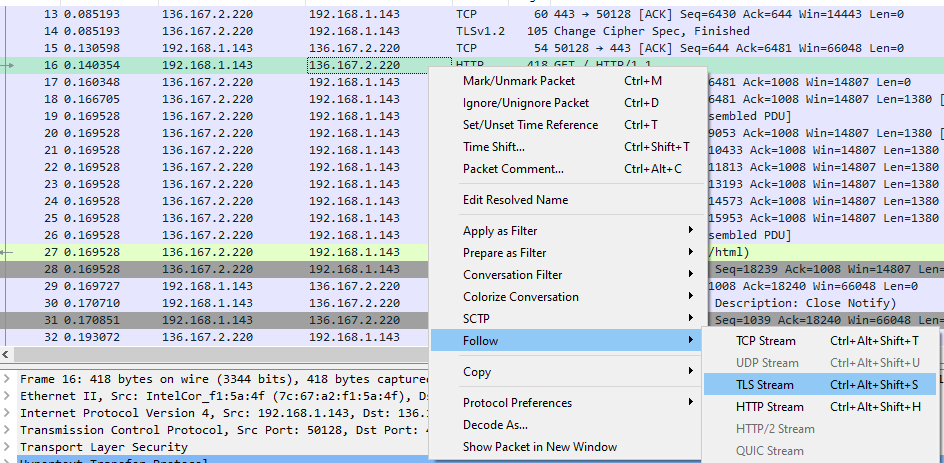
The following screenshot was taken from a packet capture of https://www.bc.edu. Before the path to the SSLKEYLOGFILE is configured in the Wireshark preferences, there is nothing but encrypted traffic. You can follow along with this example by opening the file   
“[www.bc.edu-withkeys.pcapng](http://www.bc.edu-withkeys.pcapng)”. It has been saved with the keys included.



The SSLKEYLOGFILE environment variable had been created long before the browser visited the web site, and before the packet capture was taken. (see <https://redflagsecurity.net/2019/03/10/decrypting-tls-wireshark/> ) Once the preference was updated to point to the key log file, green decrypted packets appeared in Wireshark.

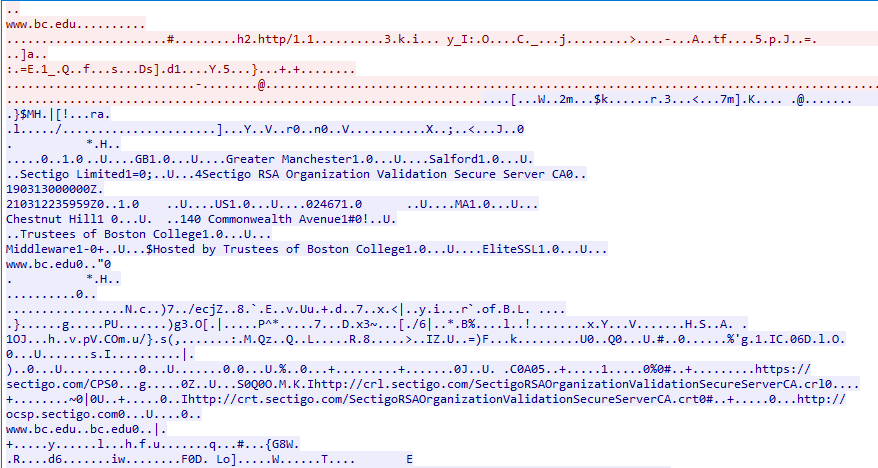


If you right-click on one of the decrypted packets, you will see new options in Follow. The choices for TCP Stream, TLS Stream, and HTTP Stream are now available.



### Follow TCP Stream

Follow TCP Stream will show just the encrypted traffic. There is some plain text visible, but that is only the unencrypted portion of the TLS handshake.



### Follow TLS Stream

Follow TLS Stream will show the unencrypted data. However, HTTP is often compressed with gzip. In this example we can read the headers, but much of the content is unreadable because it is compressed.



### Follow HTTP Stream

Follow HTTP Stream makes the entire content readable (as long as it is not binary data, like images.)



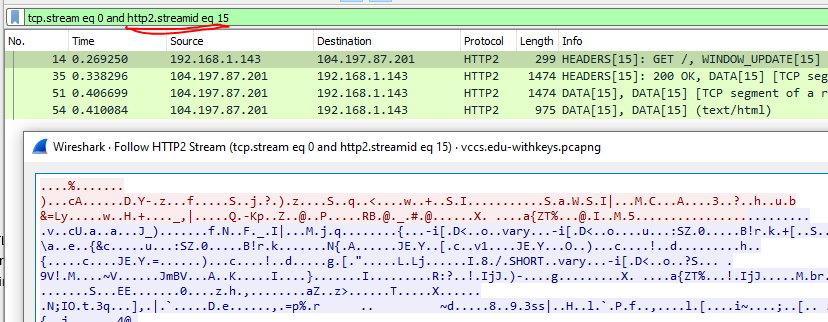
## HTTP/2 encrypted with TLS 1.3

Other than the handshake being shorter, and RSA not allowed for key exchange, the packet capture of TLS 1.3 does not look much different from TLS 1.2. There is a huge difference between HTTP/1.1 and HTTP/2. HTTP/1.1 tends to open several connections as it downloads different parts (images, etc.) of a site. To speed downloads, HTTP/2 opens one connection and then uses streams within that connection to perform its various tasks. Here are some good links to learn about HTTP/2:  
<https://developers.google.com/web/fundamentals/performance/http2>  
<https://kinsta.com/learn/what-is-http2/>

You can follow along with this example by opening the file “vccs.edu-withkeys.pcapng”. It has been saved with the keys included.

## Follow Stream

The first two options, Follow TCP Stream and Follow TLS Stream, work just as they did before. The Follow HTTP/2 stream works somewhat differently. Now it follows one stream withing the HTTP/2 connection, not the entire connection as in HTTP/1.1. It also does not seem to do as well in rendering the plain text version of the connection in the Follow Stream window.



However, the packet data pane still shows the decrypted data. Much of the basic information from HTTP/1.1 is there.

