

Visualising Malicious Network Activity

by Aled Owain Mason
Student no: 1130538

CM3203: One Semester Individual Project (40 credits)

Supervised by Prof. Omer F. Rana

Moderated by Mr. Michael Daley

A project submitted in partial fulfilment of
BSc Computer Science with Security and Forensics



School of Computer Science and Informatics
May 2014

Abstract

Many large organisations now implement Intrusion Detection Systems (IDSs) to mitigate the increasing cyber-threat. These IDSs produce log files containing information about each threat that the IDS has prevented. These logs are an *important resource*, as they contain key information about the types of threats that are emerging, as well as useful metadata linked to each entry. However understanding this data is a difficult task, as there are thousands of cyber-attacks being launched every day towards large networks which implement an IDS.

The aim for this project is to analyse and interpret these log files taken from Cardiff University Information Services (INSRV) IDSs. This analysis would aim to answer some research questions surrounding these large datasets. To do this, a software environment was developed to efficiently process this data and to create dynamic visualisations “on-the-fly”. This project was developed by using a variety of tools including D3.js, a JavaScript visualisation library.

Acknowledgements

With many thanks to my supervisor Prof. Omer Rana for his guidance and support during this project. Also a big thanks to Mr. Damian Southard for discussing project ideas and providing the network data; without whom this project would not have been possible. Lastly I would also like to thank my family for their ongoing support throughout my studies at University.

Contents

1	Introduction	1
1.1	Preface	1
1.2	Data Visualisation	1
2	Background	2
2.1	Cyber Threats	2
2.1.1	Port Scanning	2
2.1.1.1	Stealth SYN Scan	2
2.1.2	Malware	3
2.1.2.1	Virus	3
2.1.2.2	Spyware	3
2.1.3	Denial of Service	3
2.1.3.1	SYN Flood	4
2.1.3.2	Distributed DoS	4
2.1.4	Vulnerabilities & Exploitation	4
2.1.4.1	Buffer Overflows	4
2.1.4.2	SQL Injection	4
2.2	Intrusion Detection	4
2.2.1	Pattern Matching	5
2.2.2	Statistical Anomaly	5
2.3	The Problem	5
2.3.1	Analysing the Dataset	6
2.4	Stakeholders	7
2.5	Existing Solutions	7
3	Specification & Design	8
3.1	Overview	8
3.2	Changes from Initial Plan	8
3.3	Visualisation Designs	8
3.4	System Description	9
3.5	Use Cases	9
3.5.1	Core Requirements	9
3.5.2	Real Time Functionality	11
3.6	UML Sequence Diagrams	14
3.7	Database Design & Schema	16
4	Implementation	17
4.1	Acknowledgements	17
4.1.1	Git	17
4.1.2	Dropbox	17
4.1.3	Bootstrap	17
4.1.4	MAMP	17
4.1.5	JQuery	18
4.1.6	D3.js	18
4.1.6.1	bl.ocks.org	18

4.2	Critical Source Code Explained	19
4.2.1	Uploading Log Files	19
4.2.2	IFrames	19
4.2.3	Requesting Data	19
4.2.4	Pie Chart	20
4.2.5	World Map	21
4.3	Problems Encountered	23
4.3.1	PHP Memory Limit	23
4.3.2	Pie Chart Labels	23
4.3.3	Performance Issues	24
4.3.4	Returning JSON	24
4.3.5	D3 Errors	25
4.3.6	CSS Styling Issues	26
4.3.7	Rickshaw	26
5	Results and Evaluation	27
5.1	Visualisation Examples	28
5.2	Software Limitations	31
5.3	Evaluation of Approach	32
5.4	Feedback from INSRV	32
6	Future Work	34
6.1	Real Time Functionality	34
6.2	CVE Database	34
6.3	Additional Data Sources	35
6.4	Open Source	35
6.5	CEReS Project	36
7	Conclusion	37
	Reflections on Learning	38
	Appendices	40
	References	40

1 Introduction

1.1 Preface

With a huge increase of malicious cyber-attacks in recent years [1], cyber-security has been of critical importance to large organisations which operate large IT infrastructure. These organisations often implement a countermeasure in the form of an Intrusion Detection System (IDS) to mitigate these cyber-threats. These IDSs are capable of producing log files, containing information about each cyber-threat that was prevented. However understanding this data is a difficult task, as a log file usually contains thousands of entries.

The aim for this project will be to build a software tool where a user can efficiently analyse and interpret these kinds of log files. Cardiff University Information Services (INSRV) have kindly supported this project by providing numerous datasets to experiment with, taken from their IDSs. This project will also aim to answer some research questions surrounding this data:

- What is the most common type of attack launched towards a given network?
- What time of the day is the busiest with respect to malicious traffic?
- Which country do most cyber-attacks originate from?
- Which sub-networks do most cyber-attacks target?

1.2 Data Visualisation

“Information overload” is a term used countless times in our modern society, although for this project its quite appropriate. For example, in one log file provided by INSRV; over 400,000 threats were recorded. Interpreting this data in its raw format will be difficult and troublesome. “Fortunately, we humans are intensely visual creatures” [2]; so by designing and using various visualisation techniques, we can derive firm conclusions from a given dataset. In a TED Talk in 2010, David McCandless, an “information designer” and author of “Information is Beautiful”, speaks about data visualisation as a form of “knowledge compression” [3].

Of course creating these visualisations for our dataset will still be a tedious task by hand [2]. So with this in mind, this project will aim to build a software environment, capable of creating various visualisations “on-the-fly”, in order to communicate information effectively to the end-user.

2 Background

In March 1989, Sir Tim Berners Lee proposed the idea of a “web of notes with links between them” [4]. Today this idea is the backbone of the World Wide Web, driven by the internet. It has become a multifaceted environment with millions of users everyday, permeating all kinds of industries and government.

With these developments however, there are an increasing amount of cyber-attacks launched towards inter-connected networks for malicious purposes [1]. Because of this, many new technologies have been developed, such as Intrusion Detection Systems and Firewalls.

Cyber-crime is estimated to cost the UK “£27bn per annum” [5] from a report by Detica - a world leading cyber-security firm. These are committed by “foreign intelligence services” [5] which participate in industrial espionage, including intellectual property theft, to improve their domestic “industries and economy” [5]. There are also “organised crime networks” who solicit online scams [5] and “hacktivists” [6] who engage in illegal computer activities to promote a campaign of activism.

In 2011, the head of GCHQ Iain Lobban reported that cyber-attacks were at a “disturbing level” [7] and were attempting to “steal British ideas and designs...to gain commercial advantage or to profit from secret knowledge of contractual arrangements” [7]. So what are these cyber-attacks and how do they work?

2.1 Cyber Threats

2.1.1 Port Scanning

Port scanning is usually the first order of business when it comes to cyber-attacks. Port scanning is a way to gain information about which network ports are accepting connections [8]. With this information, malicious hackers know where and how to transmit their threats. There are thousands of ports available to establish a connection which usually all map to a unique application or service, e.g. HTTP runs on port 80. Although attempting to do this is quite difficult as many new security technologies, including IDSs, can detect when a port scan is taking place. However there have been efforts to circumvent these countermeasures, such as a stealth SYN scan.

2.1.1.1 Stealth SYN Scan

To understand how this port scan works, we must first discuss a TCP/IP connection. Transmission Control Protocol (TCP) “maintains reliable connections” [8] as it ensures that all data that was sent, is accurate and unchanged. It does this by first establishing a TCP connection using a 3-way handshake (see Figure 1). This involves sending 3 unique packets. First an initiator would send a SYN (synchronise) packet to the listener. The listener would then reply with a SYN-ACK packet; this message acknowledges the initiators first packet and sends another SYN packet. Finally the initiator would reply with an ACK (acknowledgement) packet.

This type of port scan will attempt to open a TCP connection with the listener by sending a SYN packet. If a SYN-ACK packet returns, then it must be “accepting connections” [8]. The attacker will then send a RST (reset) packet, which will close the

open connection on the listeners machine, to prevent an accidental DoS attack (explained below).

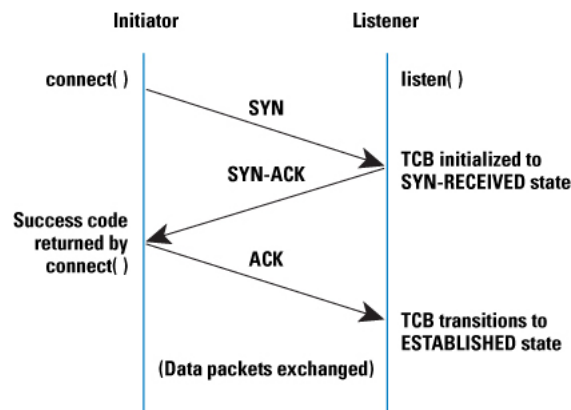


Figure 1: A TCP 3-way handshake [9].

2.1.2 Malware

Malicious software, often shortened to “malware”, is a computer program “designed for some malicious purpose” [10], often to disrupt a computer system’s operations. Malware comes in many different varieties, including viruses and spyware.

2.1.2.1 Virus

A computer virus is a program that is able to create copies of itself and spread across various computer networks [11]. These viruses often contain a “payload” which carries out malicious actions [10]. These can vary depending on the seriousness of the program, but typically they “damage data files” [11].

2.1.2.2 Spyware

Spyware is a piece of malicious software designed to gather information about the user without their knowledge [12]. This could include a user’s “keystrokes, screenshots, authentication credentials... and other personal information” [13]. This can lead to a variety of other crimes including “identity theft and credit card fraud” [13].

2.1.3 Denial of Service

Denial of Service (DoS) attacks aim to disrupt a networked service or resource to prevent legitimate users from accessing it. This method is most common among “hacktivists” who wish to prevent legitimate access to a particular website. In 2008, the online hacktivist group “Anonymous” made headlines as they targeted the Church of Scientology through denial of service attacks [6]. Some of the most common denial of service attacks are described below.

2.1.3.1 SYN Flood

SYN Flooding exploits the TCP 3 way handshake design by sending the listener (e.g. a web server) many SYN requests (usually from spoofed IP addresses), in an attempt to half-open a TCP connection with the listener. As most servers will have a “finite table that can only track so many incoming connections” [8], this will slow down the server and prevent legitimate users from accessing the resource.

2.1.3.2 Distributed DoS

A distributed denial of service attack is also a common type of cyber-attack which involves using many hosts to attack a single resource. As denial of service attacks aim to overload the target system, a distributed DoS (commonly known as DDoS) attack is usually more dangerous because typically the attacker(s) hold a greater bandwidth compared to the target system. This approach is common among hacktivist groups and organised criminals but is also carried out by involuntary victims.

This approach is called a “Botnet” and involves compromising, usually through a virus, a number of innocent people’s machines and turning them into “bots” or “zombies”. These machines are then under the control of the attacker who subsequently uses them to carry out a DoS attack towards another online resource/ server [8].

2.1.4 Vulnerabilities & Exploitation

Vulnerabilities and exploitation techniques may be the most dangerous of all cyber-threats, as these are actually “flaws or oversights in the design of the program” [8] and attacks like these are made by knowledgeable hackers. One example of such a vulnerability is the “Heartbleed” bug found in the popular OpenSSL library. This allowed a knowledgeable hacker to steal information intended to be protected, by exploiting the “SSL/ TLS encryption used to secure the internet” [14].

2.1.4.1 Buffer Overflows

A buffer overflow is simply when a program allows data to be inserted into a buffer which exceeds the size originally intended for that buffer [8]. This usually results in that program crashing, as when excess data is inserted into the buffer, it overwrites adjacent memory’s data. This is quite a serious vulnerability in a program’s implementation, as an attacker could target the system’s availability and gain control of the system [15].

2.1.4.2 SQL Injection

When interacting with a database online, it is most likely implemented with Structured Query Language (SQL). SQL injections are a method for hackers to interact directly with the database, through a vulnerable access method like a web form [10]. This could lead to unauthorised access or deletion of the data stored on the database.

2.2 Intrusion Detection

With all these security issues and an emerging global cyber-threat, many large networks have implemented Intrusion Detection Systems as a way to prevent these malicious attacks. An IDS automates “the process of monitoring events in systems or networks to

detect intrusions” [10]. A networked IDS is placed between the source and destination hosts and monitor packets bound to access the inner-network, in an attempt to establish if the network packet is a malicious threat [10]. It does this by using a variety of techniques including pattern matching and statistical anomaly.

2.2.1 Pattern Matching

Signature based detection, also known as pattern matching, is a method used to detect cyber-threats. This method involves looking at “traffic and behaviour that matches the patterns of known attacks” to a “signature database” [16]. However this method is effective only if the attack has already been recorded in the database and cannot detect new emerging threats.

2.2.2 Statistical Anomaly

Statistical based intrusion detection aims to identify malicious activity through “systematically analysing audit trail data” [17]. This involves measuring a “baseline profile” [16] to estimate a normal network connection. Then monitoring connections for “actions that are outside of those normal parameters” [16]. This method is different from pattern matching as it can detect threats that are not already well-known and can prevent “zero-day” attacks (threats where “no patch exists” [18]) from having an impact on a computer system.

2.3 The Problem

These IDSs have the ability to produce log files containing information about each threat that the IDS has stopped. These logs are an *important resource*, as they contain key information about the types of threats that are emerging, as well as useful metadata linked to each entry.

However making sense of this data is a difficult task as each log file contains an extremely large amount of fields and entries (see Figure 2). This project aims not only to solve this problem, but attempts to answer some research questions surrounding this data.

Figure 2: Sample of a raw dataset.

The datasets for this project were taken from INSRV intrusion detection systems. These IDSs are manufactured by *Palo Alto Networks*¹, a cyber-security firm which makes various products to counter cyber-threats. To analyse each entry in the dataset, INSRV have also provided a brief manual [Appendix A] on how to interpret these log files. This manual contains information on what information each field contains. Each threat entry has 42 respective fields, shown below are the ones of interest for this project, with a brief description of each.

- ¹Palo Alto Networks available at: <https://www.paloaltonetworks.com>

These kind of log files are a good source of information for analysing malicious network activity. They contain comprehensive information regarding the metadata (including timestamps, IP addresses and threat type) of countless cyber-threats. However this type of data is also very restricted; as we are unable to analyse the entire network session - which contains low level pcap data (raw network packets).

Also note that we are unable to *detect* malicious attacks from these datasets - as our dataset only provides an indicator of a threat occurring. This would involve analysing additional datasets and further research and development into intrusion detection.

2.4 Stakeholders

A potential stakeholder for this project would be network administrators/ engineers and other professionals working in a NOC (network operations centre) environment. This group of professionals would be interested in a way to quickly derive facts and statistics about the network they manage.

Damian Southard who works in Cardiff University's INSRV as a Security Engineer has mentioned a visualisation tool like this will be beneficial for himself in a NOC environment and for data analysis in general. He also mentioned that this tool is useful for his management; as they like to include graphs and visualisations for their own reports.

Providing an easy way to derive facts and generate visualisations will also be of interest to groups of people who want to clearly illustrate the statistics of these cyber-attacks. One such group is CESG (Communications Electronic Security Group)², the "UK Government's National Technical Authority for Information Assurance" [19]. They provide advice to the UK government departments, "the wider public sector" and the "UK's Critical National Infrastructure" [19].

2.5 Existing Solutions

From my research, there are only a handful of software tools that already do this. Kibana³, is a software program available from *elasticsearch* and is able to visualise, in real-time, logs and other time-stamped data. Although this software is quite difficult to install, especially for a person without a technical background. This project aims to offer a simpler way to provide visualisation capabilities to an end-user, while also tailoring the results for malicious network data.

²CESG available at: <https://www.cesg.gov.uk/>

³Kibana available at <http://www.elasticsearch.org/overview/kibana/>

3 Specification & Design

3.1 Overview

The aim for this software is to be functional but also simple to use. For this reason I have decided to create a web application. With many different programming languages available to develop web applications and with each containing a variety of different built-in and open-source libraries, it is a good choice for this project. Also websites are simple to use, as they do not require the user to install any specific software packages to benefit from their applications.

For the implementation, I plan on using a MySQL database to store each log file, so that it's possible to query the dataset using PHP. With this returned data I will then use a JavaScript library (such as D3.js) to create each of the visualisations needed.

This system will initially be designed to work with INSRV data; thus will be limited to only be compatible with *Palo Alto Networks* IDSs which run *PAN-OS* and generate a “threat” log file.

3.2 Changes from Initial Plan

In my initial plan, I proposed developing visualisation capabilities for real-time network activity and to link each log entry with the common vulnerabilities and exposures⁴ database. After some thought however, these objectives will now be extended requirements and will only be completed if the project's overall progress is ahead of schedule.

3.3 Visualisation Designs

To create various visualisations for the research questions originally intended for this project, we must first discuss what kind of visualisations are suitable for each.

What is the most common type of attack launched towards a given network?

This question only deals with one variable, so is suited toward a simple bar or pie chart.

What time of the day is the busiest with respect to malicious traffic? As this question needs to consider multiple variables, i.e. time and number of threats, this is suitable for a bar/ line chart. With time on the x-axis and number of threats on the y-axis. We could also use a stacked bar or multi-series line chart to visualise different kinds of threats.

Which country do most cyber-attacks originate from? This question could be visualised using a simple bar graph, although there is also the option of using a world map visualisation, with a colour gradient to determine a greater amount of threats originating from a particular country.

Which sub-networks do most cyber-attacks target? This question could also be visualised using a pie chart, although we could also graph the network topology.

⁴CVE database available at: <http://cve.mitre.org/>

3.4 System Description

To visualise a raw dataset, the user must first be able to upload their log files to the database. Once this is completed, the user should be redirected to a “dashboard” environment whereby the user can immediately view some statistics about their log file. The user should then be able to navigate the website to view the visualisations generated from their dataset.

If a user should want to return to the website at a later time, they will be given a unique identifier associated with their log file, so they can refrain from re-uploading their dataset multiple times. This ID can also be shared with other colleagues, to allow ease of access to the website.

As this software is being built as a website, I plan on using web technologies such as AJAX, to query the server and subsequently, the database. I also plan on keeping each visualisation separate, by using HTML IFrames.

3.5 Use Cases

After researching the problem area, I created the below use cases and communicated this with *Damian Southard* from INSRV. I wanted the software tool to be easy to use and provide a very simple way to analyse various log files. To do this I obviously needed the capability for a user to upload their own files. I also needed to view the log file itself and generate visualisations on-the-fly.

3.5.1 Core Requirements

Use Case:-	<i>Upload Log File</i>	
Description:	Basic Flow:	Alternate Flows:
User is able to upload a log file to the website.	1. User clicks the “Upload” button from the website. 2. User selects a log file from their machine. 3. Browser begins uploading the file to the server. 4. <i>include::</i> Server verifies integrity of uploaded file and sequentially writes entries to the database. 5. webpage re-directs to a new page, displaying a unique ID.	4A. Server cannot verify the file. 4B. The upload gracefully quits.
	Pre-Conditions:	Post-Conditions:
	None.	Log file is successfully uploaded to the database. User is re-directed to a new page.

Use Case:-	<i>View Log</i>	
Description:	Basic Flow:	Alternate Flows:
User is able to view a table containing their log file.	1. Server pulls the log file information from the database. 2. Log file entries are formatted and displayed on the website. 3. <i>extend::</i> User inputs a filter and presses the “Go” button. 4. Log file entries are updated to match the filter.	3A. User inputs an invalid filter and presses the “Go” button. 3B. The system informs the user of a problem.
	Pre-Conditions:	Post-Conditions:
	User has already logged into the system with their unique ID/ uploaded a log file.	Log file is successfully displayed on the user’s screen.

Use Case:-	<i>Enter Code</i>	
Description:	Basic Flow:	Alternate Flows:
User is able to login using their unique ID/ pre-generated code.	1. User clicks the “Enter a pre-generated code” button from the website. 2. User enters their unique ID and presses “Go”. 3. <i>include::</i> Server verifies if the unique ID exists on the system. 5. webpage re-directs to a new page.	3A. User inputs an invalid code and presses the “Go” button. 3B. The system informs the user of a problem.
	Pre-Conditions:	Post-Conditions:
	None.	User is re-directed to a new page containing the log file specified.

Use Case:-	<i>View Visualisations</i>	
Description:	Basic Flow:	Alternate Flows:
User is able to enter a visualisation environment for their IDS log file.	1. User clicks the “Visualisation” link from the website. 2. System begins pulling the log file information from the database and processes the data to create various visualisations.	n/a
	Pre-Conditions:	Post-Conditions:
	User has already logged into the system with their unique ID / uploaded a log file.	User is able to view various visualisations for their log file.

3.5.2 Real Time Functionality

Use Case:-	<i>Enter IDS Server Info</i>	
Description:	Basic Flow:	Alternate Flows:
User is able to login to the system, connecting their Intrusion Detection System.	1. User clicks the “Visualise Real-Time IDS Data” link from the website. 2. User enters their login credentials and presses the “Go” button. 3. <i>include::</i> System connects to the IDS. 4. website re-directs to a new page, displaying information about the IDS in real-time.	3A. System is unable to connect to the IDS. 3B. System informs the user of a problem.
	Pre-Conditions:	Post-Conditions:
	User holds correct credentials to login to their IDS.	User is logged into the system and is able to view information about their IDS.

Use Case:-	<i>View Real Time Log</i>	
Description:	Basic Flow:	Alternate Flows:
User is able to view the IDS threat log in real time.	1. Server pulls the log file information from the IDS. 2. Log file entries are formatted and displayed on the website. 3. <i>extend::</i> User inputs a filter and presses the “Go” button. 4. Log file entries are updated to match the filter.	3A. User inputs an invalid filter and presses the “Go” button. 3B. The system informs the user of a problem.
	Pre-Conditions:	Post-Conditions:
	User has already logged into the system using the “Visualise Real-Time IDS Data” option.	User can view their IDS threat log in real-time.

Use Case:-	<i>View Real-Time Visualisations</i>	
Description:	Basic Flow:	Alternate Flows:
User is able to enter a visualisation environment for their IDS threat log in real-time.	1. User clicks the “Visualisation” link from the website. 2. System begins pulling the log file information from the IDS and processes the data to create various visualisations.	n/a
	Pre-Conditions:	Post-Conditions:
	User has already logged into the system using the “Visualise Real-Time IDS Data” option.	User is able to view various visualisations from their IDS threat log in real-time.

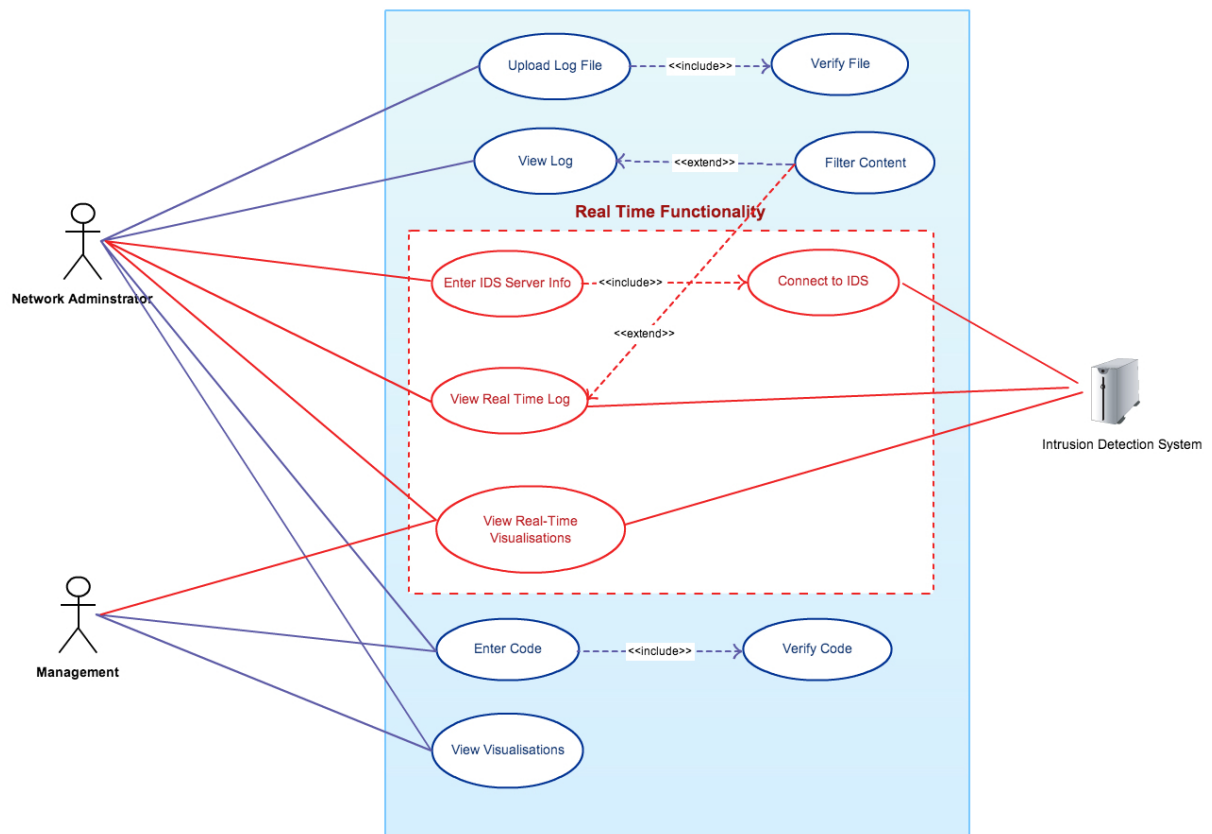


Figure 3: Use Case Diagram^a

^aImage generated from <https://creately.com/>

3.6 UML Sequence Diagrams

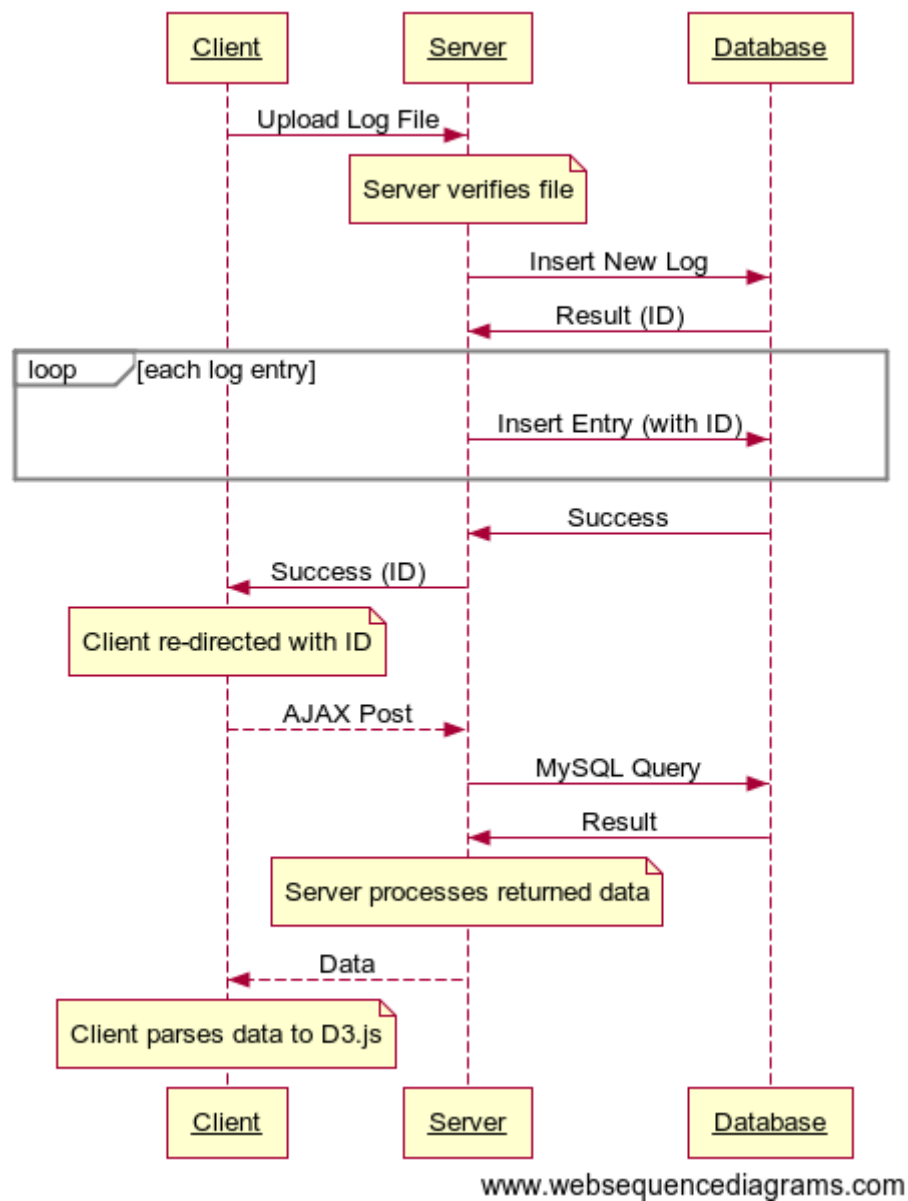


Figure 4: Basic Flow^a

^aImage generated from <https://www.websequencediagrams.com/>

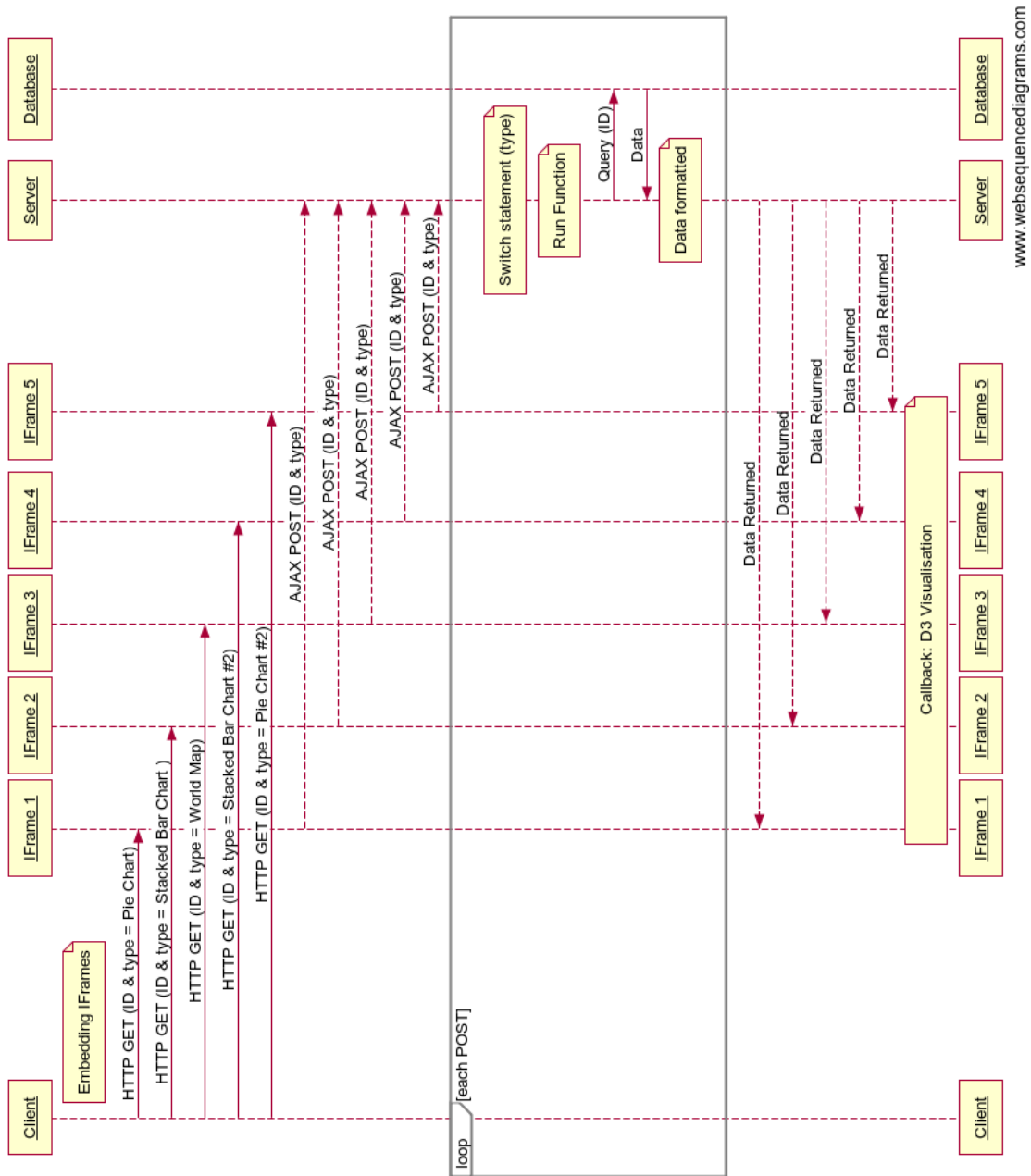


Figure 5: Sequence Diagram for Visualisation Page^a

^aImage generated from <https://www.websequencediagrams.com/>

3.7 Database Design & Schema

A MySQL database will be used to store each log file. As MySQL is a relational database, each log file will be added to multiple tables. The first table, *Log*, will contain the unique identifiers for each complete log file. The second table, *Description*, will contain the information for each entry inside a log file. To link these tables, the primary key field (*logid*) from the *Log* table will be the foreign key in the *Description* table. The *Description* table will then have a primary composite key, consisting of *logid* and *entryid*, where *entryid* is an incrementing integer beginning at 0 for each log file.

As this project is being developed to be compatible with INSRV data, the database will need to be configured for the Intrusion Detection Systems that INSRV operate. These are made by *PaloAlto Networks*. So by using the PaloAlto Networks PAN-OS Tech Note document [Appendix A] (a user guide describing how to interpret a log file), I have been able to design the *Description* table so that each column corresponds to the field name as it appears in the log file (see Figure 6).

Planning ahead to make this system compatible with other kinds of IDS log files, I have added in the schema below a “type” field, so that in future, the database could distinguish different kinds of log files, which would link to different tables - determined by their type. This way, the system could be extended to support numerous log file types.

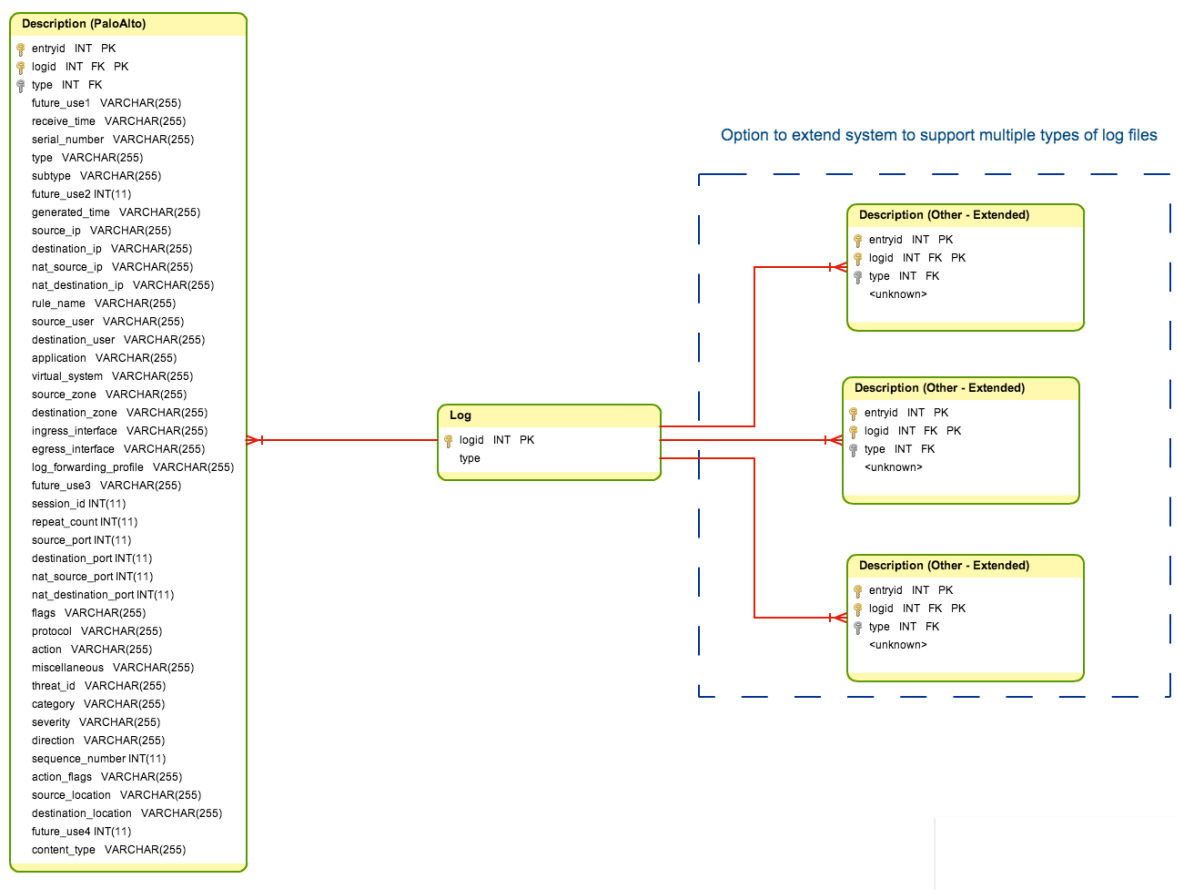


Figure 6: Database Design Diagram^a

^aImage generated from <https://creately.com/>

4 Implementation

4.1 Acknowledgements

Before discussing the development of this project specifically, I wanted to acknowledge the underlying technologies that made this system possible and mention the supporting applications that aided with general project progression.

4.1.1 Git⁵

Git is a “powerful, flexible and low-overhead version control tool” [20]. I used this tool to create a hierarchical structure of nodes, each with a unique repository. With this feature, I was able to “revert” to a previous “commit” (version), or create a new “branch” to add new features or fix bugs, with the option of switching the branch I was working on, creating a very stable environment for project progression.

4.1.2 Dropbox⁶

This project is backed-up using Dropbox, a free online file hosting service. This service provided a seamless method for backing up critical source-code. After installation, a special folder appears on a user’s machine. Each file a user adds to their Dropbox folder will automatically be synchronised to “Dropbox’s secure online servers” [21], thus providing an online backup.

4.1.3 Bootstrap⁷

Bootstrap is a “popular front-end framework” [22] for developing web applications. I chose to implement Bootstrap as a front-end framework for my website as it provided open-source HTML templates, CSS stylesheet and a JavaScript library. It also allowed me to create an aesthetically pleasing website in a short amount of time.

4.1.4 MAMP⁸

MAMP is a software stack which includes all the technologies needed for “installing a local server environment” [23]. “AMP” is an acronym which stands for the technologies that are installed with MAMP - (Apache, MySQL, PHP). MAMP was an excellent choice for deployment as it provided everything that was needed to begin creating a website.

As this project needed some way to store large datasets, MySQL was an obvious choice for implementation as it is a scalable, high performance database management system [24]. Also with “81%” [23] of all websites using PHP and an extensive library of built in functions [25], PHP was another excellent choice for building this web application.

⁵Git available at <http://git-scm.com/>

⁶Dropbox available at <https://www.dropbox.com/>

⁷Bootstrap available at <http://getbootstrap.com/>

⁸MAMP available at <https://www.mamp.info/en/>

4.1.5 JQuery⁹

JQuery is a “feature-rich JavaScript library” [26]. It provided me a method to use advanced features, such as AJAX, without writing the underlying source code myself with an “easy-to-use API” [26]. It is also worth noting that Bootstrap’s JavaScript library actually depends on JQuery being declared first in the website’s source code.

4.1.6 D3.js¹⁰

Data Driven Documents or “D3” is a “JavaScript library for creating visualisations” [2]. As this project is reliant on creating functional visualisations, D3 was the perfect choice for developing these; as D3 is “extremely fast” and supports “large datasets” [27].

D3 uses chain syntax, a programming technique used for “chaining methods together with periods” to “perform several actions with a single line of code” [2]. This provides a simple way to “generate and manipulate” [2] SVG (Scalable Vector Graphics) elements. SVG is an image format used on the web and is “defined using markup code similar to HTML” [2]. By using D3 and these paradigms, it is possible to create various charts, graphics and visualisations which are driven by the data they represent.

4.1.6.1 bl.ocks.org

This website, made by *Mike Bostock*, provided free examples of source code for D3 visualisations which I was able to adapt for use with my data. *The initial source code for my implemented pie chart, stacked bar chart and multi-series line chart was copied from bl.ocks.org* [28] [29] [30].

⁹JQuery available at <http://jquery.com/>

¹⁰D3.js available at <http://d3js.org/>

4.2 Critical Source Code Explained

4.2.1 Uploading Log Files

The first crucial section of this software is the upload functionality. With this, log files are uploaded and written to the MySQL database. To do this, I first had to generate a unique ID for each log file. I did this by adding a null entry into the log table and by using the `mysql_insert_id()` method to return the ID that will identify the log file; seen in Listing 1. Next, each log file *entry* is separated into each respective field and appended to a string named *\$query*. This string will then be the *insert* query performed by the database, seen in Listing 2.

```
1 $query = "INSERT INTO log VALUES (null)";
2 $result = mysql_query($query) or die(mysql_error());
3 $logid = mysql_insert_id();
```

Listing 1: Getting a unique ID

```
1 foreach ($input as $entry) {
2     $query = "INSERT INTO description VALUES ('$logid', '$entryid', ";
3     $entry = str_getcsv($entry, ",", '"');
4
5     foreach ($entry as $key) {
6         $query .= "'".mysql_real_escape_string($key)."',";
7     }
8     $query = substr($query, 0, -1); // Remove last comma
9     $query .= " ";
10
11     $result = mysql_query($query) or die(mysql_error());
12     $entryid++;
13 }
```

Listing 2: Writing to the database sequentially.

4.2.2 IFrames

To add a layer of abstraction to the visualisation webpage, each visualisation is embedded as a HTML IFrame. To ensure that these IFrames generate the correct visualisations, the log file ID is parsed to the web page by using a HTTP GET Request inside the IFrame source address, seen in Listing 3.

```
1 <iframe name="ifr" id="ifr" src="graphPie.php?id=<?php echo $id; ?>&type
   =11" scrolling="no" frameborder="0" marginheight="0" frameborder="0"
   class="auto-height" width="500" height="500"></iframe>
```

Listing 3: Using a HTTP GET Request to load an IFrame

4.2.3 Requesting Data

The project implements AJAX (Asynchronous JavaScript and XML). This is a “way to use existing standards” [31] to exchange data with a server and update web pages “without reloading the whole page” [31]. Although it is important to note that this

technology does not restrict its usage with XML (Extensible Markup Language), as this project uses JSON (JavaScript Object Notation) as its primary data type.

Using JQuery's AJAX method, this piece of JavaScript code sends a HTTP POST request to the server, giving the ID number of the log file and the type of graph requested as POST variables. When the data is eventually returned, the callback function is executed which contains the returned data d , seen in Listing 4.

The server will then assign these POST variables to the variables $\$id$ and $\$type$ respectively. Then the server runs a switch statement so it may call the appropriate function and return the correct dataset needed for the visualisation to display correctly, seen in Listing 5.

```
1 $.post("methods/code.php", {id:<?php echo $id;?>, type: <?php echo $type  
;?>}, function(d){
```

Listing 4: AJAX Post Request using JQuery

```
1 $id = $_POST['id'];  
2 $type = $_POST['type'];  
3  
4 switch ($type) {  
5     // Pie Charts  
6     case 11:  
7         kindOfAttacks($id);  
8         break;  
9  
10    case 12:  
11        destinationZone($id);  
12        break;  
13  
14    // Stacked Charts + Line Charts  
15    case 21:  
16        threatsTime($id);  
17        break;  
18  
19    case 22:  
20        severityTime($id);  
21        break;  
22  
23    // Maps  
24    case 31:  
25        worldMap($id);  
26        break;  
27 }
```

Listing 5: Switch Statement

4.2.4 Pie Chart

To draw a pie chart to the web page, the server first returns the data in JSON format. It does this by creating an associative array containing each attribute name (called *label*) and corresponding value (called *count*). This array is then appended to another array named $\$output$, before being returned to the client, seen in Listing 6.


```

1 $query = "SELECT subtype, COUNT(subtype) as X FROM description WHERE
    logid = $id GROUP BY subtype order by X asc";
2 $result = mysql_query($query) or die(mysql_error());
3
4 $output = array();
5 while ($row = mysql_fetch_assoc($result)) {
6     $data = array("label" => $row['subtype'], "count" => intval($row['X']))
7     ;
8     array_push($output, $data);
9 }
echo json_encode($output);

```

Listing 6: Server returning data for a Pie Chart

D3 then takes this data to create the pie chart. This is done by firstly transforming the data by using *d3.layout.pie*. This function takes my JSON data and outputs an array of objects which includes “start angles” and “end angles” - essential for making a pie chart. Then to draw the slices of the pie chart, the code uses a built in d3 function called *d3.svg.arc*, which draws the slice using an SVG “path”.

4.2.5 World Map

To draw the world map visualisation, I first needed to link the full country name (as recorded in the log files) to it’s corresponding 3 letter country code (to be compatible with the DataMaps¹¹ JavaScript library). This was done by using an associative array [32] mapping the 3 letter country code (*key*) to the country name (*value*), seen in Listing 7. Then these key/value pairs are flipped, so that the countries recorded in each log file could be translated to the country code, seen in Listing 8

```

1 <?php
2 $country_codes = array(
3     'AFG'=>'AFGHANISTAN',
4     'ALB'=>'ALBANIA',
5     'DZA'=>'ALGERIA',

```

Listing 7: Country Code mapped to Country Name

```

1 $country_codes = array_flip($country_codes);

```

Listing 8: Flipping the key/value pair

Next, after calling the map function, the database would be queried to return the correct data. This data is then inserted into an array named *\$output*, with the *key* as the country code and the *value* as another array, seen in Listing 9.

This country code is selected by using the previously declared associative array *\$country_codes*. Note that as the country name was stored in upper-case format, the *strtoupper()* method was used so that the log entry could match the array key.

The value of this array is another associative array. This will contain a “fillKey” value (e.g. “veryhigh”, “low” etc) that eventually will determine the colour of the country when

¹¹DataMaps JavaScript library available at: <http://datamaps.github.io/>

drawn on the map. It will also contain a “threats” value, which holds the total amount of threats logged from that particular country.

Finally this data is given to the DataMaps JavaScript library, to generate a World Map. The method shown in Listing 10, draws the map inside the *container* div box and uses the “fillKey” values to select a colour to draw.

```

1 while ($row = mysql_fetch_assoc($result)) {
2   if ($row['X'] > 5000){
3     $output[$country_codes[strtoupper($row['source_location'])]] = array("
4       fillKey" => "veryhigh", "threats" => $row['X']);
5   }
6   else if ($row['X'] > 1000){
7     $output[$country_codes[strtoupper($row['source_location'])]] = array("
8       fillKey" => "high", "threats" => $row['X']);
9   }
10  else if ($row['X'] > 500){
11    $output[$country_codes[strtoupper($row['source_location'])]] = array("
12      fillKey" => "med", "threats" => $row['X']);
13  }
14  else if ($row['X'] > 100){
15    $output[$country_codes[strtoupper($row['source_location'])]] = array("
16      fillKey" => "low", "threats" => $row['X']);
17  }
18  else {
19    $output[$country_codes[strtoupper($row['source_location'])]] = array("
20      fillKey" => "verylow", "threats" => $row['X']);
21  }
22 }

```

Listing 9: Formatting the data ready for D3

```

1 var map = new Datamap({
2   element: document.getElementById("container"),
3   projection: 'mercator',
4   fills: {
5     defaultFill: "#ABDDA4",
6     veryhigh: "#700404",
7     high: "#A30505",
8     med: "#BA1C1C",
9     low: "#C93636",
10    verylow: "#D15252"
11  },

```

Listing 10: Generating the Map

4.3 Problems Encountered

4.3.1 PHP Memory Limit

When attempting to upload files, the website would often crash with the error message - “500 Internal Server Error”. This was due to PHP’s memory limit being exceeded when uploading large files. To rectify this, I modified the PHP settings found in the “PHP.ini” file to increase the “memory_limit” and “post_max_size”, to allow larger log files to be supported with the system.

4.3.2 Pie Chart Labels

When drawing a pie chart, I realised that some labels were obscured because the label was too large for the pie slice, seen in Figure 7. To rectify this I implemented a function [33] to rotate each label and place them on the edge of the chart, seen in Listing 11. Although this still is not foolproof, as some labels, representing small data series, occasionally overlap.

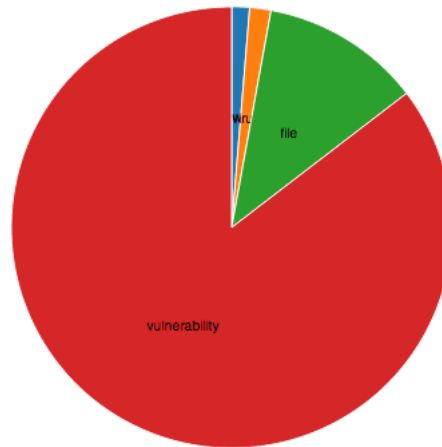


Figure 7: Pie Chart with obscure labels

```
1  var pos = d3.svg.arc().innerRadius(radius + 2).outerRadius(radius + 2);
2
3  var getAngle = function (d) {
4      return (180 / Math.PI * (d.startAngle + d.endAngle) / 2 - 90);
5  };
6
7  g.append("text")
8      .attr("transform", function(d) {
9          return "translate(" + pos.centroid(d) + ") " +
10             "rotate(" + getAngle(d) + ")"; })
11      .attr("dy", 5)
12      .attr("font-size", "13px")
13      .style("text-anchor", "start")
14      .text(function(d) { return d.data.label; });
```

Listing 11: D3 Function to rotate the labels

4.3.3 Performance Issues

One major problem with the website was the slow loading times of a particular graph. To examine this issue, I used Google Chrome's console feature where it was evident that this was caused by long waiting periods from an AJAX request, seen in Figure 8. I then checked my source code where I realised I was querying the MySQL database inside a *for* loop; with another inner loop to process the returned data, seen in Listing 12. This caused my run time complexity to increase to $O(n^2)$. To resolve this, I designed a more sophisticated query that would return the full dataset needed, seen in Listing 13. This query is performed outside of any loop, decreasing the run time complexity to $O(n)$. The loading time is also decreased, seen in Figure 9.

```

1  for ($i=0; $i < 24; $i++) { // Outer Loop
2      $query = "SELECT subtype, count(subtype) as x FROM description WHERE
3      hour(generated_time) = $i AND logid = $id GROUP BY subtype";
4      $result = mysql_query($query) or die(mysql_error()); // Queried db
5
6      $data = array();
7      $data['Time'] = "$i:00";
8      while ($row = mysql_fetch_assoc($result)) { // Inner Loop
9          $data[$row['subtype']] = intval($row['x']);
10     }

```

Listing 12: Inefficient source code

```

1  $query = "SELECT hour(generated_time) as hours, subtype, count(subtype)
2  as X from description where logid = $id group by subtype, hours order by
3  hours asc";
4  $result = mysql_query($query) or die(mysql_error());
5
6  while ($row = mysql_fetch_assoc($result)) { // Single Loop
7      $key = $row['hours'];
8      if (!isset($output[$key])){
9          $output[$key] = array('Time' => "$key:00");
10     }
11     $output[$key][$row['subtype']] = intval($row['X']);

```

Listing 13: Efficient source code

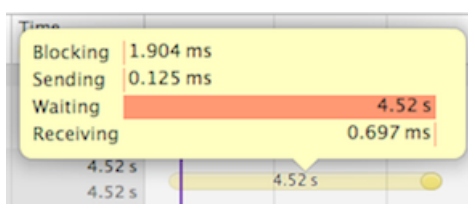


Figure 8: Increased loading times.

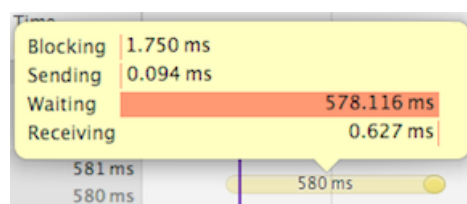


Figure 9: Decreased loading times.

4.3.4 Returning JSON

Even though the returned data was in the correct format, d3 would not accept it as an argument. This was due to a data-type error, where d3 was expecting a supported

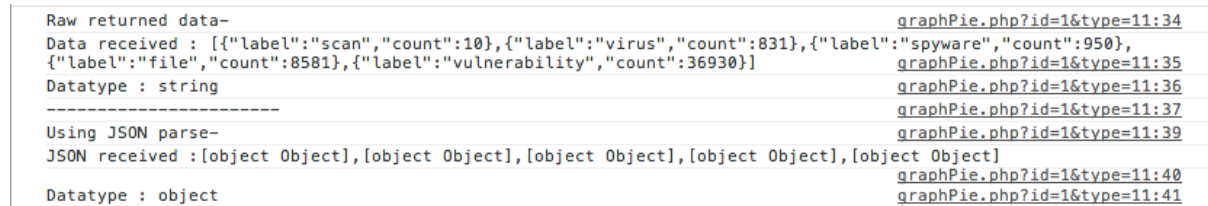
data-type (such as *JSON*) but receiving a *string*. To fix this issue, I had to include the *json_encode()* method in my PHP code, which would return the JSON representation of the value, shown in Listing 14. Then on the client side, this data will need to be parsed as a JSON object, by using the *JSON.parse()* method, shown in Listing 15. Figure 10 illustrates the before and after effects of using these commands.

```
1 echo json_encode($output); // Returning JSON
```

Listing 14: JSON Encode Method (Server Side - PHP)

```
1 data = JSON.parse(d);
```

Listing 15: JSON Parse Method (Client Side - JavaScript)

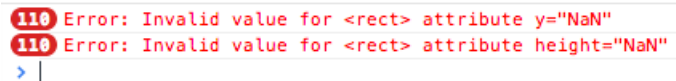


```
Raw returned data-                                     graphPie.php?id=1&type=11:34
Data received : [{"label":"scan","count":10}, {"label":"virus","count":831}, {"label":"spyware","count":950},
{"label":"file","count":8581}, {"label":"vulnerability","count":36930}]      graphPie.php?id=1&type=11:35
Datatype : string                                           graphPie.php?id=1&type=11:36
-----                                                    graphPie.php?id=1&type=11:37
Using JSON parse-                                           graphPie.php?id=1&type=11:39
JSON received : [object Object],[object Object],[object Object],[object Object],[object Object]
Datatype : object                                           graphPie.php?id=1&type=11:40
                                                           graphPie.php?id=1&type=11:41
```

Figure 10: Client's console displaying the returned data

4.3.5 D3 Errors

Another issue was various graphs not appearing due to D3 errors, seen in Figure 11. This was caused by missing attributes in the JSON data whilst being parsed to D3 for graphing. This occurred because the MySQL database does not return empty or zero values after a query has taken place. To fix this issue I created an array called *\$helper* which would contain all the possible values the output could contain. Then I would iterate through the output data to check if each attribute has been included and append any missing attributes before the data is parsed to the client, seen in Listing 16.



```
110 Error: Invalid value for <rect> attribute y="NaN"
110 Error: Invalid value for <rect> attribute height="NaN"
> |
```

Figure 11: D3 errors appearing in the console.

```
1 $helper = array('vulnerability', 'file', 'virus', 'spyware', 'scan', '
    packet');
2 foreach ($helper as $h) {
3     if (!isset($output[$key][$h])) {
4         $output[$key][$h] = 0;
5     }
6 }
```

Listing 16: Iterating through the helper array

4.3.6 CSS Styling Issues

Another problem which was encountered when developing these visualisations was conflicting CSS styling properties. Each visualisation relies on specific CSS properties and as all these visualisations were displayed on the same page, different properties would conflict and cause the visualisations to misbehave. I needed a way to separate these visualisations from each other, but also allow them to be displayed on the same page.

To do this, I modified my design to implemented HTML “iframes”. These are essentially a webpage inside another webpage by embedding the document after specifying the address. These not only rectify the issues that were occurring (because each CSS property will be limited to a specific iframe), but offered more control over the design of the website in general, as visualisations that appear more than once can be amended by editing a single document.

4.3.7 Rickshaw

This project originally used Rickshaw¹², a JavaScript library built on D3 for creating various time series graphs. Rickshaw offered a higher layer of abstraction and simpler creation of graphs compared to D3 by itself. Although these graphs were suitable and early development was going well, after testing with larger datasets, the performance would dramatically be decreased. Another issue with Rickshaw was the lack of supporting documentation it offered. Because of these issues, in week 9 I decided to discard what I had made so far using Rickshaw and switch to a straight D3 approach. I feel that this was a good decision, as the new visualisations work much more efficiently. Even though this decision added to the time lost during this project, many of the concepts needed to build the new visualisations I had already learned during my time developing with Rickshaw.

¹²Rickshaw available at: <http://code.shutterstock.com/rickshaw/>

5 Results and Evaluation

I feel that this project has successfully implemented a software environment, capable of achieving the core requirements stated in the “Specification and Design” section.

When entering the system for the first time, the user is given 3 options. This includes uploading a log file, entering a pre-generated code and visualising real-time data (future scope). After uploading a file, the user is then provided with a unique ID, so that they are able to return to the website at a later time, without re-uploading their log file (see Figure 12).

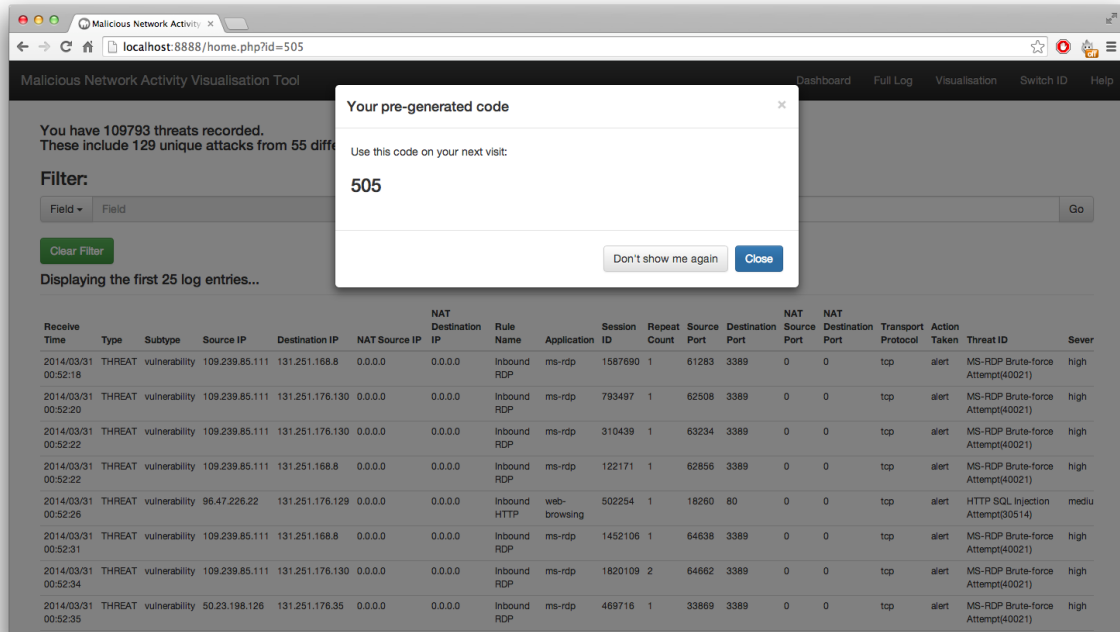


Figure 12: “Splash” screen - home.php

Once the user is “logged in” using either a log file or unique ID, the first 25 log entries are displayed on their screen, along with a generic statement about their log file. Here the user is also able to specify a data filter, to reduce the results shown on the user’s screen to match the filter specified (see Figure 13).

You have 109793 threats recorded.
These include 129 unique attacks from 55 different countries.

Filter:

Field: subtype virus Go

Clear Filter

Displaying the first 25 log entries...

Receive Time	Type	Subtype	Source IP	Destination IP	NAT Source IP	NAT Destination IP	Rule Name	Application	Session ID	Repeat Count	Source Port	Destination Port	NAT Source Port	NAT Destination Port	Transport Protocol	Action Taken	Threat ID
2014/03/31 00:53:29	THREAT	virus	93.184.220.20	10.195.160.13	93.184.220.20	131.251.252.178	Student Web	web-browsing	243220	6	80	55684	80	1407	tcp	deny	Virus/Win32.WGeneric.bxind[221837]
2014/03/31 00:53:35	THREAT	virus	93.184.220.20	10.195.160.13	93.184.220.20	131.251.252.178	Student Web	web-browsing	141028	3	80	55691	80	58600	tcp	deny	Virus/Win32.WGeneric.bxind[221837]
2014/03/31 00:53:41	THREAT	virus	93.184.220.20	10.195.160.13	93.184.220.20	131.251.252.178	Student Web	web-browsing	4786	1	80	55694	80	62365	tcp	deny	Virus/Win32.WGeneric.bxind[221837]
2014/03/31 00:54:36	THREAT	virus	124.95.136.34	10.198.96.127	124.95.136.34	131.251.252.176	Student Web	funshion to Internet	1403662	2	80	49465	80	59451	tcp	deny	Virus/Win32.WGeneric.cdqhw[205968]
2014/03/31 00:56:04	THREAT	virus	93.184.220.20	10.200.96.17	93.184.220.20	131.251.252.74	Student Web	web-browsing	340977	6	80	54319	80	3100	tcp	deny	Virus/Win32.WGeneric.bxind[221837]
2014/03/31 00:56:10	THREAT	virus	93.184.220.20	10.200.96.17	93.184.220.20	131.251.252.74	Student Web	web-browsing	126135	3	80	54325	80	25540	tcp	deny	Virus/Win32.WGeneric.bxind[221837]
2014/03/31 00:56:16	THREAT	virus	93.184.220.20	10.200.96.17	93.184.220.20	131.251.252.74	Student Web	web-browsing	199241	1	80	54331	80	19489	tcp	deny	Virus/Win32.WGeneric.bxind[221837]
2014/03/31 00:58:05	THREAT	virus	111.161.35.75	10.205.67.0	111.161.35.75	131.251.252.19	Student Web	funshion to	1896134	2	80	53786	80	22962	tcp	deny	Virus/Win32.WGeneric.bxind[27368]

Figure 13: Dashboard screen, using the filter feature - home.php

I believe the visualisations generated (shown in the below section) are very clear and easy to interpret. Each visualisation has a title and short caption displayed on the side, describing what information the visualisation is trying to convey to the user.

5.1 Visualisation Examples

What is the most common type of attack launched towards your network?

We can see that a "vulnerability" attack is the most common type of threat launched towards your system with a total of 36930 entries. Followed by a "file" attack with 8581 entries.

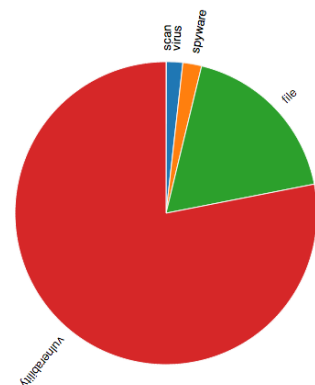


Figure 14: Pie Chart illustrating types of attack (with description).

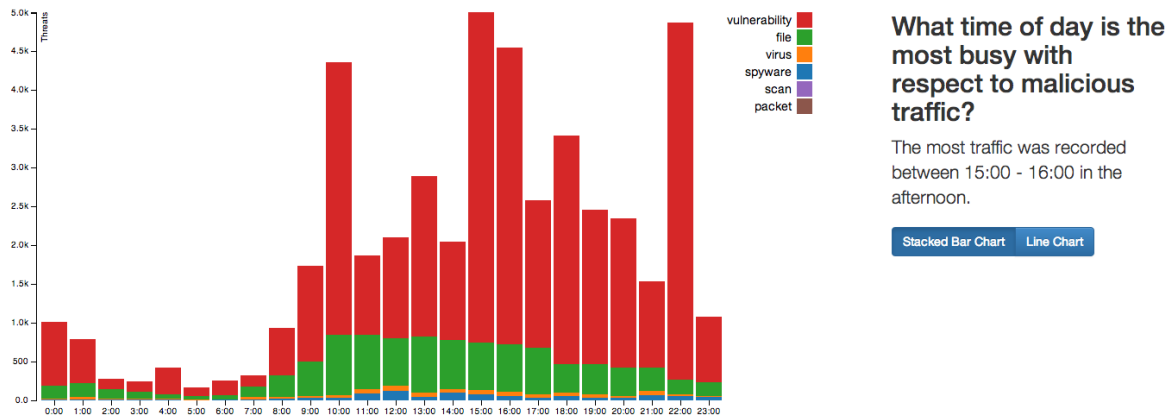


Figure 15: Stacked Bar Chart illustrating types of attack over time (with description).

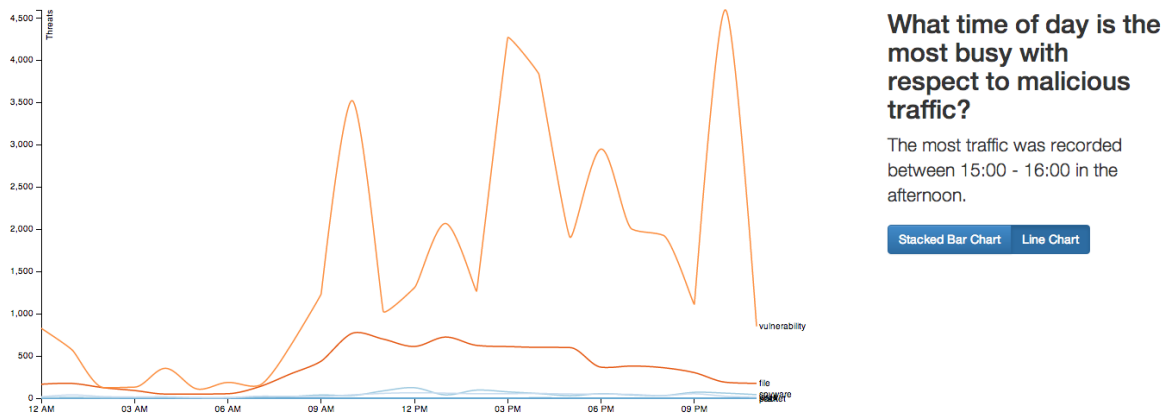


Figure 16: Line Chart illustrating types of attack over time (with description).

Which country do most threats originate from?

The most threats were recorded to come from Singapore, with a total of 11540 log entries, followed by United States with 8291 entries.

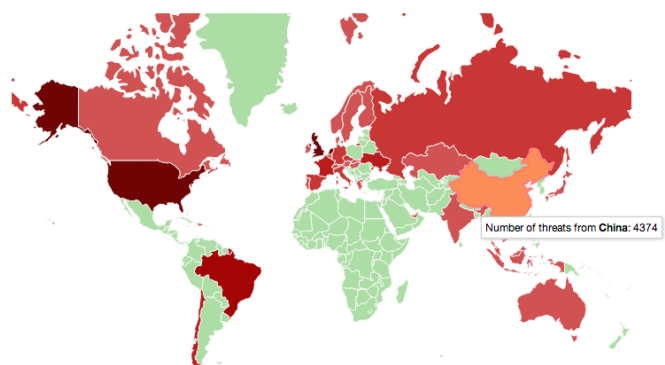


Figure 17: World Map illustrating the countries where malicious attacks originate (with description).

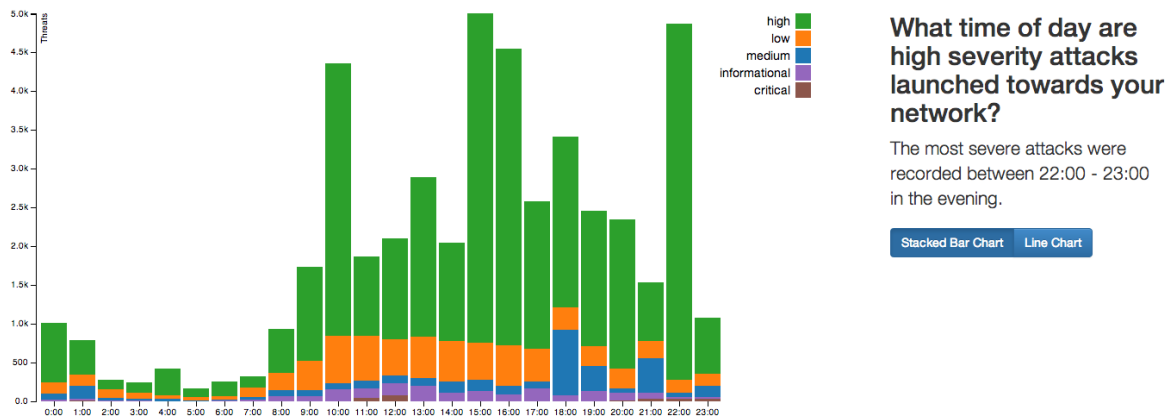


Figure 18: Stacked Bar Chart illustrating different severity attacks over time (with description).

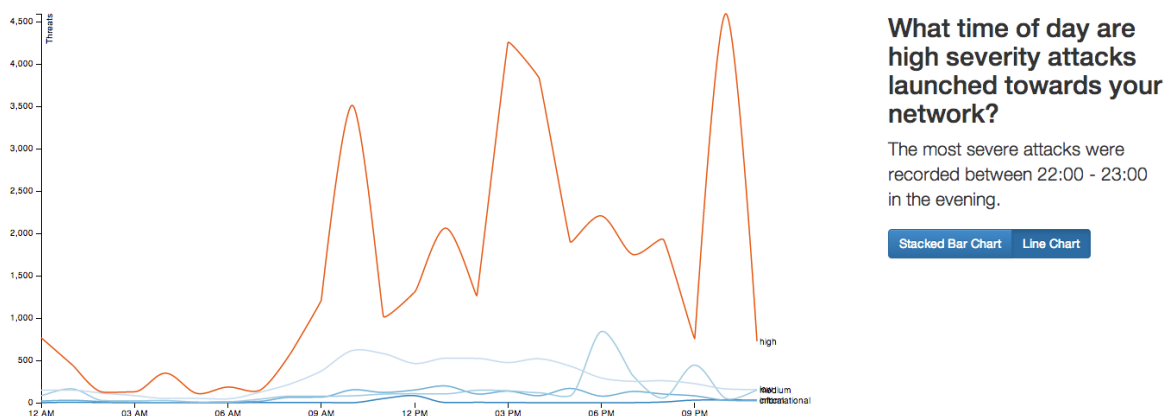


Figure 19: Line Chart illustrating different severity attacks over time (with description).

What is the most targeted zone in your network?

The most threats were recorded to be targeting Campus and Reslan zones.

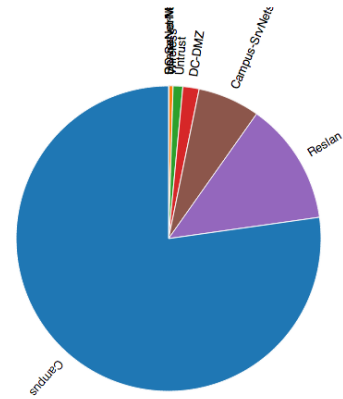


Figure 20: Pie chart illustrating most targeted zones (with description).

5.2 Software Limitations

I feel that these graphs and visualisations work well to communicate information across to the end-user. Although there are obviously a few things that could be improved upon. One feature that could be improved is the avoiding overlapping captions, which label each data series in the pie and line charts. This occurs because some data series have very small values, compared to other series with much larger values.

Another feature I think could improve the system is making the graphs interactive. Static visualisations are indeed a good tool, but by making “animated transitions and well crafted interfaces”, this software could make “exploring” this data much more enjoyable [2].

One issue that I suspect could be occurring is the source location being obscured in some entries. This could be done by using an *anonymity service* such as *Tor*¹³.

Originally developed by the U.S Navy, Tor is a “network of virtual tunnels” [34] which provide a highly sophisticated and anonymous method of communicating over the internet. It was originally designed to protect “government communications” but today it is used by variety of users, including whistle-blowers, activists and journalists [34]. By using Tor, malicious traffic, as recorded in the log file, will exit the Tor network at random nodes (and subsequently IP/ country) each time a message is communicated [34].

This is a difficult issue to detect, although we could argue that a series of contiguous threats originating from the same alleged source IP, is not using this kind of service and is a genuine threat from the suspected country logged. We could also suspect that malicious threats of the same nature (measured by Threat ID and destination IP) recorded at short increasing intervals could be coming from the same source location, although we still would not be able to identify this location as it would be obscured. However a consideration for a future iteration of this project would be to include a alternative view of the world map visualisation to show only source locations which are measured to be accurate, by the above procedures.

Another obvious limitation for this software is that it is only compatible with *Palo Alto Networks* devices that generate a “threat” log file. So another consideration for

¹³Tor available at: <https://www.torproject.org/>

future iterations would be to allow additional log file formats to be supported. This should be a fairly straight-forward task, as when designing the database to store these log files, I added designs for additional tables. Then a new upload method will need to be implemented as well as new methods for querying and formatting the data.

Although there is already a method to add a basic filter to display only certain kinds of threats, this is limited to the “Dashboard” page. In future iterations of this software I would implement a method to filter the data displayed in each visualisation, making the system more responsive for user queries.

5.3 Evaluation of Approach

I feel that this software implements very suitable technologies for the project aims and objectives. This software is very easy to use, with a user-friendly interface implemented with Bootstrap¹⁴. Also as this software is implemented as a website, it is able to run on a variety of platforms.

As a web application, this tool utilises popular web technologies including CSS, PHP, MySQL and JavaScript, making it easily extendible in future. D3 was also chosen as the method for generating the visualisations for this project. I chose D3 as it is an excellent “piece of software that facilitates generation and manipulation of web documents with data” [2]. It is also a well supported library with many tutorials and examples online. Using D3 however meant that generating these visualisations was not a simple process; as D3 does not generate predefined visualisations. It did however provide the most flexibility, “exposing the full capabilities of underlying technologies” [35].

Each visualisation is enclosed inside an IFrame, which provides a layer of abstraction from the underlying code. These IFrames are embedded web pages which act like a kind of factory design pattern. This works well as new kinds of visualisations can be added simply by creating a new web page (to include the visualisation code) and creating a new IFrame linking to the new page.

Because of this design, the simplest way to provide these IFrames with the $\$ID$ and $\$type$ variables (needed to request data from the server) was to use a HTTP GET request. Although another way to do this would be to use session cookies. These allow “users to be recognised inside a website” [36] so that each web page could remember what log ID the user is requesting.

I feel that the might also be susceptible to a brute-force attack from a malicious user. For example, any user can enter the website and attempt to *guess* a unique ID, which is already stored on the system. For this reason, one preventive method would be to deploy the website inside a locally trusted network. Although this could also be prevented by adding some kind of password facility, creating a 2-stage verification system.

5.4 Feedback from INSRV

This software received positive feedback from Damian Southard when he completed the feedback form, available in *Appendix B*. He confirmed that the software was easy to use and that the visualisations were “on par” with current solutions that INSRV had in operation. Also detailed in the feedback form were ideas for future iterations of the

¹⁴Bootstrap available at <http://getbootstrap.com/>

software. This would include real-time data handling and visualisation, custom reporting and exporting of data and custom views. Some of which I describe in more detail below.

6 Future Work

6.1 Real Time Functionality

The first improvement this project could implement is the handling of real-time data, directly communicated from an IDS. This was at first a goal in my initial plan, but unfortunately I did not have time to complete this functionality. *Damian Southhard* mentioned in his feedback form [see *Appendix B*], that this would be a key feature in an operational environment.

Real-time visualisation could be implemented by using RabbitMQ¹⁵, a “messaging broker” [37] which provides a “common platform to send and receive messages” [37]. This application uses the Advanced Messaging Queuing Protocol (AMQP) which enables “client applications to communicate with conforming messaging middleware brokers” [38]. It works by messages being “published” to an *exchange*. The *exchange* then routes the message over a network (e.g. the internet) towards the receiving *queue*. The receiving “consumers” are then able to read the messages by reading their respective *queue* [38], seen in Figure 21.

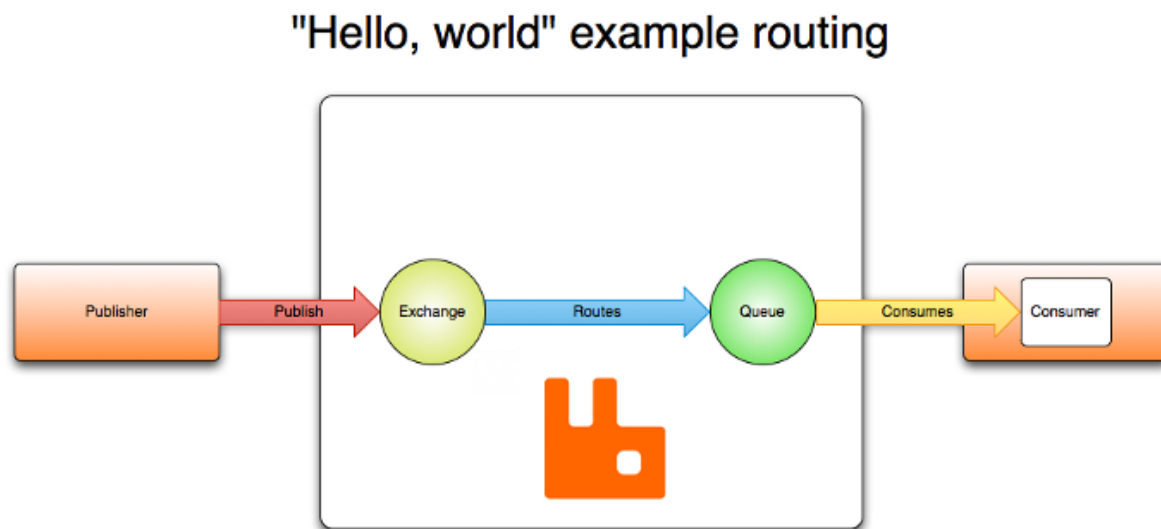


Figure 21: AMQP Model

By using this networking architecture, my software could be extended to continuously read from the RabbitMQ *queue* and return the results to the client immediately for formatting or visualisation purposes.

6.2 CVE Database

Included with the log files from INSRV were 3 XML files. These files contain more specific information about each threat entry, it also links each threat to a CVE (common vulnerabilities and exposures) identifier. This is done by taking the numerical identifier enclosed in parenthesis from the Threat ID field and mapping that to the “entry name” in

¹⁵RabbitMQ available at: <https://www.rabbitmq.com/>

the XML file (see Figure 22). By linking to the CVE database, it will allow a stakeholder to “quickly and accurately access information about the problem” [39] that their IDS just prevented. This could help with further research in this area of intrusion detection.

```

▼<vulnerability>
  ▼<entry name="35931" p="yes">
    ▼<threatname>
      HP Data Protector OmniInet Opcode Buffer Overflow Vulnerability
    </threatname>
    ▼<cve xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <member>CVE-2011-1865</member>
    </cve>
    <category>overflow</category>
    <severity>high</severity>
    ▼<affected-host>
      <server>yes</server>
    </affected-host>
    <default-action>alert</default-action>
  </entry>
  ▼<entry name="35933" p="yes">
    ▼<threatname>
      HP Data Protector OmniInet Opcode 27 Buffer Overflow Vulnerability
    </threatname>
    ▼<cve xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <member>CVE-2011-1865</member>
    </cve>
    <category>overflow</category>
    <severity>high</severity>
    ▼<affected-host>
      <server>yes</server>
    </affected-host>
    <default-action>alert</default-action>
  </entry>
  ▼<entry name="34168" p="yes">
    ▼<threatname>
      HP OpenView Data Protector Application Recovery Manager Buffer Overflow Vulnerability
    </threatname>
    ▼<cve xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <member>CVE-2009-3844</member>
    </cve>
    <category>overflow</category>
  </entry>

```

Figure 22: XML File with CVE IDs

6.3 Additional Data Sources

Linking additional data sources was another suggestion from *Damian Southard*. By linking IDS logs with session data, it should be possible to display and visualise statistical anomalies that the IDS detected. For example, we could display a line graph with 2 series: a normal network session (or baseline) and the malicious network session in question. We would then be able to visualise what triggered the IDS alert by looking at the anomaly - e.g. $>x$ HTTP Requests made in n seconds or an exponential increase in UDP outbound packets.

6.4 Open Source

By making this project open source publicly available, third-parties will be given access to the underlying source code which builds this application. Software with an open source licence will allow anyone to study, change and distribute the software as they please [40]. Websites such as GitHub¹⁶ have become popular as they store “over 12.5 million repositories” [41], where the public can collaborate on various software projects.

¹⁶GitHub available at: <https://github.com/>

Doing this will allow the public to contribute to this project by, for example, adding different kinds of visualisations or supporting additional kinds of datasets. As the stakeholders are professionals who work in IT, this could prove to be a popular option.

6.5 CERE S Project

I am delighted to mention that this project has been invited to support a research project funded by the EPSRC (Engineering and Physical Sciences Research Council). “This Global Uncertainties Consortia for Exploratory Research in Security (CERE S) grant aimed to bring social and computer scientists together to better understand cyber security” [42], which is certainly one of this project’s aims. See *Appendix C* for a short set of slides made to summarise this project and contribute to the CERE S project.

7 Conclusion

This project has addressed the issue of “information overload” with regards to malicious network activity. With the increased need for cyber-security in our modern society, this project aimed to analyse the large growth of data logs and attempt to convey information surrounding these cyber threats through the use of visualisation techniques.

There were many unforeseen problems that arose during the course of this project that made some of the initial goals impossible to achieve. The implemented software is however capable of creating the kinds of visualisations useful for analysing cyber threat activity.

These visualisations also provide the capability to answer some research questions that encompass these large datasets. But this is not yet foolproof, with some alterations to be made; including the support of additional datasets. But on the whole, this tool has made large steps towards a reliable solution for efficient data analysis.

Although this software is not yet ready for an operational environment it has been positively received by *Damian Southard*, a Security Engineer working at INSRV. With further development and by following feedback provided by Damian, this tool could be deployed as a permanent addition for INSRV’s network operations.

Reflections on Learning

This project has taught me many lessons about approaching unfamiliar problems, as well as new technical skills invaluable when studying a subject such as computer science.

From the beginning of this project I decided to use Git as a version control system. I had used this in the past with some issues getting it to work correctly, which made me reluctant on using it again. However I feel that this was one of the best decisions I had made early on, as it gave me the opportunity to safely record my progress and revert any mistakes I may have made. It also gave me the opportunity to teach myself about a popular technology excellent for collaboration. Something that will be an excellent asset in future, when working in an IT role.

One of the main skills I feel that has improved the most is my programming ability and the confidence to approach unfamiliar problems. I have worked on web applications in the past, which has given me a very basic understanding of the underlying technologies that build the web. For this project however I had to expand these skills to build a fully functioning website. I feel that although this went well, I know I will be able to apply a lot of knowledge to the next time I build a project similar to this.

I have always been quite effective with regards to time constraints. Throughout my degree course I have had to manage multiple assignments with overlapping deadlines. However this project has been entirely different as there is only a single deadline; even though there are several deliverables. To solve this problem I created a project plan which organised my time well, although this meant that unforeseen problems would have a dramatic effect on progression. Often it felt like a project plan did not exist at all. To rectify this problem in future, I plan on allowing additional time for each task, even though I may feel that I can achieve certain tasks without this extra allowance.

Throughout this project I often felt confused with regards to the project specification and requirements, which generally slowed my progression. I feel it would have been beneficial to speak to Damian from INSRV earlier about project ideas and to gain clearer goals and designs for this project. Also it would have been useful to hold regular meetings with INSRV, so they would be able to monitor my progress. In future I plan to be more consistent with regards to communication with stakeholders and seek clarification when needed.

Ultimately I feel like this project has been one of the toughest and most rewarding challenges of my university course. I feel like my technical skills have improved dramatically after building this software environment, as well as my ability to learn after reading numerous tutorials, documentation, books and other resources. I now look forward to undertaking further projects during my career, knowing that I have learned much from this experience.

Glossary

AJAX	Asynchronous JavaScript and XML - A technique used to send messages to a server and dynamically update web pages.
API	Application Programming Interface - A set of tools available for building software applications.
Cyber-threat	A malicious action taken place on a computer system, usually over the internet.
D3/ D3.js	Data Driven Documents - A JavaScript library used for the creation of visualisations based on data.
HTML	Hypertext Markup Language - A markup language used for encoding documents used predominantly on the World Wide Web.
HTTP	Hypertext Transfer Protocol - The primary protocol used to browse the World Wide Web.
IDS	Intrusion Detection System - A security measure used to detect and prevent cyber-threats.
INSRV	Cardiff University Information Services - The department which operate Cardiff University's computer networks and services.
JSON	JavaScript Object Notation - A data type which uses human readable text to represent attribute value pairs.
NOC	Network Operations Centre - A location where network monitoring and management takes place.
SSL/ TLS	Secure Sockets Layer/ Transport Layer Security - Cryptographic algorithms used to protect communications over the Internet.
SVG	Scalable Vector Graphics - An image format used on the Web.
XML	Extensible Markup Language - A markup language used for encoding documents to be both human and machine readable.

Appendices

- A - PaloAlto Networks, PAN-OS Tech Note.
- B - Feedback Form completed by Damian Southard.
- C - Slides for CERE S Project.

References

- [1] Asseo L. *Cyber-Attacks Are Increasing Threat, FBI Director Says*. 2013. Available: <http://www.bloomberg.com/news/2013-11-14/cyber-attacks-are-increasing-threat-fbi-director-says.html> (accessed 15 Apr 2014).
- [2] Murray S. *Interactive Data Visualization for the Web*. Sebastopol, CA: O'Reilly Media. 2013.
- [3] McCandless D. *The beauty of data visualization*. 2010. Available: http://www.ted.com/talks/david_mccandless_the_beauty_of_data_visualization (accessed 13 Apr 2014).
- [4] Berners-Lee T. *Information Management: A Proposal*. 1990. Available: <http://www.w3.org/History/1989/proposal.html> (accessed 14 Apr 2014).
- [5] Detica. *The Cost of Cyber Crime*. 2011. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/60943/the-cost-of-cyber-crime-full-report.pdf (accessed 13 Apr 2014).
- [6] Singel R. *War Breaks Out Between Hackers and Scientology - There Can Only Be One*. 2008. Available: <http://www.wired.com/2008/01/anonymous-attac/> (accessed 16 Apr 2014).
- [7] BBC News. *GCHQ chief reports 'disturbing' cyber-attacks on UK*. 2011. Available: <http://www.bbc.co.uk/news/uk-15516959> (accessed 18 Apr 2014).
- [8] Erickson J. *Hacking: The Art of Exploitation. 2nd Edition*. San Francisco, CA: No Starch Press, Inc. 2008.
- [9] Eddy W M. *Defenses Against TCP SYN Flooding Attacks*. n.d. Available: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/syn_flooding_attacks.html (accessed 16 Apr 2014).
- [10] Calder A, Watkins S. *IT Governance. 5th Edition*. London, UK: Kogan Page Ltd. 2012.
- [11] United States Computer Emergency Readiness Team. *Virus Basics*. n.d. Available: <http://www.us-cert.gov/publications/virus-basics> (accessed 21 Apr 2014).
- [12] Federal Trade Commission. *Monitoring Software on Your PC*. 2005. Available: <http://www.ftc.gov/sites/default/files/documents/reports/spyware-workshop-monitoring-software-your-personal-computer-spyware-adware-and-other-software-report/050307spywarerpt.pdf> (accessed 18 Apr 2014).

- [13] United States Computer Emergency Readiness Team. *Spyware*. 2008. Available: http://www.us-cert.gov/sites/default/files/publications/spywarehome_0905.pdf (accessed 21 Apr 2014).
- [14] CODENOMICON. *The Heartbleed Bug*. 2014. Available: <http://heartbleed.com/> (accessed 4 May 2014).
- [15] Rawat M. *Buffer Overflow Basics*. 2014. Available: <http://resources.infosecinstitute.com/buffer-overflow-basics/> (accessed 22 Apr 2014).
- [16] Shinder D. *SolutionBase: Understanding how an intrusion detection system (IDS) works*. 2005. Available: <http://www.techrepublic.com/article/solutionbase-understanding-how-an-intrusion-detection-system-ids-works> (accessed 2 May 2014).
- [17] Mathew D. *Choosing an Intrusion Detection System that Best Suits your Organization*. 2002. Available: <http://www.sans.org/reading-room/whitepapers/detection/choosing-intrusion-detection-system-suits-organization-82> (accessed 2 May 2014).
- [18] Graham J, Howard R, Olson R. *Cyber Security Essentials*. Boca Raton, FL; Auerbach Publications. 2011.
- [19] CESG. *About CESG*. 2014. Available: <https://www.cesg.gov.uk/AboutUs/Pages/aboutusindex.aspx> (accessed 30 Apr 2014).
- [20] Loeliger J, McCullough M. Introduction. In: *Version Control with Git*. Sebastopol, CA: O'Reilly Media. 2012. p1-8.
- [21] Dropbox. *Where does Dropbox store everyone's data?* n.d. Available: <https://www.dropbox.com/help/7/en> (accessed 21 Apr 2014).
- [22] Bootstrap. *Get Bootstrap*. n.d. Available: <http://getbootstrap.com/> (accessed 21 Apr 2014).
- [23] MAMP. *Manage your websites locally*. 2014. Available: <https://www.mamp.info/en/> (accessed 22 Apr 2014).
- [24] MySQL. *Top Reasons to Use MySQL*. 2014. Available: <https://www.mysql.com/why-mysql/topreasons.html> (accessed 22 Apr 2014).
- [25] PHP. *Function Reference*. n.d. Available: <http://www.php.net/manual/en/funcref.php> (accessed 22 Apr 2014).
- [26] JQuery. *Write Less, Do More*. 2014. Available: <http://jquery.com/> (accessed 22 Apr 2014).
- [27] D3. *Data-Driven Documents*. 2013. Available: <http://d3js.org/> (accessed 22 Apr 2014).

- [28] bl.ocks.org. *Pie Chart*. 2012. Available: <http://bl.ocks.org/mbostock/3887235> (accessed 5 May 2014).
- [29] bl.ocks.org . *Stacked Bar Chart*. 2012. Available: <http://bl.ocks.org/mbostock/3886208> (accessed 5 May 2014).
- [30] bl.ocks.org . *Multi-Series Line Chart*. 2012. Available: <http://bl.ocks.org/mbostock/3884955> (accessed 5 May 2014).
- [31] W3Schools. *AJAX Tutorial*. 2014. Available: <http://www.w3schools.com/ajax/default.ASP> (accessed 30 Apr 2014).
- [32] Small Dog Studios. *Country Abbreviation PHP Array*. 2011. Available: <http://blog.smalldo.gs/2011/02/country-abbreviation-php-array/>
- [33] Stackoverflow.com. *Preventing Overlap of Text in D3*. 2013. Available: <http://stackoverflow.com/questions/14534024/preventing-overlap-of-text-in-d3> (accessed 6 May 2014).
- [34] Tor Project. *Tor: Overview*. n.d. Available: <https://www.torproject.org/about/> (accessed 1 May 2014).
- [35] Bostock M. *Data-Driven-Documents*. 2011. Available: <http://strongriley.github.io/d3/> (accessed 5 May 2014).
- [36] AllAboutCookies. *What are session cookies used for?* n.d. Available: <http://www.allaboutcookies.org/cookies/session-cookies-used-for.html> (accessed 4 May 2014).
- [37] RabbitMQ. *Features*. 2014. Available: <https://www.rabbitmq.com/features.html> (accessed 30 Apr 2014).
- [38] RabbitMQ. *AMQP 0-9-1 Model Explained*. 2014. Available: <https://www.rabbitmq.com/tutorials/amqp-concepts.html> (accessed 30 Apr 2014).
- [39] CVE. *Frequently Asked Questions*. 2014. Available: <http://cve.mitre.org/about/faqs.html#a13> (accessed 30 Apr 2014).
- [40] Andrew M St. Laurent. Open Source Licensing, Contract and Copyright Law. In: *Open Source & Free Software Licensing*. Sebastopol, CA: O'Reilly Media. 2004. p. 1-13.
- [41] GitHub. *Features*. 2014. Available: <https://github.com/features> (accessed 1 May 2014).
- [42] Cardiff School of Computer Science & Informatics. *Security experts contribute to £1m research studies*. 2014. Available: <https://www.cs.cf.ac.uk/newsandevents/securityresearchgrants.html> (accessed 1 May 2014).