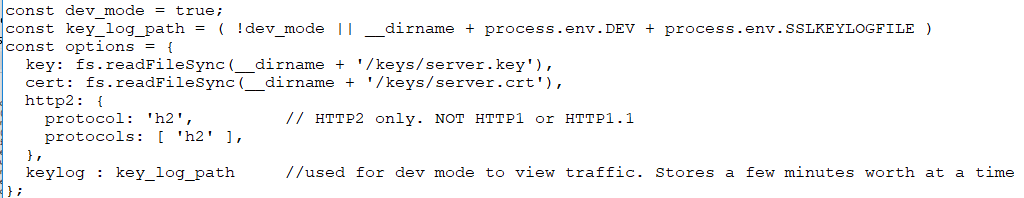
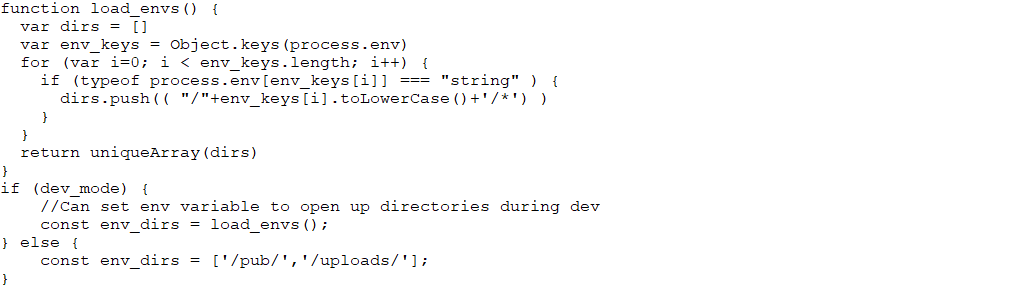
Objective--Network Traffic Forensics (Part 3)

# Solution (or part of it)

The interesting parts of the app.js file are here.  




We are looking for the SSLKEYLOGFILE, according to [HTTP/2 Decryption and Analysis in Wireshark](https://www.youtube.com/watch?v=YHOnxlQ6zec). Sure enough, there is a line with exactly what we are looking for.  
const key\_log\_path = ( !dev\_mode || \_\_dirname + process.env.DEV + process.env.SSLKEYLOGFILE )

The environment variable is SSLKEYLOGFILE.

The function load\_envs() takes all the environment variables, converts them to lower case and pushes them into a list. That is strange, but maybe it is trying to make the code scalable as the [environment variables article](https://codeburst.io/process-env-what-it-is-and-why-when-how-to-use-it-effectively-505d0b2831e7) suggests. You had better be careful with your environment variables if you do that.

The if statement opens directories to all environment variables if dev\_mode is True. If dev\_mode is False it opens the directories put and uploads.

When we look back up to the constants, we find this, so the application is in dev\_mode.  
const dev\_mode = true;  
It appears our developer was not careful with the environment variables.

Therefore, the server should be opening a directory or file like the value stored in sslkeylogfile. Browsing to that directory gives us this, so it appears the file name is http2packalyzer\_clientrandom\_ssl.log.  


What a weird and wonderful (for attackers) that error message is!

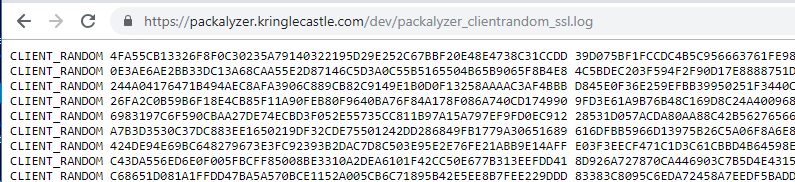
However, /opt/http2packalyzer\_clientrandom\_ssl.log/ looks strange. Let’s go back to the constant that created that string.  


We know that dev\_mode is True, so !dev\_mode is False. The OR ( || ) is using short-cut execution. If the first part of the OR is True, the entire statement is True so the second part does not need to be executed. If the first part is False, the second part must be evaluated to determine if the statement is True or False. The second part is only executed when the first part is false.

Therefore, this is executed.  
\_\_dirname + process.env.DEV + process.env.SSLKEYLOGFILE

The internal function \_\_dirname gives the current directory. Then process.env.DEV must give the value that the DEV environment variable points to. Finally, process.env.SSLKEYLOGFILE gives the value of the SSLKEYLOGFILE. So,  
 \_\_dirname is /opt/  
 process.env.DEV is http2  
 process.env.SSLKEYLOGFILE is packalyzer\_clientrandom\_ssl.log

The missing ‘/’ in the code we just examined makes http2 look like part of the file name, but it is not. The file name is packalyzer\_clientrandom\_ssl.log.

We didn’t find the file in the /pub directory. The /opt/http2/ directories are local to the server, not what is published by the webserver. Let’s hope the web directory is dev/; after all, there is a DEV environment variable.  
<https://packalyzer.kringlecastle.com/dev/packalyzer_clientrandom_ssl.log/>  


Yes! Copy the contents of the page and paste it into a text editor. We can move on to decrypting packets. Finally!

# Hand In

Follow the steps in [Chris’ video](https://www.youtube.com/watch?v=YHOnxlQ6zec) and see what you can glean from the pcap file you downloaded from Packalyzer. It would be nice to find an answer to the objective, but if not, credentials are always good!

Note: Download the packet capture file, and then grab the SSLKEYLOGFILE soon afterwards. If there’s a large time delay, the keys may not match the capture file.

1. Is there a user name and password in the pcap file?
2. If you find credentials, where would be a good place to use them? If you are lucky, you will find something with ‘secret’ in the name. What is it?