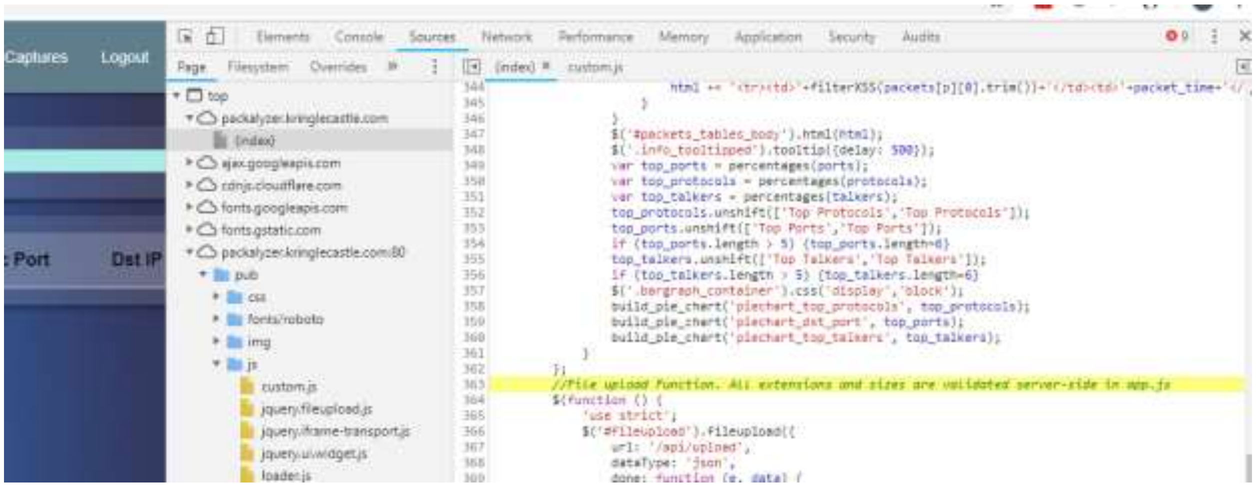
Objective--Network Traffic Forensics (Part 2)

Solution (or part of it)

Following Mary's hint about comments in the HTML index, we see this.

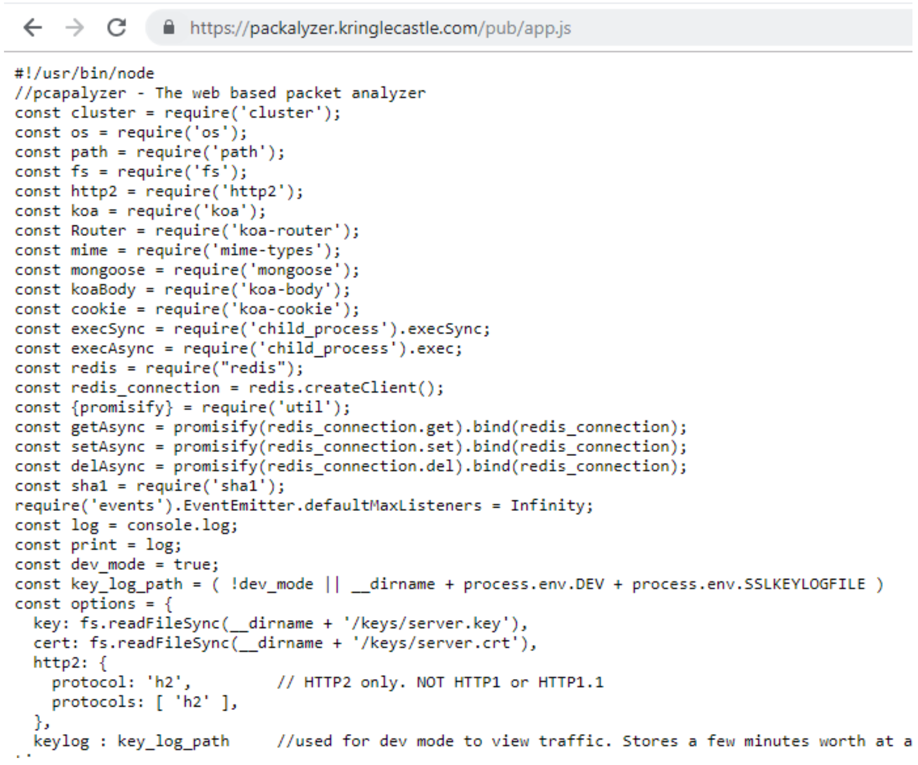


```
                html += '<tr><td>'+filterXSS(packets[p][0].trim())+'</td><td>'+packet_time+'</
            }
        }
        $('#packets_tables_body').html(html);
        $('.info_tooltipped').tooltip({delay: 500});
        var top_ports = percentages(ports);
        var top_protocols = percentages(protocols);
        var top_talkers = percentages(talkers);
        top_protocols.unshift(['Top Protocols','Top Protocols']);
        top_ports.unshift(['Top Ports','Top Ports']);
        if (top_ports.length > 5) {top_ports.length=6}
        top_talkers.unshift(['Top Talkers','Top Talkers']);
        if (top_talkers.length > 5) {top_talkers.length=6}
        $('.bargraph_container').css('display','block');
        build_pie_chart('piechart_top_protocols', top_protocols);
        build_pie_chart('piechart_dst_port', top_ports);
        build_pie_chart('piechart_top_talkers', top_talkers);
    }
};
//File upload Function. All extensions and sizes are validated server-side in app.js
$(function () {
    'use strict';
    $('#fileupload').fileupload({
        url: '/api/upload',
        dataType: 'json',
        done: function (e, data) {
```

```
//File upload Function. All extensions and sizes are validated server-
side in app.js
```

Now we know the name of a file in the server-side source code.

In the same screenshot (above) we can see a little of the directory structure of the server. Perhaps the apps.js file lives in /pub? Bingo!



```
#!/usr/bin/node
//pcapalyzer - The web based packet analyzer
const cluster = require('cluster');
const os = require('os');
const path = require('path');
const fs = require('fs');
const http2 = require('http2');
const koa = require('koa');
const Router = require('koa-router');
const mime = require('mime-types');
const mongoose = require('mongoose');
const koaBody = require('koa-body');
const cookie = require('koa-cookie');
const execSync = require('child_process').execSync;
const execAsync = require('child_process').exec;
const redis = require("redis");
const redis_connection = redis.createClient();
const {promisify} = require('util');
const getAsync = promisify(redis_connection.get).bind(redis_connection);
const setAsync = promisify(redis_connection.set).bind(redis_connection);
const delAsync = promisify(redis_connection.del).bind(redis_connection);
const sha1 = require('sha1');
require('events').EventEmitter.defaultMaxListeners = Infinity;
const log = console.log;
const print = log;
const dev_mode = true;
const key_log_path = ( !dev_mode || __dirname + process.env.DEV + process.env.SSLKEYLOGFILE )
const options = {
  key: fs.readFileSync(__dirname + '/keys/server.key'),
  cert: fs.readFileSync(__dirname + '/keys/server.crt'),
  http2: {
    protocol: 'h2',         // HTTP2 only. NOT HTTP1 or HTTP1.1
    protocols: [ 'h2' ],
  },
  keylog : key_log_path     //used for dev mode to view traffic. Stores a few minutes worth at a
```

The next step

If you examine app.js carefully, you will find that it does very strange things. There is a constant that seems to be exactly what we are looking for. By the way process.env in JavaScript makes environment variables available to the code, as described here.

The JavaScript has big blobs of binary data, but fortunately we can ignore them. They make good signposts, though. Look at the code just above the first blob of binary. It does really weird things with environment variables and directories.

Try looking for the file you want using the hints in the code and the Packalyzer URL. If you are lucky you will find the weird and descriptive error that Mary talks about. Then you can use that in the URL to download the key file. You may need to look at the constants in the code to guess which directory the file is in though.

Download the file.

Hand In

1) What is the environment variable that points to the file you need?

2) Is the server running in dev_mode?

3) Based on 404 errors from the server, what is the actual name of the file?

4) If you look carefully at the code (in the constant section) that builds the path to the file you want, you will see that it is missing a '/'. Does this affect your answer for question 3?

5) When you download the file, you won't find it in the /pub directory (/pub/filename won't work.) However, the constant should give you a hint about what directory should be. What is in the first line of the file we need (just to see if you were able to download it.)