Intro

Most IT people are somewhat familiar with Wireshark. It is a traffic analyzer, that helps you learn how networking works, diagnose problems and much more.
One of the problems with the way Wireshark works is that it can’t easily analyze encrypted traffic, like TLS. It used to be if you had the private key(s) you could feed them into Wireshark and it would decrypt the traffic on the fly, but it only worked when using RSA for the key exchange mechanism. As people have started to embrace forward secrecy this broke, as having the private key is no longer enough to derive the actual session key used to decrypt the data. The other problem with this is that a private key should not or cannot leave the client, server, or HSM it is in. This lead me to coming up with very contrived ways of man-in-the-middle myself to decrypt the traffic (e.g. sslstrip or mitmproxy).

**Session Key Logging to the Rescue!**

Well my friends I’m here to tell you that there is an easier way! It turns out that Firefox and Chrome both support logging the symmetric session key used to encrypt TLS traffic to a file. You can then point Wireshark at said file and presto! decrypted TLS traffic. Read on to learn how to set this up.

**Setting up our Browsers**

We need to set an environmental variable.

**On Windows:**

Go into your computer properties, then click “Advance system settings” then “Environment Variables...”
Add a new user variable called “SSLKEYLOGFILE” and point it at the location that you want the log file to be located at.

On Linux or Mac OS X:

```bash
$ export SSLKEYLOGFILE=~/path/to/sslkeylog.log
```

You can also add this to the last line of your

```bash
~/.bashrc
```

on Linux, or

```bash
~/.MacOSX/environment
```

on OS X so that it is set every time you log in.

The next time that we launch Firefox or Chrome they will log your TLS keys to this file.
Edit: If you are having trouble getting it to work on OS X take a look at the comments below. It seems that Apple has changed how environmental variables work in recent versions of OS X. Try launching firefox and wireshark within the same terminal window with,

```bash
# export SSLKEYLOGFILE=/Users/username/sslkeylogs/output.log
# open -a firefox
# wireshark
```

Thanks Tomi for sharing this.

**Setting up Wireshark**

You need at least Wireshark 1.6 for this to work. We simply go into the preferences of Wireshark

Expand the protocols section:
Browse to the location of your log file

The Results
This is more along the lines of what we normally see when look at a TLS packet,
This is what it looks like when you switch to the “Decrypted SSL Data” tab. Note that we can now see the request information in plain-text! Success!

**Conclusion**

I hope you learned something today, this makes capturing TLS communication so much more straightforward. One of the nice things about this setup is that the client/server machine that generates the TLS traffic doesn’t have to have Wireshark on it, so you don’t have to gum up a clients machine with stuff they won’t need, you can either have them dump the log to a network share or copy it off the machine and reunite it with the machine doing the packet capture later. Thanks for stopping by!

**References:**

Mozilla Wiki

Imperial Violet

jSSLKeyLog

Photo Credit: Mike