MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Computer Networks and system security at [redacted]).

It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on the university website provided the source is acknowledged.

[redacted]

[redacted]

MSc. in Computer Networks and Systems Security

[redacted] - Advanced Computer Science Masters Project

22rd July 2022

FIREWALL SIMULATOR

Name: Rahul Dhingra
Student ID: [redacted]
Supervisor: [redacted]

ABSTRACT

Firewall in computing embodies a rather straightforward concept. The basic operability of the firewall revolves around filtering the data packets. Every packet the pass through the firewall is checked along a set of rules which are programmed to it. These rules decide the acceptance or the denial of the packets. Owing to the rate of advancement in technology, many new security threats have been into existence like malwares, DOS attacks, trojan horses etc. These security threats motivate security analysts to constantly review and develop advanced security measures. Therefore, to keep up with the security threats, rulesets for firewall are constantly updated and tailored as per the requirements. These rules require a platform to reflect upon their efficiency. Rule misconfiguration has been an emerging area of opportunity for many firewall systems. These misconfigurations will impact the firewall functionality. Hence, firewall rule set requires testing before being implemented on actual firewalls. Testing the firewall rules require an actual or virtual firewall, which can be expected to be expensive. To address the infrastructural and deployment challenges involved in testing the efficient ruleset for the firewall, a firewall simulator is designed which uses HTML for deployment. It enables students, researchers and professionals to test different variations of the rules rather than going through all the complexities involved in setting up a virtual environment. Therefore, allowing unlimited access to a ready to use virtual firewall platform. Through this project, two terminals are deployed using a web page. Both these terminals are assigned individual IP addresses provisioning the user to change the IP addresses, if required. One terminal is inside a private network and the other one can be used as an outsider. The terminals are designed to operate and accept a few Linux commands like ipconfig, Nmap etc. The project aims to bring together those terminals as an efficient rule testing platform for the researchers, students and professionals. The project concludes with the basic rule testing firewall system and could work as an initial step towards a potentially effective security testing platform.

Table of Contents

CHAPTER ONE

INTRODUCTION

Firewall is a security barrier between a private network and the internet. The main functionality of a firewall constitutes testing of all the incoming packets against a set of rules. Furthermore, the packets are accepted or discarded basis the fulfilment or failure of the criteria. The basic understanding of the firewall, highlights plenty of opportunity areas for researchers and engineers. According to Gouda and Liu (2007), the set of rules defining a firewall often fall into conflicts, owing to the complex structure and requirements. Hence, rigorous testing is conducted on the available rule set to achieve homogeneous categories for risk assessment. The project aims to answer the following research questions:

RQ1: Can a firewall emulator be used to test different rules and further improve the system?
RQ2: How does a firewall respond to the set of rules?
RQ3: Can the rules be tested in a virtual environment to test the operability?

This report concludes a null hypothesis stating that there is an efficient way to test the rule set for a firewall using a simulator. It also confirms that the simulator can be used with other available technologies of efficiently rule designing to build a system which could design and test the rules simultaneously.

The conventional firewall design requires plenty of efforts and investment in designing and implementation phase. This is due to the increase in complexity requirements for the firewall to provide efficient protection. Multiple vulnerabilities can arise due to the drawbacks in design and implementation (Kamara et al., 2003) To test the design and implementation of the firewall, the rules need to be tested along the firewall design policy (created based on the requirements). The testing of the rules requires the deployment of a firewall platform, and the rules could be tested on the deployed system. In 2016, a hardware firewall was designed for testing various rules to improve the security feature of the firewall. The design and implementation involved increased cost for designing and deployment. (Antonov et al., 2016) This implied that testing of the rule chain would be time and resource consuming and can be resolved if the environment is pre-emulated and the user does not need to deploy a firewall every time to test the rules. Therefore, the main aim of this project is to emulate a firewall environment using HTML and JavaScript which enables the user to access the linux firewall terminal through a website. User can upload set of rules and test them on the emulated environment to understand the efficiency and stability of the chains. Therefore, the emulated environment can provide a testing platform for the firewall rules and eventually support in answering the research questions listed.

This report is a structured presentation of the project and explains each step descriptively. The initial section of the report focuses on the introduction of the concepts required for understanding the project's core objective. It describes the basic functionality of the firewall which is also implemented by the emulator designed through the project. Following the introduction, the second section describes the initial idea, background research and the consideration involved in the implementation of the project. Furthermore, the later section of the report focuses on the design and implementation phase which describes each segment of the project in detail and its usage in the completion of the project. The final section discusses the evaluation of the project and the results which eventually answers the research question and justifies the null hypothesis. Henceforth, the report is divided into 4 core chapters:

Chapter One: This chapter focuses on introduction and background around the topic. This background theories are imperative for understanding the basic concepts of firewall and computer security. The chapter elaborates and explains the crucial concepts required for the base of this project.

Chapter Two: The second chapter includes background research and studies around the topic. This is where the literature review is presented. This chapter analyses the existing work in the field of firewall and helps builds a corelation between the existing studies and the presented idea.

Chapter Three: After discussing the background behind the project idea, the following chapter focuses on design and implementation of the model. This chapter highlights the important segments of the code along with its implementation. It also discusses the technology used and a logical reasoning behind choosing them.

Chapter Four: This final chapter is based on the reflection of results and analysis of the project. This section constitutes testing and evaluation. Therefore, it depicts the success of the project and relates it with the future studies and research that can be carried out.

1.1 What is a firewall?

The basic firewall functionality works around a generic concept of packet filtering. The main idea behind a firewall is to put a security measure that will check all the packets arriving at a network against a pre-defined ruleset. The rules test every packet and basis the fulfilment of the criteria accepts or drops a packet (He, 2020). However, the packet filtering firewall can be considered as one of the most basic type of firewalls. Different types of firewalls are also addressed in the later section of the report.

1.2 Firewall VS Routers

Often, routers and firewalls are confused with each other, majorly because the functionality of both the devices requires forwarding the packets to the destination. The job of a router is to analyse the destination of the packet and forward it to the desired device. On the contrary, firewalls will analyse the packets against the rules and act accordingly (Hunt, 2002).

Furthermore, to understand the operability of the firewall, it is imperative to understand the operation of TCP/IP. As, the TCP/IP protocol works on packet segmentation and forwarding, hence, understanding the concept provides the base for understanding the firewall.

1.3 What is TCP/IP Protocol?

"This section of the report is aimed to elaborate the TCP/IP protocol which is important to build the pedestal for a working firewall. TCP/IP is a dual layer protocol responsible for assembling and forwarding the package. The first layer is called Transmission Control Protocol (TCP). TCP is responsible for managing the data packets. Its primary role is to assemble the packets into messages. Furthermore, its counterpart is called internet protocol (IP). The IP layer is the lower layer of the protocol, and it is responsible for managing the address part for each packet. The address part holds the destination address and supports in forwarding the packets to the right destination" (Nath and Uddin, 2015).
The architecture of the TCP/IP model is embodied by the 4 layers.
These layers are:

- Application layer
- Transport layer
- Internet Layer
- Network interface layer / Link Layer

Each layer supports encapsulation. Therefore, each layer has a specified functionality, and all the layers work together to achieve an effective connection.

"The network interface layer or link layer is responsible for the end connection. It controls the sending and accepting the data packets to and from the network. The confined purpose of accepting and dropping the TCP/IP packets supports its independence of network access methods. The objective of the internet layer is to manage the addressing and supports the routing protocols" (Mundra and Taeib, 2015).

"Although all the layers in the architecture works simultaneously, however, the application and transport layer work in the close dependency. As the name suggests, Transport layer is aimed to support the transport of the packets between the two hosts. The purpose of the layer is to transfer the packet to the destination application layer. The application layer on the other hand generates the data to be transferred. The transport layer uses protocols like TCP and UD for the transmission depending upon the requirement of the connection" (Kurose and Ross, 2013).

The TCP/IP architecture is an important concept for the basic firewall operation. The TCP/IP traffic captured from the network is segmented into data packets. These packets are both collected and sent at the application layer. As stated before, firewall is responsible for checking every packet against a set of rules. The process of examining the packets happens at the application layer on both the networks. Therefore, the elaborated understanding of firewall packet filtering requires detailed knowledge of the TCP/IP protocol

1.4 How does packet filtering work?

The simplest structure of an IP data packet constitutes two segments, header part and the data part. The header part is updated at all the different layers of the TCP/IP architecture. The header part has two sub-parts, with the primary one consisting of the IP header for both source and destination. The secondary subpart constitutes the TCP (Transmission Control Protocol) UDP (User Datagram Protocol) part which is designed to explain the underlying protocol. On the contrary, the data part consists of the information or application payload (Northrup, 2009).

The basic packet filtering mechanism majorly involves dealing with the header part of the packet. Packet filtering involves analysing the header of every packet against a set of rules and eventually decide to accept or drop a packet. The basic operation of packet filtering enables the firewall to be fast and economical. As the packet filtering firewall analyses the header part, thus it can accept or block the packets on different criteria like IP address of the source or destination, TCP/UDP ports for both source and destination, etc. The simplicity of the functionality also enables the firewall to allow or deny connection to and from certain ports, Ip addresses, hosts, networks, etc. Therefore, a packet filtering firewall performs the basic security function and is commonly used because of its simple structure and cost effectiveness (Kaur, Panda and Dhaliwal, 2013). The project is aimed to emulate a packet filtering firewall using the HTTP. The emulator provides the user with two Linux based terminals. These terminals can be used to design and test firewall rules. The emulator is also equipped with a tab to upload a rule set which will be used to analyse the packet for acceptance or denial.

The packet filtering firewalls are followed by various other types of firewalls. Owing to the new techniques and approaches used by the hackers and malicious users, a persistent demand is prevailing towards the advancement of security measures, including, firewalls. The next section of the report focuses on different types of firewalls and their functionalities.

1.5 Types of firewalls

The increasing complexity of the networks is one of the major reasons for the existence of different categories of firewalls. Firewalls operate on three basic approaches to protect a network. These approaches are:

- Packet filtering
- Circuit proxy
- Application proxy

Furthermore, many authors clustered these firewalls into two main categories: Transport level and application level (Abie, 2000). As discussed, packet filtering firewall are most used, however, the other categories may turn out to be more secure that the basic one. Like the packet filtering firewall, the circuit proxy firewalls protect the network by checking the header part of the packet. However, the major difference is in the destination address. These firewalls send all the packets to the proxy destination address which after acceptance updates the address to the desired destination address. It enables the system to conceal the destination address but discourages the change of the destination address post acceptance. The application proxy firewalls are more complexly structured than both packet filtering and proxy firewalls. These firewalls collect all the information from the user and base on the available information makes the decision to authenticate a user. These firewalls are expensive and requires high maintenance and updating costs. Therefore, these firewalls are majorly used by the organisations dealing with critical data or information (Abie, 2000).

1.5.1 Packet inspecting firewall

The packet inspecting firewall is the amalgamation of the different types of firewalls. The role of the packet inspecting firewall is to inspect the content of the packet along with the header part. The inspecting system tests the data part of the packet to make complex networks possible (Andress, 2014). The captivating concept of packet inspecting firewall is outside the scope of this project but could be pivotal for future research which can be carried out to emulate the functionality of other types of firewalls.

1.6 Linux Firewall

This section of the report highlights the firewall functionality using Linux operating system. Linux is an open-source operating system which enables developers to use plethora of kernel-based codes and provide flexibility to the system. Linux command line enables user to access the firewall for the system using a set of commands.

1.6.1 IP Tables

"The linux kernel is integrated with a default firewall which is called Netfilter. This enables iptables packet filter to operate and manage the firewall using linux command line. IP tables are configured using ip table chains. These chains house the rules used to determine the operation of the firewall. The rules chain together in order to provide a smooth functionality of the firewall. Therefore, the order of the rules holds significant importance in designing chains. The failure to this adherence of the correct order may lead to errors" (Marmorstein and Kearns, 2005).

1.6.2 Table Chains

As mentioned before, chains are a set of rules organised in an order. Each packet is assessed on the rules in a strict order one after another. This could also trigger the packets to jump through a few rules multiple time (Eychenne, 2019). The IP packets passing through the filter table is processed around a set of rules sent through the chains. The filter table contains three pre-defined chains. These are:

- Input: This chain includes all the incoming traffic which includes the packet with a destination address of the host computer
- Output: The packets originating from the source computer enter this chain

Forward: This chain contains all the packets which are neither originated from nor designated for the host computer. These are the packets that are just using the host machine for routing. All such packets enter the Forward chain (Slider, 2021)

Each rule mentioned in the chains have a very specific function. When the packet traverses the chain, the packet is examined against each rule one after the other, until it matches one or the list exhausts. If the packet meets the rule, the chains then decide of the action on the packet basis the instruction passed from the rule set. The action is usually referred to as target (Eychenne, 2019).

1.6.3 IP table rules

The set of instructions that determines the action taken on the packets are called rules. Former sections explain that the rules are organised in the chains and the packets are analysed against each rule in an organised order. These rules contain specific instructions designated for checking the packets. The specific instructions are designed for both packet and target. Echyenne (2019) suggested that the packet is traversed along the rule if it matches the criteria set for the packet, the system focuses on target's assigned value. The value that can be assigned to the target is majorly categorised into the following:

- User-defined chains
- Extensions
- Special values

As the name suggests, User defined chains consists of user define targets. This means that the packet is handed over to the chain and the action is decided by the chain

Special values: These values are ACCEPT, DROP or RETURN. As the name suggests, the function of these values is to accept the packet through the firewall, discard or drop the packet or send the packet to the previous chain it arrived from (Eychenne, 2019).

1.7 Updating the rules – Addition, deletion and insertion

Every firewall requires a set of rules to check the acceptance of a criteria and decide the desired action or target. These rules can be added, deleted and inserted into the chains by using a few basic linux commands. These commands are important for the base of the project as the project involves setting up the rules on the linux firewall incorporated using HTML.

1.7.1 Rule insertion

"The rules can be inserted into a chain using the command iptables -i. For example, the following command is an example of adding the rules in the chain.

```
# iptables -I INPUT 1 -p tcp -m tcp --dport 80 -j ACCEPT
```

The command mentioned above allows the TCP access on port number 80. This clearly means if the TCP packets arrive with the destination port 80 it will be accepted. The command above has multiple regular expressions that provide the meaning to the command. Which means that the rule is the first in the chain and the rule is for the TCP packets. The target for the rule is to accept the packet if the TCP packet is destined for port 80" (3.12.1.2 Inserting Rules in a Chain, 2013).

"The simplest working of the firewall explains that the packet received is traversed along the rule set, as in this case there is only one rule added to the chain. Therefore, the packet is traversed on the only rule in the chain and if the packet is a TCP packet with the destination as port 80, it is accepted all the other packets are dropped.
To confirm that the rule has been added to the chain a simple command iptables -l is used.

```
# iptables -L --line-numbers
```

This command displays the rules in the chain with the line and the number of the rule. The output of this command in the given scenario is as follows.

```
Chain INPUT (policy ACCEPT)
num   target      prot  opt   source        destination
1     ACCEPT      tcp    --   anywhere      anywhere       tcp dpt:http
```

The above output confirms the insertion of the rule in the chain" (3.12.1.2 Inserting Rules in a Chain, 2019)

1.7.2 Adding and deleting the rules

"Like insert, the add and delete commands are used by adding different regular expression to the command. The expressions used are -a and -d for adding and deleting the rules to and from the chain respectively. For example:

```
# iptables -A INPUT -s 192.168.1.0 -j DROP
```

This command adds the rule to the chain, which analyses all the packets and the packets coming from the network 192.168.1.0 are dropped

On the similar grounds, delete is used to remove the rule from the chain. For this command it is imperative for shell to be aware of the number of the rule to be removed. Therefore, the command used to delete a rule is:

```
# iptables  -D  INPUT 3
```

This command is used to delete the third rule from the chain" (Eychenne, 2019).

All the concepts explained in this chapter of the report are crucial for understanding the design and implementation of the emulator. The terminals displayed in the project are the linux based terminals and are programmed to respond to the basic linux commands. This enables the emulator to accept the commands and the rules from the user to train the firewall emulator system. The project requires the basic understanding of packet filtering, ip tables, ip rules, etc. Hence, this chapter plays a substantial role as the basic building block of the project.

Chapter Two

Background and research

2.1 Initial idea

Firewall is a security service protecting a network from outside world/ internet. As discussed, the rise in malicious attacks has led to increase in demand for improving and updating the network security feature including firewall. According to cisco, leading developers of the network and networking devices, firewall is classed as the first line of defence in the network for over 25 years (What Is a Firewall?, 2022). Considering the first line of defence, the firewall rules and design should be secure and should require constant updating to maximize the security potential. Plenty of analysts and researchers have contributed to the field of firewall design to maximize the security potency. Some of these studies have been mentioned in the previous chapter. These studies worked as the base of this project and are discussed in detail in the other sections of the report. The persistent requirement for better firewall policies and structure has led to the demand of a testing environment to test the designed chains virtually before the real-life implementation. Testing in virtual environment is of imminent importance in computer science due to the following major factors.

1. It aids in improving the QoS (Quality of Service) of the end software, application and functionality due to rigorous testing and modification (Netto et al., 2011).
2. It provides a cost-effective platform for testing; avoiding the cost of infrastructure, energy and resources (Corrêa Souza, Nunes and Delamaro, 2018).

Although, there are variety of reasons students and researchers may use the virtual environment for. However, the reasons mentioned above were the motivator behind creating a software firewall to test the rules and chains for developing better firewall policies and structures.

Inspired by the existing studies, the idea for this project is to use JavaScript, HTML, and CSS to create two linux terminals with some basic functionalities. The reason for the restricted functionality was the time frame and complexity of the project. These terminals provide access to the iptables functionality in linux. This can be used to test the rules on the firewall by uploading the rules to the firewall by using a button. The firewall functionality is tested on the rules and the test results are presented in the later chapter of the report.

2.2 Literature review

As supported before, firewall is the most widely used security feature in many organisations; both commercial and educational. Therefore, pivotal studies and research may help the growth and advancement of the topic. Plenty of researchers have undertaken studies to understand different aspects of firewalls. This section of the report focuses on the literature review conducted on the research work carried out within the scope of this project.

Firewalls are managed by firewall policies and configurations. The firewall policies help decide the fate of data packets. An error in the policy can make the firewall deviate from its basic operation and could cause either the acceptance of illegitimate packets causing the holes in security or discarding legitimate packets causing disruption of business as usual. This motivated the authors, to design a concept of diverse firewall design which categorises the firewall errors into specification induced and design induced errors. These are used to improve the design of the firewall (Liu and Gouda, 2008).

In 2004, another study was conducted to identify the problems in firewall rule misconfigurations. These misconfigurations cause the firewall to misbehave and hence requires a system to manage the filtering rules. The study focused on creating a Firewall Policy Advisor. This tool managed the generic firewall policy and filtering rules enabling the minimisation of the rule misconfiguration (Al-Shaer and Hamed, 2004).

Another tool developed by the researchers was called FIREMAN. Trying to resolve the same problem of firewall rule misconfigurations, authors introduced analytical toolkit to test the system for inconsistencies, policy violations, inefficiencies, etc. i.e., the misconfigurations in the firewall rules. The model tested all the possible IP packets and all possible paths to derive an efficient model with minimal misconfigurations. This tool is another indication of the requirement of a system to improve the firewall rules and policies (Yuan et al., 2006).

These existing theories led to the motivation behind generating optimal firewall models. Different researchers have attempted to create efficient firewall models for the purpose of minimizing the risk and cost involved in designing the firewall. One such idea was proposed in 2015, when the authors proposed an optimized model of firewall based on Service-Grouping. In this model, an algorithm tests all the rules on rule service list. The algorithm lists out all the conflict segments and eventually resolve those conflict segments by using the action constraint strategy. The rules are listed separately and then merged by using a merging algorithm to create a rule set of most efficient rules. This methodology evidences another technique for creating an optimized system for firewalls (Zhang and Huang, 2015).

The available research and studies indicate the constant need of improving the firewall rules and policies. As mentioned earlier, this is the major motivation behind developing a support system for improving the security feature of firewall. The previous work in this area explains the need for constantly changing the firewall policy design and filtering rules. Changing these rules and policies for their better versions require a platform to continuously test the rules and analyse the outcome. Many protocols, algorithms and policies have been designed to improve the system for choosing the firewall rules and policies. However, there is still an increasing need of a platform to test the rules and policies to analyse if the rules serve the desired purpose efficiently. This worked as a motivation towards the idea of this project. As proposed earlier, this project provides a platform for those rules to be tested and examined if the desired outcome is achieved.

2.3 Considerations

The growth of technology may have exposed the computer science industry to a variety of considerations towards the society. These considerations are majorly legal, ethical, social and professional. This section of the report aims to address majority of those considerations in reference to this project.

2.3.1 Ethical Considerations

Lately, ethical considerations have been an integral part of computer science. Ethics in technology can be simply defined as the set of moral rules or values governing the system. These moral values could require a balance between right to information, privacy, dependency, etc. Owing to the vastness of the topic many researchers have attempted to conceptualise a framework to better define the ethical issues. For example, Norberto Patrignani (2018) has attempted to create a framework to define and address impact of computers on the different section of the society. This highlights the importance of ethical considerations for the project.

Similarly, some authors have tried to specifically address the ethical considerations involved in the human involvement in the research in CS (Computer Science). Human participation in technology research involves considering various aspects of Human rights to avoid any infringement (Buchanan et al., 2011). Therefore, any project or research involving human participation may require a various level of ethical approval. In the presented project, no human participation was required. Therefore, it eliminated the need of getting an ethical approval from the University.

Post addressing the ethical considerations towards the University, the project poses some privacy and intellectual property rights concerns. The firewall emulator can be used by illegitimate users to test the effectiveness of the attack. This can enable the user to find the drawbacks in the attack tools and make necessary changes to improve the attack.

Similarly, it poses risk to intellectual property rights as well. As the code, if made available, could attract people to copy or tweak the code for malicious use, leading to the infringement of the rights. Falling prey to illegitimate user may lead to the breach of computer misuse act 1990 which states that the unauthorised access to the system or software with an intent to cause harm is an infringement of the computer misuse act (CMA) (Gov.uk, 2022).

2.3.2 Social Impact

There is a thin line between the ethical and social considerations of the project. The social aspect diverts more towards the impact on the society rather than considering the risk posed by an individual. Many researchers have evidenced the heightened impact of technology on the society. This is due to the increase in the user base for technology. The authors also expect the technology to lead the professional sector as well (Au-Yong-Oliveira, Gonçalves, Martins and Branco, 2018).

Owing to the vast societal coverage for technology, the impact on society has been a mix of both positive and negative. Similarly, the project poses both positive and negative impact on the society. For the positive impacts, it enables different users to test the strength and accuracy of their firewall rules/chains. It also provides an infrastructure free environment for the researchers, simplifying the firewall deployment for the user. Some policy errors in the firewall configuration pose a gigantic threat on the society. These policy errors can have a negative impact on the society as it can be used to block certain traffic against the user's will. Furthermore, a firewall policy error may lead to the breach in the security by creating holes that can be exploited by the malicious content available on the internet. It also provides opportunity to the hackers to exploit the vulnerabilities (Liu, 2007). Therefore, a cautious approach should be adapted while designing the firewall policies to minimise the social impact of the firewall. Similarly, in the project, user is required to create the rules after critical evaluation of the rule and its impact on the firewall policies to mitigate the negative aspect of the social impact.

2.3.3 Legal Implication

The legal threats posed by technology is evident through all the malicious content, attacks and tools being easily available to the user. Many authors confirmed the increasing threat technology may inflict on the security. For example, the damage that could be induced by a malicious user on the defence systems, electrical gird, theft etc. These give rise to the need for designing a legal framework for the efficient use of technology (Hathaway et al., 2012). The proposed project can fall into some of the legal categories as well. Primarily, as the emulator is presented through HTML i.e., web application. Therefore, it poses a direct threat to the identity and sensitive user information. Although, in this project, the webpage is not hosted and makes use of the hard drive to make the emulator available locally. However, it can be hosted online to provide remote access to the users as well. This could lead to potential threat of collecting user information which is one of the factors listed in the Data Protection Act (DPA). Although, plenty of laws have been created in the field of cybersecurity. However, still a large variety of challenges have not been addressed, leading to the constant updating to the policies. For the scope of this project, some of the legal issues are also covered by the computer fraud and Abuse Act (CFFAA 1984). The act states that the misuse of available technology for fraud and damage is a legal offense (Veale and Brown, 2020). Therefore, if the tool is used for the purpose of design unethical rules or tools to cause damage to the system, it may be considered illegal and have some legal repercussions. To mitigate legal impact, the software distribution can be licenced and managed which would keep track of the users and minimise illegitimate use.

CHAPTER THREE

DESIGN & IMPLEMENTATION

3.1 Design

The main purpose of the project is to provide a platform aimed to address the research questions mentioned in the beginning of the report. To enable the firewall functionality on the terminal it was imperative to enable the iptables functionality. The main goal is to provide the iptables functionality to the simulator hosted on a Web page. This goal statement draws many other questions and the need to answer those questions. The initial questions about the design and implementations were as follows:

- How should the simulator look?
- What functions should it perform?
- How can the results be analysed?
- What all problems does it address?

This set of questions define the base for the design and implementation of the project. This chapter of the report aims to address all the questions related to the design and implementations. It lists out all the challenges faced during the life cycle of the project. Furthermore, this section also highlights the software, tools and languages used to achieve the desired outcome.

3.2 Requirements

The project required a combination of HTML, CSS and JavaScript to achieve all the functionalities intended to achieve through the application. However, due to lack of time and required knowledge, the terminals couldn't be incorporated with all the functionalities of a Linux terminal. This can be a future research opportunity to enhance the terminal operability. The list of requirements created during the planning phase of the project are listed below.

1. Design the terminal with the linux functionalities majorly to process the commands like ipconfig, nmap and iptables
2. Iptables functionality allowing the terminal
3. Provide two buttons to upload and download the rules to and from the simulator
4. Test the functionality to validate the working

These objectives were further classed into sub objectives to work as the checkpoints for the progress of the report. Plenty of challenges were encountered while designing the model. These challenges were specific to the code and functionality. The tools and languages used were compared with the other available technologies and the decision was made based on the tools best fit for the project. The further section of this chapter talks about the technology used and its comparison with the other available tools and the reason for choosing the technology and background about design implementation.

Software/ Technology/ Language used:

1. Microsoft Visual Studio Code
2. Node.js
3. HTML (see Appendix I)
4. CSS (see Appendix I)

5. JavaScript

3.2.1 Visual Studio Code
Visual studio code is a free open-source cross platform tool used to write and compile the code written in different languages. Visual studio code has been built on the electron platform which strives to provide consistent experience across different platforms (Introducing Visual Studio Code, 2019). There are plenty of other text editors available in the market which could perform similar actions to visual studio like Notepad++, Sublime Text, UltraEdit, Atom etc. The main reason behind using Visual studio code was a mix of experience with the tool and the ability to process variety of code on the same application on different platforms.

The reason behind using the other languages as compared to their alternate is explained in the further section of this chapter along with design and implementation of the individual component.

3.2.2 Node.js
Node.js is an open-source platform providing run-time environment which was originally built to provide fast and scalable applications. It was developed to aid both client-side and server-side scripting using JavaScript as the core language. One of the main advantages of using Node.js is that it is event-driven and uses non- blocking input output model which makes it efficient and low memory consumption platform (Chhetri, 2016)

As mentioned above, Node.js can used for both front-end and back-end development. However, for this project it has been used to develop the front-end application as back-end development was not required for this project, owing to the functionality and operability.
There were plenty of reasons for using Node.js as it supported the course of the project. However, one of the main reasons for using it was the availability of NPM (see Appendix I). NPM was used to install additional packages which were required to ease out the development process. The list of some important packages used for the project is mentioned below.

- browserify
- Watchify
- File-saver
- Jquery
- Gulp
- Underscore
- Vinyl-buffer

This is a non-exhaustive list which implies many other packages were used as well. However, those packages supported the background functionality. Furthermore, these were a few packages that worked as the base of the project as these supported some major functionalities of the project. These packages reduced the time taken to implement many steps which were required to be repeated over the life cycle of the project. One of the main examples of that is the use of Watchify.js. This package was used to build the code every time before running. It is imperative to build the code every time to implement the changes made in the code. The development life cycle of the code requires constant changes and updating to remove the errors or improve the functionality. All these changes required for the code to be built again before running. This is done automatically by watchify.js and eventually supported in saving plenty of time that could have been wasted.

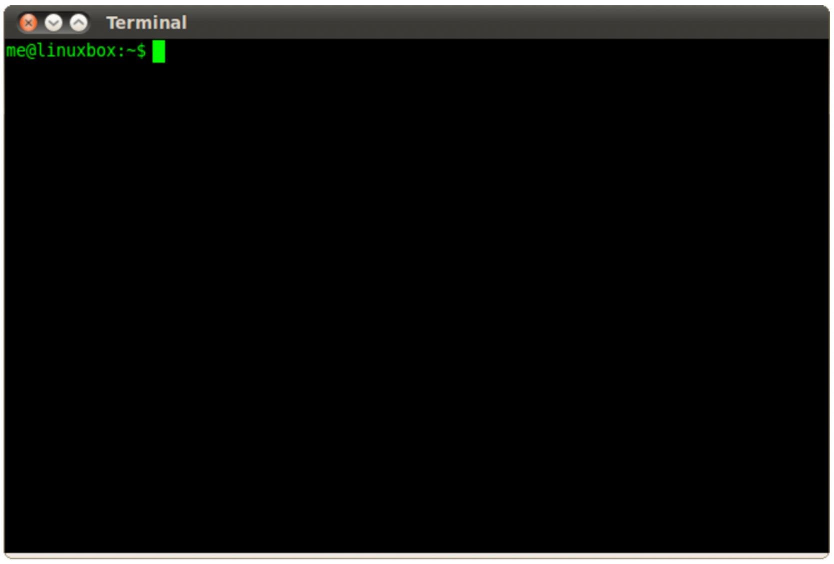
Furthermore, another package which provided immense support in the project was browserify.js. This package was used to provide a structure to the code. This structure was provided by organising different code-segments into different files. This enabled access those snippets individually and make the changes and error rectification easy.

Another important package that simplified the development of the code is called Gulp.js. The main purpose of the package is to make JavaScript programming simple and efficient. The package supported the programming by making the code smaller and efficient. This was done by removing the debugger statements automatically. These statements were either removed or converted into a comment.

3.3 The simulator's outlook - GUI

The design of the simulator demands at least 2 terminals. These terminals are named as the Internal and External. As the name suggest, the terminal named internal refers to a device in a network and external is used for the device outside the network. The main purpose of the external terminal is to try to send the packets to the internal terminal. The window should be equipped with two buttons to upload and download the rule list or chains to the firewall terminal. Two more buttons are placed to insert or delete the rules from the chains. Furthermore, the terminal should imitate the appearance of the linux based terminal and should respond to the linux commands as well. The terminals are created using a mix of HTML and CSS and the appearance of the created terminal matches the appearance of the linux based terminal. The two images mentioned below show the comparison in the appearance between the original linux terminal and the one created using CSS.

Linux terminal: The original linux terminal



```
me@linuxbox:~$
```

Figure 3 (i)

Initial terminal design created for the project

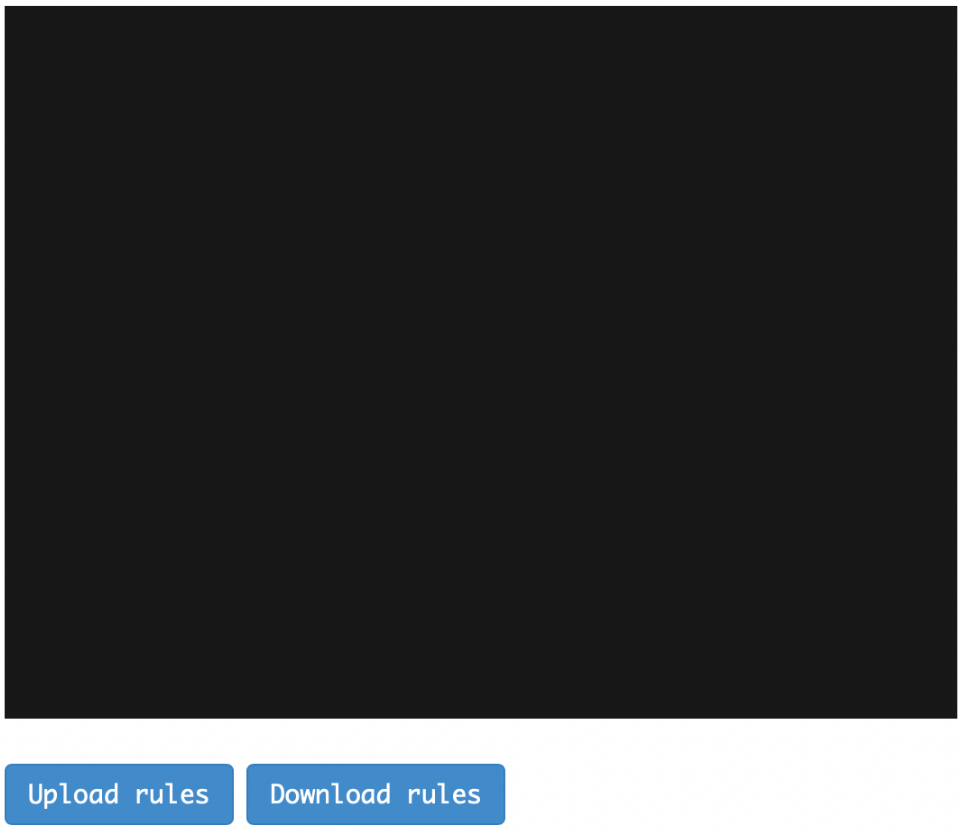
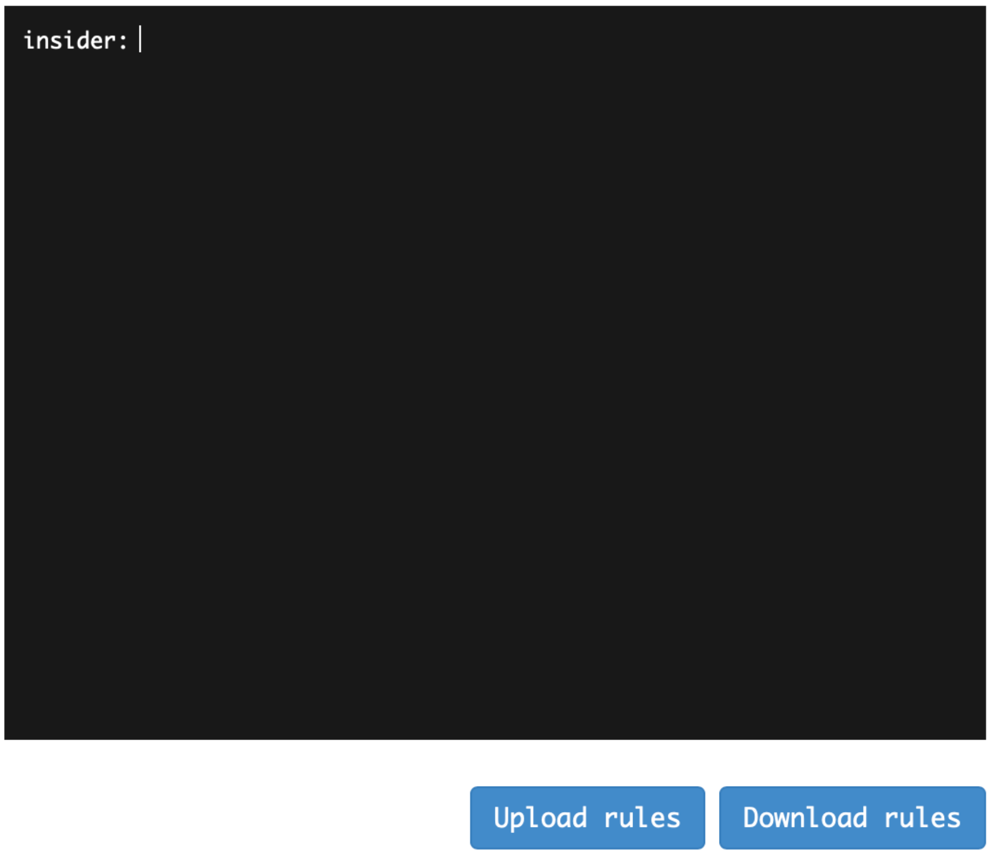


Figure 3(ii)

The GUI for the model has been kept simple. As the main purpose of the code was to answer the research question by developing a platform to test the firewall functionality and rules. Therefore, the focus of the code was to be able to implement the basic packet dropping firewall. One of the challenges encountered while coding the GUI was to make it like a linux terminal. Every linux based terminal display two main aspects. Primarily, it is the user logged into the terminal. This is done by displaying the name of the user before every command. The second important aspect was a cursor, especially a blinking cursor to mark the position of the text in the terminal. Imitating this functionality was a big challenge that took plenty of refining and changes to the code. The image previously displayed is the embodiment of the initial terminal without the username and the cursor. The output received post successfully updating the code is depicted by Figure 3(iii).

Terminal's outlook



```
insider: |
```

Figure 3(iii)

CSS was used to design the appearance of the terminal. The use was limited to define the border and the size of both the terminals. As mentioned before, the primary aim of the project was to implement the functionality, therefore the terminal design lacked some features. The final CSS code was the result of plenty of changes incorporated throughout the duration of the project.

As CSS was used to design the appearance of the terminals. HTML was used to design rest of the outlook or body of the landing page. It was used to put the text to the page, sharing some important information. It was also used to create the upload and download button for the application. The HTML code was a bit straightforward code to provide GUI and basic operational structure to the code.

As the firewall functionality was implemented using a webpage. The website or page must be hosted for the user to be able to access it. The hosting service is usually paid. Moreover, the University does not provide any networking environments like Network, router, etc. The combination of both former and the later, led to the use of the loopback address 127.0.0.1. This loopback address provides the functionality of working on the local disk rather than hosting in on a platform.

3.4 Implementation

3.4.1 Linux functionality Challenges

The terminal portrayed some linux functionality. It responded to a few linux commands like ipconfig, nmap and iptables. The limited availability of time and resources was the barricade which restricted the user from achieving a full scale linux terminal functionality. To minimize the impact of this challenge, a couple of mitigation strategies were used.

I. The main HTML body, explicitly lists out the commands that can be used in the terminal
II. If the user tries to enter a different command as mentioned on the landing page, the screen displays a command not known error. This functionality has been adapted from shell scripting which lists out the unrecognised command

The images displayed below evidence the mitigation strategies and supports the project on the desired path

Commands mentioned on the landing page (See Appednix VII)

Terminals' response to legitimate linux commands (See Appednix VII)

The major challenge was to consider what functionalities to include. The logic for the project deliverable required only three major commands to be read by the terminal. Therefore, to focus on the basic operability, all the commands other than the ones required were not included in the iteration.

3.4.2 Classes / Scripts

As the project revolves around the firewall in linux, therefore, implementing the functionality of the basic shell scripting was a challenge. Different classes are used in the code which segregated the functionality of each of the commands. The commands which are recognised by the terminal are listed below:

- Iptables
- Nmap
- Set ip
- Ipconfig
- clear

All these commands are network-oriented commands i.e., these commands are used to analyse or manage a particular network. The objective of the simulator is to provide a platform that supports firewall functionality which means protecting a private network from the internet. Therefore, these commands worked as the base of the model.

3.4.3 iptables

It is used for creating, managing and inspecting the tables with IP data packets. The purpose of the command is to manage the rules in the chain. It must demonstrate diverse functionality like insert, append etc. To demonstrate the concepts of object-oriented programming visual studio and node.js organizes all functionalities of the tool into different builds, classes or modules. Some languages also call it as functions. After the successful design and implementation of the GUI, the following steps included designing and programming the iptables which was pivotal for the tool to behave like a firewall. The iptables functionality is implemented using classes in JavaScript. The class or the script is named iptables.js. The primary function of the script is to insert or append new rules to the chain. The script was created to check a rule exists in the history already and basis that it calls different functions to perform desired actions. The functions were placed in the code in the order of required functionality. The functions within the iptables class are included in the code in the following sequence:

- _loadhistory: The purpose of the function is to check if there is any rule in the history i.e., the chains. If there is, then the program needs to append the new rule to the list otherwise adds the rule to the list
- addRule: The aim of this simple function is to check if the chain has any rules and push the newly added rule accordingly
- insertRule: This function is like the addRule function, or it can also be referred to as the extension of the previous function. Once the addRule analyses the information in the table, insertRule inserts the added rule to the list
- deleteRules: As the name suggests, the function is used to delete the rule from the table. It just requires the chain and the position as the parameters. The selected rule is deleted by the function and the remaining rules are concatenated together
- listRules: It lists all the rules in the chain
- resetRules: The purpose of this function is to reset the rules in the chain so that the user can start afresh

In addition to the imperative set of rules, some additional functions are included in the same class. These functions are getUniquechains, isPortopened, isAddressWithinRange, filterByChain and _tableConatinsRules. These functions are named according to the actions they perform.

All the functions above are clubbed into the iptables class which works as an important module for the project

3.4.4 Output message

Post setting up the basic iptables functionality, the next step involved displaying the message on the screen. To incorporate this, a class called message.js is designed. The purpose of this class is to provide the time stamp and date with output on the terminal window. It fetches the date and time by using some inbuilt functions like getHours(), getMinutes() and getSeconds()

3.4.5 Command Processor

The CommandProcessor (see Appendix II)is a crucial class for the project. It is used to design and execute all the processes required to inherit the firewall functionality on the online application. The project demanded user to choose from the list of available options to make changes to the rule in iptables. In linux, this is achieved by using regular expressions. To provide this operability to the code, a class named CommandProcessor.js is constructed. This class enables the user to accept the regular expressions like F, A, I, D, S, P, J, dport etc. These expressions provide user the flexibility to select from the list of options as mentioned earlier. The functionality is assigned to each of this regex in the code. This class describes how should the rules be processed. The main objective of user control is achieved using this class. This part of the class is based on a simple if-else iterative loop. These expressions are used primarily with the iptables command to explain the intended action to the compiler. The code snippet from the class which uses if-else iterative loop to determine the course of action and simultaneously calls the function designed to perform the task. For example, if user enters iptables -i INPUT 1 –dport 80 -j ACCEPT the compiler calls the function insertRule with three parameters, two of which are INPUT and 1. Furthermore, the third parameter is the value returned by newRule.build() fuction.

To implement another group of functions, the class uses a switch statement which is used to take actions on the messages using commands on the terminal. The statement takes msg as the parameter and compares it with all the cases in the statement. In this case, when the command matches any of the case, a relative function is called to perform the desired action. The commands addressed by the switch are del msgs , clear, save msgs, and ipconfig. These commands perform basic operations like removing the messages from the screen, saving the messages and displaying the ip address of the terminal. Any other commands entered by the user displays command not found error as explained in the previous sections.

The final set of operations performed by the class involves nmap and managing the ports. Nmap is an open source linux based tool which is used for scanning the vulnerabilities in a network (Shah et al., 2019). For this project, nmap has been used for an active port scanning functionality. It is prevalent for testing if the firewall is performing the desired action. Another function named process was used to check if the port is open-ended or closed. Furthermore, basis the firewall rules the status of the port is changed.

A major challenge encountered during the project was to be able to save the files containing the rules. Furthermore, the demand to make the tool compatible on different devices. Apart from defining and constructing the class and functions, the code also includes some additional classes or scripts which are taken from the online resources. These functions provide the supporting operability to the tool. These functions include:

- FileSaver.js – to save the or commands (see Appendix III)
- Bootstrap – to support cross platform operations (see Appendix IV)

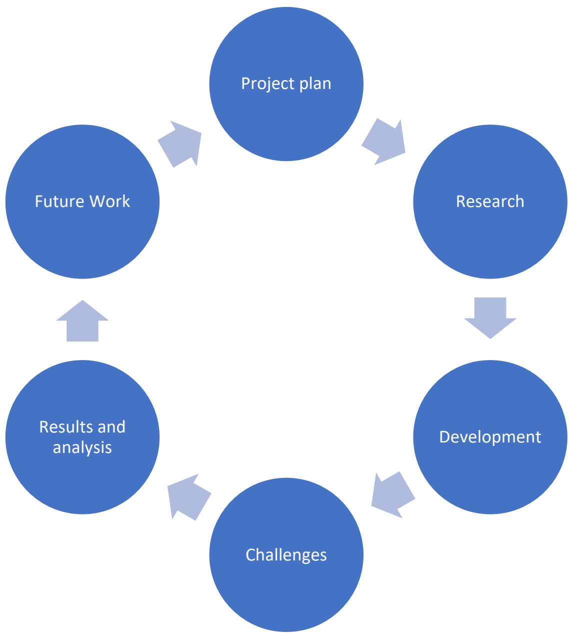
CHAPTER 4

REFLECTION & CONCLUSION

The underlying chapter highlights the evaluation of the created tool and concludes the project. Reflection is an integral part of project management. Schon suggested that practicing reflection in project management during academia supports students to transfer the skill in the workplace. (Schön, 1983) Many theories and research highlighted that the reflection is often excluded from the project, owing to the lack of time (Anbari, Carayannis and Voetsch, 2008). However, authors have conducted surveys and studies to stress on the importance of reflection in the real-world environment. Reflection on the studies and projects prepare the students to tackle the real time challenges that they encounter at workplace (Flynn and Levie, 2021).

Self-reflection on the project can be achieved by encapsulating the project in a few basic questions. The journey of this project extracted a few phases important for the project life cycle. Therefore, the reflection of the report revolves around the parameters mentioned below:

Project life-cycle phases



4.1 Plan

The project plan included designing an online application to implement the firewall functionality. The purpose of the project was to design a platform that would be able to process the firewall rules. The idea was to make the platform readily available for students and researchers to test the rules and eventually make a strong and efficient security system. The project is designed to answer a few research questions which are mentioned in the beginning of the report. Furthermore, the phased approach in designing the project supported the efficiency of the project. “It can also be concluded that at the end of the project plan that the project provides an efficient and a simple platform for testing the firewall". It also embodies the output from the firewall to a variety of rule set. The project also opened new doors to the future researchers to implement the designed chain/rule set in a virtual environment with minimalistic infrastructure.

4.2 Research

Research is imperative for improving the knowledge base systematically which can be eventually used to identify the deficit in the existing theories. These deficits can be minimised by using the results derived from the research in building new concepts or reflecting upon the existing ones (Pramodini, 2002).

The literature review conducted for the project lists plenty of work carried out around improving the efficiency of the firewall system. These studies suggested a variety of methods for designing firewall rules to avoid conflicts in the chains. However, the rules still require a platform to be tested upon before using it in organisations or critical systems. This project suggests a platform where users could test these rules and make required amendments, eventually, improving the firewall chain. The idea for the project is to use HTTP to implement a firewall in a client-server environment. The firewall simulator is inspired by the previous research work and is motivated to support the requirement of a readily available environment.

The research also pointed towards the existing technology performing a similar function with different tools like Virtual environment Emulator, Software Define Networking, Mininet, controllers, etc. For example, a few authors proposed implementation of a distributed firewall where each switch in the network acts as a firewall. The interesting approach is backed with a successful test of the firewall on a virtual machine (Kaur, Kaur and Gupta, 2016).

Another group of authors implemented the firewall using a controller. They used POX controller to implement the firewall virtually by using Virtual Box. They tested the firewall on the hardware as well. (Kaur, Kaur and Gupta, 2016).

Therefore, testing has been evidenced to be pivotal for implementing a firewall. However, the testing environment is generally emulated using a virtual environment which requires an emulator tool to emulate linux based environment. Although, implementing a software firewall is more economical than using a hardware firewall. However, implementing a software firewall is complex and requires a substantial understanding of virtualisation. The idea of using HTTP to implement the firewall functionality, can tackle the complexity challenge involved in setting up the environment. HTTP can make it simple for user to directly access the firewall and its functionalities rather than investing efforts and time in emulating the environment. This project produces a prototype for a HTTP based firewall. With future work and advancement in the model, the project can provide a highly efficient firewall platform

4.3 Development

Like any other project, the development happened in phase. Post research, the primary phase in the project was designing a GUI. The user interface is kept generic due to time constraint. Furthermore, the GUI was updated often during the duration of the project due to the pertaining challenges. The development of GUI was followed by assigning values and function to different aspects of the program. Due to the lack of time and resources, only a few basic commands could be coded into the tool. These commands display the basic functionality of a generic packet filtering firewall. Different classes are designed separately and then clubbed together in a script. During the development phase, many errors were encountered. However, those challenges were addressed pertaining to research and required amendments. Post the scripts, the project focused on designing and programming the buttons to upload and download the rules. The save file and bootstrap scripts found during the research was included at this step as well to achieve the desired functionality of the project.

4.4 Challenges

The lifecycle of the project encountered many challenges. These challenges related to the design, code and functionality. This section of the report is the reflection of these challenges.

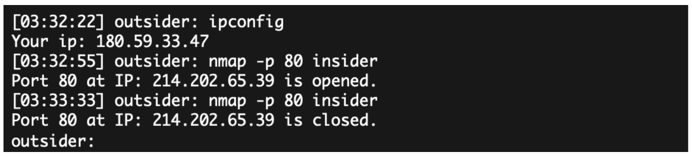
A. Functionality challenges: The terminal was expected to perform all the basic functionality of the firewall. The desired functionality required the terminals to be able to send the ICMP stream to each other. This is achieved by using the ping command. However, due to lack of knowledge and time, this could not be achieved in the project
B. GUI: One of the major challenges faced during the development of GUI was to display a blinking cursor on the terminal. In the initial phases of project development, the terminal was designed without a blinking cursor. However, it could not replicate the functionality of a terminal without the cursor. Henceforth, many open-source codes and Q&A website provided an idea which made the code possible
C. Presentation: The tool is presented using a simple GUI which presents the user with two terminals. The usability of the tool is not efficient as it does not explain the user what should be done. Although, the HTML landing page recommends a few commands that could be used but still does not enlighten the user on the capability of the tools
D. Operability: Although the project was able to demonstrate the basic functionality of a firewall, however, the model could not behave as a modern-day firewall. The functionality inherited by the firewall is of a packet filtering firewall. The modern security demands an upgraded version of the firewall like stateful inspection which analyses the entire traffic (Anwar, Abdullah and Pastore, 2020). Due to time constraint the project focused on the functionality of a basic packet filtering firewall as stateful inspection could not be achieved
E. Rules: As previously mentioned, the simulator is designed to perform the basic functionality. However, there is no governance on the rules to be entered by the user. The simulator will accept a few rules which might not be possible in the real world

4.5 Results and evaluation

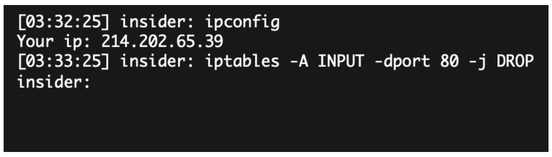
This section of the report focuses on the results achieved after testing the designed model. All the assigned functionalities are tested one by one. The results from all the tests are collated and evaluated.

Basic Packet Filtering

The functionality tested first is the basic packet filtering which is imperative to describe the standard operating procedure for the project. The findings are presented below.



```
[03:32:22] outsider: ipconfig
Your ip: 180.59.33.47
[03:32:55] outsider: nmap -p 80 insider
Port 80 at IP: 214.202.65.39 is opened.
[03:33:33] outsider: nmap -p 80 insider
Port 80 at IP: 214.202.65.39 is closed.
outsider:
```

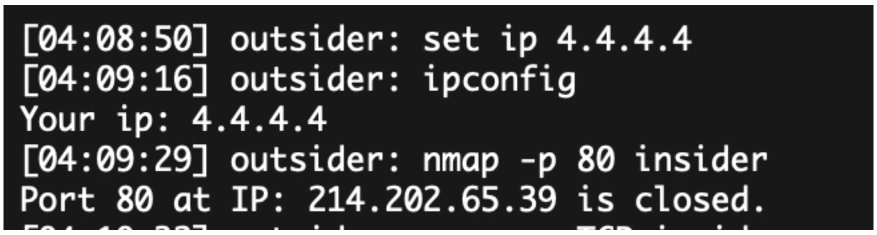


```
[03:32:25] insider: ipconfig
Your ip: 214.202.65.39
[03:33:25] insider: iptables -A INPUT -dport 80 -j DROP
insider:
```

Figure 4(ii)

The first step is to check the IP address of both the terminals. This is achieved by using ipconfig. The outsider terminal then checks if the terminal inside the network is accepting the packets on the TCP ports by checking if the port is open. Figure 4(ii) displays that the TCP port on the terminal inside the network is open. A rule is then appended which indicates the firewall to drop all the packets destined for the TCP port for the insider. After the successful implementation of the rule, nmap is used again on the outsider to check if the terminal inside the network is still accepting the packets on port 80. The result highlights the closure of the port 80 which implies all the packets sent to insider on port 80 will be dropped.

Checking packet filtering after changing the IP address of the outsider

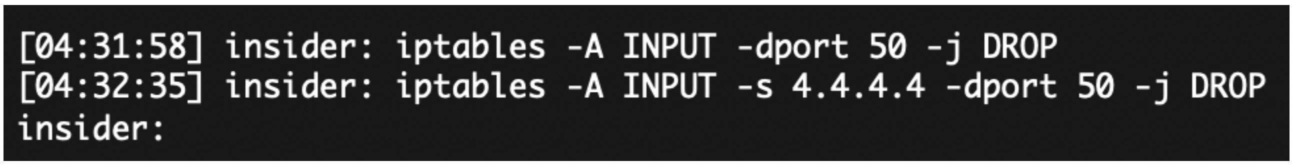


```
[04:08:50] outsider: set ip 4.4.4.4
[04:09:16] outsider: ipconfig
Your ip: 4.4.4.4
[04:09:29] outsider: nmap -p 80 insider
Port 80 at IP: 214.202.65.39 is closed.
```

Figure 4(iii)

The set ip command is used to change the IP address of the outsider which means changing the location of the outsider. As in the previous step, all the packets receiving at port 80 are dropped. Therefore, even after changing the ip address as well nmap confirms that the port is still closed.

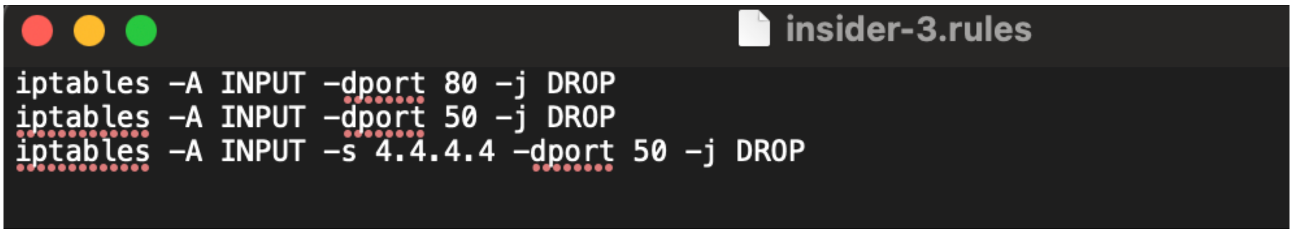
Downloading the rules
The next step is to check the functionality of the download rules button. To check this functionality first a couple of rules are appended to the INPUT chain



```
[04:31:58] insider: iptables -A INPUT -dport 50 -j DROP
[04:32:35] insider: iptables -A INPUT -s 4.4.4.4 -dport 50 -j DROP
insider:
```

Figure 4(iv)

Figure 4(iv) confirms that 2 new rules have been appended to the INPUT chain. The first rule drops all the packets arriving at port 50 and the second rule drops the packets coming from 4.4.4.4. This example also highlights the drawback in the system. The designed system is effective to test the rules, however, is not advisable to design the rules because the simulator will accept both the rules and the third rule is redundant, owing to the second rule. Henceforth, the simulator is best suitable with the effective rule configuration protocols mentioned before. To test the functionality of the download button, a file is downloaded on the system.



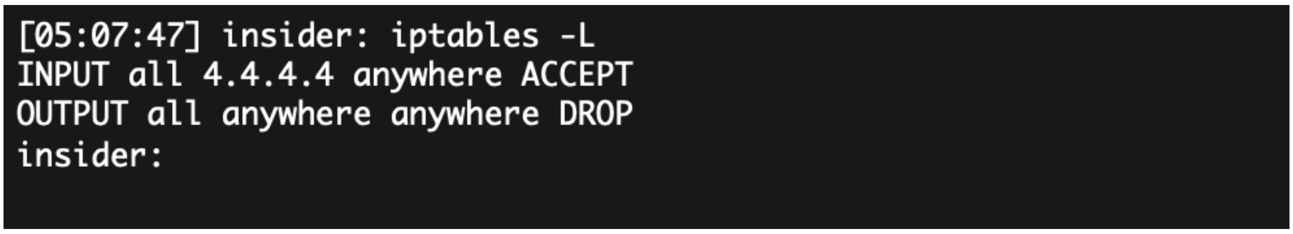
```
insider-3.rules
iptables -A INPUT -dport 80 -j DROP
iptables -A INPUT -dport 50 -j DROP
iptables -A INPUT -s 4.4.4.4 -dport 50 -j DROP
```

Figure 4(v)

It is evident from Figure 4(v), all the three rules added to the insider can be seen in the downloaded file (see Appendix V)

Uploading the rules
This function is imperative while dealing with large number of rules. Often, the firewall needs to be programmed with plenty of rules to achieve a particular level of security. Therefore, instead of

typing all the rules individually, a file is created called bespoke.rules which contains 2 rules (see Appendix V) for the purpose of testing. However, in real world this function is useful in handling large chunk of rules. After uploading the file using upload button the inside terminal is checked for the existing rules. Figure 4(VI) displays the uploaded rules.



```
[05:07:47] insider: iptables -L
INPUT all 4.4.4.4 anywhere ACCEPT
OUTPUT all anywhere anywhere DROP
insider:
```

Figure 4(VI)

The tests performed on the simulator confirms that the terminals can perform the functionality of the firewall. This implies the successful implementation of iptables. There are some existing open-source projects which have already implemented the iptables technology easily available on the internet, which implies that it could be used directly in the projects without the need of designing one from the scratch. However, for this project the iptables functionality has been designed from the scratch owing to the following requirements:

1. Iptables available on the internet are designed in C, and it requires a compiler and run time environment for implementation adding to the need of infrastructure and knowledge
2. The platform used for the project is a webpage to make the firewall independent of the infrastructure. Hence, instead of testing the firewall in virtual environments and on hardware firewalls, the users can test it on the local computer or a webserver eradicating the requirement of an emulated environment, infrastructure and understanding of the programming skills

4.6 Future work

There are multiple challenges and drawbacks in the project. Many of these challenges have been addressed in the previous sections of the report. Furthermore, these challenges and gaps in the existing study leads to the origin of future work that can be done in this area. As the project enables the user to test the firewall rules, so the user can use existing efficient techniques to design effective rules which prevents misconfiguration. However, the efficiency of the rules can be tested by using this model.

Primarily, the future work could involve hosting this tool on a webserver to provide access to the user from different machines without the need of transporting the tool using data storage devices. Secondary advancements include inculcating Netfilter functionality to the tool. Netfilter is an upgraded version of iptables with multiple extended features like stateless and stateful firewall, NAT and port translation, packet logging etc. Netfilter is a successor of iptables (Netfilter,2021). Therefore, implementing Netfilter instead of iptables can improve the functionality of the project. The project lacked some crucial components of Linux and networking. For instance, the terminals are not able to communicate with each other or send an ICMP stream to a device. Adding this functionality will not only support the project to look better on a visual ground but also improve the overall performance of the emulator.
Furthermore, this emulator was designed with an idea of making a technology/application which is imperative for testing. Hence, the simulator can be clubbed with existing studies which involves different methods for designing efficient firewall rules. The amalgamation of these concepts will lead to a development of an efficient firewall system.

The report concludes the efficient functionality of an emulated firewall system. Furthermore, it is safe to say that the project was successful and performed all the functions as planned. The future work mentioned in the former section will provide added feature to the firewall along with expanding the coverage of the commands. Henceforth, the project has shown a real-world implementation and tremendous growth over the life cycle of the project.

References

1. Abie, H., 2000.An Overview of firewall technology. Research Gate, pp.1-2.
2. Abie, H., 2000. CORBA Firewall Security:Increasing the Security of CORBA Applications. Research Gate, pp.1-2.
3. Al-Shaer, E. and Hamed, H., 2004. Modeling and Management of Firewall Policies. IEEE Transactions on Network and Service Management, 1(1), pp.2-10.
4. Anbari, F., Carayannis, E. and Voetsch, R., 2008. Post-project reviews as a key project management competence. Technovation, 28(10), pp.633-643. [Accessed 11 July 2022].
5. Andress, J., 2014. The Basics of Information Security, 2nd Edition. 2nd ed. Oxford: Elsevier Inc., pp.151-152.
6. Antonov, A.P., Filippov, A.S. & Mamoutova, O.V., 2016. Next generation FPGA-based platform for network security. 2016 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT).
7. Anwar, R., Abdullah, T. and Pastore, F., 2020. Firewall Best Practices for Securing Smart Healthcare Environment: A Review. Applied Sciences, 11(19), p.9183.
8. Au-Yong-Oliveira, M., Gonçalves, R., Martins, J. and Branco, F., 2018. The social impact of technology on millennials and consequences for higher education and leadership. Telematics and Informatics, 35(4), pp.954-963.
9. Buchanan, E., Aycock, J., Dexter, S., Dittrich, D. and Hvizdak, E., 2011. Computer Science Security Research and Human Subjects: Emerging Considerations for Research Ethics Boards. Journal of Empirical Research on Human Research Ethics, 6(2), pp.71-83.
10. Chhetri, Nimesh, "A Comparative Analysis of Node.js (Server-Side JavaScript)" (2016). Culminating Projects in Computer Science and Information Technology. 5. [Accessed 14 July 2022].
11. Cisco. 2022. What Is a Firewall?. [online] Available at: <https://www.cisco.com/c/en_uk/products/security/firewalls/what-is-a-firewall.html> [Accessed 13 July 2022].
12. Corrêa Souza, A., Nunes, F. and Delamaro, M., 2018. An automated functional testing approach for virtual reality applications. Software Testing, Verification and Reliability, 28(8), p.e1690.
13. Docs.oracle.com. 2013. 3.12.1.2 Inserting Rules in a Chain. [online] Available at: <https://docs.oracle.com/cd/E37670_01/E36387/html/ol_insfwchains_sec.html> [Accessed 13 July 2022].
14. Eychenne, H., 2019. Man page of IPTABLES. [online] Ipset.netfilter.org. Available at: <https://ipset.netfilter.org/iptables.man.html> [Accessed 9 July 2022].
15. Flynn, S. and Levie, F., 2021. Towards reflective project management. Irish Journal of Technology Enhanced Learning, 6(1), pp.118-130. [Accessed 18 July 2022].
16. Gouda, M. and Liu, A., 2007. Structured firewall design. Computer Networks, 51(4), pp.1106-1120.
17. Gov.uk, 2022. Computer Misuse Act 1990. [online] Legislation.gov.uk. Available at: <https://www.legislation.gov.uk/ukpga/1990/18/contents> [Accessed 18 July 2022].
18. Hathaway, O., Crootof, R., Levitz, P., Nix, H., Nowlan, A., Perdue, W. and Spiegel, J., 2012. The Law of Cyber-Attack. California Law Review, [online] 100(4), pp.817-885. Available at: <https://www.jstor.org/stable/23249823> [Accessed 15 July 2022].
19. He, X., 2020. Research on Computer Network Security Based on Firewall Technology. Journal of Physics: Conference Series, 1744(4), p.042037.
20. Kamara, S. et al., 2003. Analysis of vulnerabilities in internet firewalls. Computers & Security, 22(3), pp.214–232.

21. Kaur, S., Kaur, K. and Gupta, V., 2016. "Implementation of stateful firewall using POX controller," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 1093-1096.
22. Kaur, S., Kaur, K. and Gupta, V., 2016. Implementing openflow based distributed firewall. 2016 International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds.
23. Kurose, J. and Ross, K., 2013. Computer Networking | A Top-Down approach. 6th ed. New Jersey: Pearson, pp.185-189.
24. Liu, A., 2007. Change-Impact Analysis of Firewall Policies. Computer Security – ESORICS 2007, pp.155-170.
25. Liu, A. and Gouda, M., 2008. Diverse Firewall Design. IEEE Transactions on Parallel and Distributed Systems, 19(9), pp.1237-1251.
26. Marmorstein, R. and Kearns, P., 2005. A Tool for Automated iptables Firewall Analysis. In: A Tool for Automated iptables Firewall Analysis. [online] California: USENIX Annual Technical Conference, pp.1-3. Available at: <https://www.usenix.org/legacy/event/usenix05/tech/freenix/full_papers/marmorstein/marmorstein_html/> [Accessed 14 July 2022].
27. Mundra, S. and Taeib, T., 2015. TCP/IP PROTOCOL LAYERING. International Journal of Computer Science and Information Technology Research, [online] 3(1), pp.415-417. Available at: <http://www.researchpublish.com> [Accessed 15 July 2022].
28. Nath, P. and Uddin, M., 2015. TCP-IP Model in Data Communication and Networking. American Journal of Engineering Research (AJER), 4(10), pp.102-107.
29. Netfilter.org. 2021. netfilter/iptables project homepage - The netfilter.org project. [online] Available at: <https://www.netfilter.org/> [Accessed 18 July 2022].
30. Netto, M., Menon, S., Vieira, H., Costa, L., de Oliveira, F., Saad, R. and Zorzo, A., 2011. Evaluating Load Generation in Virtualized Environments for Software Performance Testing. 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum,.
31. Northrup, T., 2009. Firewalls. [online] Technet.microsoft.com. Available at: <https://technet.microsoft.com/en-gb/library/cc700820.aspx> [Accessed 15 July 2022].
32. Openhub.net. 2020. The iptables Open-Source Project on Open Hub. [online] Available at: <https://www.openhub.net/p/iptables> [Accessed 19 July 2022].
33. Patrignani, N., 2018. A Conceptual Framework for Computer Ethics – The Research Center on Values in Emerging Science and Technology. [online] Rcvest.southernct.edu. Available at: <https://rcvest.southernct.edu/a-conceptual-framework-for-computer-ethics/> [Accessed 11 July 2022].
34. PRAMODINI D V (2022) "EVALUATION OF IMPORTANCE FOR RESEARCH IN EDUCATION", INTERNATIONAL JOURNAL OF SOCIAL SCIENCE & INTERDISCIPLINARY RESEARCH ISSN: 2277-3630 Impact factor: 7.429, 11(01), pp. 255–260. Available at: http://www.gejournal.net/index.php/IJSSIR/article/view/184 (Accessed: 21 July 2022).
35. Schön, D. (1983). The reflective practitioner: how professionals think in action. New York: Basic Books. [Accessed 14 July 2022].
36. Shah, M., Ahmed, S., Saeed, K., Junaid, M., Khan, H. and ur-rehman, A., 2019. Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool. 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)
37. Slider, N., 2021. HowTos/Network/IPTables - CentOS Wiki. [online] Wiki.centos.org. Available at: <https://wiki.centos.org/HowTos/Network/IPTables> [Accessed 11 July 2022].

38. Veale, M. and Brown, I., 2020. Cybersecurity. Internet Policy Review, 9(4). [Accessed 4 July 2022].
39. Yuan, L., Chen, H., Mai, J., Chuah, C., Su, Z. and Mohapatra, P., 2006. FIREMAN: a toolkit for firewall modeling and analysis. 2006 IEEE Symposium on Security and Privacy (S&amp;P'06),.
40. Zhang, L. and Huang, M., 2015. A Firewall Rules Optimized Model Based on Service-Grouping. 2015 12th Web Information System and Application Conference (WISA),.

Appendices

Appendix I

HTML – Hyper Text Markup Language

HTML is a language used to design documents to be displayed on the browser. It works in conjunction with CSS, JavaScript and other languages

CSS – Cascading Style Sheet

CSS is used for the presentation of the documents written in HTML or XML

NPM – Node Packet Manager

NPM is a packet manager used for JavaScript containing both paid and free packages.

Appendix II

Commands Processor class from JavaScript code :

```
const Message     = require("./Message");
const util        = require("../util");
const rule        = require("./ipTableRule");

class CommandsProcessor {
   static tokenise(msg) {
      return msg.split(" ");
   }


   static isValidIpv4(token) {
      const chunks = token.split('.');
      for (let c of chunks) {
         if (!c.match(/^[0-9\/]+$/i))
            return false;
         if (c < 0 || c > 255)
            return false;
      }
      return true;
   }

   findToken(msg, flag, regexps) {
      var regexpExec = flag + '\\s';
      var i = 0;

      // determine initial positions (ie. input: flag: -I { chain: "\\w+", position: "\\w+" } => { chain: 0,
position: 1 }
      // and generate regexp, e.g. "-I\s(\w+)\s(\w+)\s"
      for (var regexp in regexps) {
        if (!regexps.hasOwnProperty(regexp)) continue;
        regexpExec += '(' + regexps[regexp] + ')\\s';
        regexps[regexp] = i;
        i++;
    }

    // remove trailing \\s, ie. "-I\s(\w+)\s(\w+)\s" => "-I\s(\w+)\s(\w+)"
    regexpExec = regexpExec.substr(0, regexpExec.length - 2);

    // e.g. "iptables -I INPUT 4 hello" => ["-I INPUT 4", "INPUT", "4"]
    const matches =  new RegExp(regexpExec, 'g').exec(msg);

    if (!matches)
        return {found: false};

    // re-assign to original property, ie. chain: INPUT, position: 4
    for (i = 1; i < matches.length; i++)
        regexps[Object.keys(regexps).find(key => regexps[key] === i - 1)] = matches[i];

    // marges objects
    return Object.assign({found: true}, regexps);

}

error() {
    if (this.terminal === undefined)
        return { action: "error" };
    this.terminal.add(new Message("bash","Invalid rule."));
    return { action: "error" };
}

processIpRule(msg) {
    var newRule = rule();
    newRule.saveString(msg);

    const word = "\\w+";

    const fEmpty    = this.findToken(msg, "-F",       {}                                );
    const f         = this.findToken(msg, "-F",       { chain: word }                   );
    const a         = this.findToken(msg, "-A",       { chain: word }                   );
    const i         = this.findToken(msg, "-I",       { chain: word, position: word}    );
    const d         = this.findToken(msg, "-D",       { chain: word, position: word}    );
    const s         = this.findToken(msg, "-s",       { sourceAddr: "[\\.|\\/|\\w]+"}   );
    const p         = this.findToken(msg, "-p",       { protocol: word }                );
    const j         = this.findToken(msg, "-j",       { jump: word }                    );
    const dport     = this.findToken(msg, "--dport",  { port: word }                    );

    // build up rule
    if (a.found)
        newRule.chain(a.chain);
        if (i.found)
            newRule.chain(i.chain);
        if (s.found)
            newRule.source(s.sourceAddr);
        if (p.found) {
            if (p.protocol === "tcp" || p.protocol === "udp" || p.protocol === "all")
                newRule.protocol(p.protocol);
            else
                this.error();
        }
        if (j.found)
            newRule.jump(j.jump);
        if (dport.found) {
            if (dport.port in util.dports)
                newRule.port(util.dports[dport.port]);
            else if (dport.port > 0 && dport.port < 65535)
                newRule.port(dport.port);
            else
                this.error();
        }
        if (this.terminal === undefined)
            return newRule.build();

        if (f.found)
            this.terminal.ipTable.resetChain(f.chain);
        else if (fEmpty.found)
            this.terminal.ipTable.resetRules();
        else if (a.found)
            this.terminal.ipTable.addRule(newRule.build());
        else if (i.found)
            this.terminal.ipTable.insertRule(i.chain, i.position, newRule.build());
        else if (d.found)
            this.terminal.ipTable.deleteRule(d.chain, d.position);
        else
            this.error();
    }

    processNmap(tokens) {
        // our version requires "nmap -p <port> <address>"
        if (tokens.length !== 4)
            return this.error();

        const sourceAddr = this.terminal.ip;
        const port = tokens[2];
        var targetTerminal;
        if (this.terminal.user_name === tokens[3] || this.terminal.ip === tokens[3])
            targetTerminal = this.terminal;
        else if (this.terminal.otherTerminal.user_name === tokens[3] || this.terminal.otherTerminal.ip
=== tokens[3])
            targetTerminal = this.terminal.otherTerminal;
        else
            return this.terminal.add(new Message("bash", "Target ip address not known."));

        if (targetTerminal.ipTable.isPortOpened(port, sourceAddr))
            return this.terminal.add(new Message("console", "Port " + port +" at IP: " +
targetTerminal.ip + " is opened."));
        else
            return this.terminal.add(new Message("console", "Port " + port +" at IP: " +
targetTerminal.ip + " is closed."));
    }

    process(terminal, msg) {
        this.terminal = terminal;
        const tokens = CommandsProcessor.tokenise(msg);

        if (tokens[0] === "iptables") {
            if (tokens[1] === "-L") {
                // display current rules
                for (let r of terminal.ipTable.listRules()) {
                    terminal.add(new Message("console", r));
                }
            } else {
                this.processIpRule(msg);
            }
            return;
        } else if (tokens[0] === "nmap") {
            this.processNmap(tokens);
            return;
        } else if (tokens[0] === "set" && tokens[1] === "ip") {
            if (CommandsProcessor.isValidIpv4(tokens[2]))
                terminal.ip = tokens[2];
            else
                terminal.add(new Message("console", "Invalid IPv4 address: " + tokens[2]));
            return;
        }

        switch (msg) {
            case "":
                break;
            case "del msgs":
                terminal.deleteMsgs();
                break;
            case "clear":
                terminal.clearMessages();
                break;
            case "save msgs":
                terminal.msgManager.save();
                break;
            case "ipconfig":
                terminal.add(new Message("console", "Your ip: " + terminal.ip));
            break;
          default:
            terminal.add(new Message("bash", msg + ": Command not known."));
            break;
        }
    }

}

module.exports = CommandsProcessor;
```

Appendix III

FileSaver.js: FileSaver.js is the JavaScript code used to save a file to the system. As the project involves downloading a file including the rules on the system. It needed a JavaScript code. The code package was taken from git hub and is mentioned below:

```
/* FileSaver.js
 * A saveAs() FileSaver implementation.
 * 1.3.2
 * 2016-06-16 18:25:19
 *
 * By Eli Grey, http://eligrey.com
 * License: MIT
 *   See https://github.com/eligrey/FileSaver.js/blob/master/LICENSE.md
 */

/*global self */
/*jslint bitwise: true, indent: 4, laxbreak: true, laxcomma: true, smarttabs: true, plusplus: true */

/*! @source http://purl.eligrey.com/github/FileSaver.js/blob/master/FileSaver.js */

var saveAs = saveAs || (function(view) {
	"use strict";
	// IE <10 is explicitly unsupported
	if (typeof view === "undefined" || typeof navigator !== "undefined" && /MSIE [1-
9]\./.test(navigator.userAgent)) {
		return;
	}
	var
		  doc = view.document
		  // only get URL when necessary in case Blob.js hasn't overridden it yet
		, get_URL = function() {
			return view.URL || view.webkitURL || view;
		}
		, save_link = doc.createElementNS("http://www.w3.org/1999/xhtml", "a")
		, can_use_save_link = "download" in save_link
		, click = function(node) {
			var event = new MouseEvent("click");
			node.dispatchEvent(event);
```

```
        }
        , is_safari = /constructor/i.test(view.HTMLElement) || view.safari
        , is_chrome_ios =/CriOS\/[\d]+/.test(navigator.userAgent)
        , throw_outside = function(ex) {
            (view.setImmediate || view.setTimeout)(function() {
                throw ex;
            }, 0);
        }
        , force_saveable_type = "application/octet-stream"
        // the Blob API is fundamentally broken as there is no "downloadfinished" event to
subscribe to
        , arbitrary_revoke_timeout = 1000 * 40 // in ms
        , revoke = function(file) {
            var revoker = function() {
                if (typeof file === "string") { // file is an object URL
                    get_URL().revokeObjectURL(file);
                } else { // file is a File
                    file.remove();
                }
            };
            setTimeout(revoker, arbitrary_revoke_timeout);
        }
        , dispatch = function(filesaver, event_types, event) {
            event_types = [].concat(event_types);
            var i = event_types.length;
            while (i--) {
                var listener = filesaver["on" + event_types[i]];
                if (typeof listener === "function") {
                    try {
                        listener.call(filesaver, event || filesaver);
                    } catch (ex) {
                        throw_outside(ex);
                    }
                }
            }
        }
        , auto_bom = function(blob) {
            // prepend BOM for UTF-8 XML and text/* types (including HTML)
            // note: your browser will automatically convert UTF-16 U+FEFF to EF BB
BF
            if (/^\s*(?:text\/\S*|application\/xml|\S*\/\S*\+xml)\s*;.*charset\s*=\s*utf-
8/i.test(blob.type)) {
                return new Blob([String.fromCharCode(0xFEFF), blob], {type:
blob.type});
            }
            return blob;
        }
        , FileSaver = function(blob, name, no_auto_bom) {
            if (!no_auto_bom) {
                blob = auto_bom(blob);
```

```
				}
				// First try a.download, then web filesystem, then object URLs
				var
					  filesaver = this
					, type = blob.type
					, force = type === force_saveable_type
					, object_url
					, dispatch_all = function() {
						dispatch(filesaver, "writestart progress write writeend".split("
"));
					}
					// on any filesys errors revert to saving with object URLs
					, fs_error = function() {
						if ((is_chrome_ios || (force && is_safari)) &&
view.FileReader) {
							// Safari doesn't allow downloading of blob urls
							var reader = new FileReader();
							reader.onloadend = function() {
								var url = is_chrome_ios ? reader.result :
reader.result.replace(/^data:[^;]*;/, 'data:attachment/file;');
								var popup = view.open(url, '_blank');
								if(!popup) view.location.href = url;
								url=undefined; // release reference before
dispatching
								filesaver.readyState = filesaver.DONE;
								dispatch_all();
							};
							reader.readAsDataURL(blob);
							filesaver.readyState = filesaver.INIT;
							return;
						}
						// don't create more object URLs than needed
						if (!object_url) {
							object_url = get_URL().createObjectURL(blob);
						}
						if (force) {
							view.location.href = object_url;
						} else {
							var opened = view.open(object_url, "_blank");
							if (!opened) {
								// Apple does not allow window.open, see
https://developer.apple.com/library/safari/documentation/Tools/Conceptual/SafariExtensionGuide/
WorkingwithWindowsandTabs/WorkingwithWindowsandTabs.html
								view.location.href = object_url;
							}
						}
						filesaver.readyState = filesaver.DONE;
						dispatch_all();
						revoke(object_url);
					}
```

```
                ;
                filesaver.readyState = filesaver.INIT;

                if (can_use_save_link) {
                        object_url = get_URL().createObjectURL(blob);
                        setTimeout(function() {
                                save_link.href = object_url;
                                save_link.download = name;
                                click(save_link);
                                dispatch_all();
                                revoke(object_url);
                                filesaver.readyState = filesaver.DONE;
                        });
                        return;
                }

                fs_error();
        }
        , FS_proto = FileSaver.prototype
        , saveAs = function(blob, name, no_auto_bom) {
                return new FileSaver(blob, name || blob.name || "download", no_auto_bom);
        }
;
// IE 10+ (native saveAs)
if (typeof navigator !== "undefined" && navigator.msSaveOrOpenBlob) {
        return function(blob, name, no_auto_bom) {
                name = name || blob.name || "download";

                if (!no_auto_bom) {
                        blob = auto_bom(blob);
                }
                return navigator.msSaveOrOpenBlob(blob, name);
        };
}

FS_proto.abort = function(){};
FS_proto.readyState = FS_proto.INIT = 0;
FS_proto.WRITING = 1;
FS_proto.DONE = 2;

FS_proto.error =
FS_proto.onwritestart =
FS_proto.onprogress =
FS_proto.onwrite =
FS_proto.onabort =
FS_proto.onerror =
FS_proto.onwriteend =
        null;

return saveAs;
```

```
}(
        typeof self !== "undefined" && self
      || typeof window !== "undefined" && window
      || this.content
));
// `self` is undefined in Firefox for Android content script context
// while `this` is nsIContentFrameMessageManager
// with an attribute `content` that corresponds to the window

if (typeof module !== "undefined" && module.exports) {
  module.exports.saveAs = saveAs;
} else if ((typeof define !== "undefined" && define !== null) && (define.amd !== null)) {
  define("FileSaver.js", function() {
    return saveAs;
  });
}
```

Appendix IV

Bootstrap: As bootstrap is an open-source package. Therefore, it has been included in the project from the available libraries as mentioned below:

For JavaScript code: http://netdna.bootstrapcdn.com/bootstrap/3.1.1/js/bootstrap.min.js
For CSS code: http://netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css

Appendix V

Contents of the doCwnloaded file:

```
iptables -A INPUT -dport 80 -j DROP
iptables -A INPUT -dport 50 -j DROP
iptables -A INPUT -s 4.4.4.4 -dport 50 -j DROP
```

Contents of the uploaded file:

```
iptables -A INPUT -s 4.4.4.4 -dport 50 -j ACCEPT
iptables -A OUTPUT -dport 80 -j DROP
```

Appendix VI

Working code of the project. The project code was divided into different classes which were brought together to display the basic functionality of Object-Oriented Programming. Each class is coded separately and is presented below:

CommandProcessor.js: See Appendix II

iptables.js: const util   = require("../util");
const Netmask   = require('netmask').Netmask;

```
class IpTable {
  constructor(key) {
    this.key = key;
    this.rules = [];
  }

  _loadHistory() {
    const history = JSON.parse(window.localStorage.getItem(this.key));
    if (history) {
      const table = history[0];
      table.forEach(function(rule) {
        var newRule = rule();
        newRule.info = rule;
        this.addRule(this.tables[i], newRule.build());
      });
    }
  }

  addRule(rule) {
    if (this._tableContainsRule(rule))
      return false;

    this.rules.push(rule);
    return true;
  }

  insertRule(chain, position, rule) {
    if (this._tableContainsRule(rule))
      return false;

    var chain_rules = this.filterByChain(chain);
    if (position >= chain_rules.length)
      return false;
    chain_rules.splice(position, 0, rule);
    var other_rules = this.otherThanChain(chain);
    this.rules = other_rules.concat(chain_rules);
    return true;
  }

  deleteRule(chain, position) {
    var chain_rules = this.filterByChain(chain);
    if (position >= chain_rules.length)
      return false;
    chain_rules.splice(position, 1);
    var other_rules = this.otherThanChain(chain);
    this.rules = other_rules.concat(chain_rules);
    return true;
  }
```

```
listRules() {
   var response = [];
   for (let c of this.getUniqueChains()) {
      for (let r of this.filterByChain(c)) {
         response.push(r.toString());
      }
   }

   return response;
}

resetRules() {
   this.rules = [];
   window.localStorage.setItem(this.key, "");
}

resetChain(chain) {
   this.rules = this.rules.filter(function(rule) { return rule.chain !== chain; });
}

getUniqueChains() {
   var chains = [];
   for (var rule in this.rules) {
      chains.push(this.rules[rule].chain);
   }

   return chains.filter(function(value, index, self) { return self.indexOf(value) === index; });
}

isPortOpened(port, sourceAddr = "anywhere", chain = util.tableTypes.FILTER.chain.INPUT) {
   chain = chain.trim();
   const rules = this.filterByChain(chain);

   // console.log(this.isAddressWithinRange(rules[0].source, sourceAddr));

   // find first rule regarding our port number
   for (let r of rules) {
      // console.log(r);
      if (r.source === "anywhere" || this.isAddressWithinRange(r.source, sourceAddr)) {
         if (r.port === parseInt(port) || r.port === "") {
            // console.log(r);
            if (r.jump === "ACCEPT")
               return true;
            else if (r.jump === "DROP")
               return false;
            else
               return this.isPortOpened(port, sourceAddr, r.jump);
         }
      }
   }
```

```
        return true; // default!
    }

    isAddressWithinRange(ruleAddr, sourceAddr) {
        if (ruleAddr === sourceAddr)
            return true;
        return (new Netmask(ruleAddr)).contains(sourceAddr);

    }

    filterByChain(chain) {
        return this.rules.filter(function(rule) { return rule.chain === chain; });
    }

    otherThanChain(chain) {
        return this.rules.filter(function(rule) { return rule.chain !== chain; });
    }


    _tableContainsRule(rule) {
        for (let r of this.rules) {
            if (util.deepEqual(r, rule)) {
                return true;
            }
        }
        return false;
    }
}

module.exports = IpTable;
```

ipTableRule.js:

```
const util       = require("../util");
const e_protocol  = util.protocol;
const e_ipv       = util.ipv;

onst rule = () => ({
    info: {
        chain          : "INPUT",
        ipv            : "",
        protocol       : e_protocol.ALL,
        port           : "",
        source         : "anywhere",
        destination    : "anywhere",
        match          : "",
        jump           : "",
        goto           : "",
        interface_in   : "",
        interface_out  : "",
    fragment      : "",
    counters      : "",
    saveString    : "",
},

chain(target) {
    this.info.chain = target.toUpperCase();
    return this;
},

ipv(ipv) {
    const ipv_up = ipv.toUpperCase();
    if (ipv_up in e_ipv) {
        this.info.ipv = e_ipv[ipv_up];
        return this;
    }
},

protocol(protocol) {
    const p_up = protocol.toUpperCase();
    if (p_up in e_protocol) {
        this.info.protocol = e_protocol[p_up];
        return this;
    }
},

port(port) {
    this.info.port = parseInt(port);
    return this;
},

source(source) {
    this.info.source = source;
    return this;
},

destination(dest) {
    this.info.destination = parseInt(dest);
    return this;
},

match(match) {
    this.info.match = match;
    return this;
},

jump(jump) {
    this.info.jump = jump;
    return this;
},
  goto(goto) {
    this.info.goto = goto;
    return this;
  },

  interface_in(int_in) {
    this.info.interface_in = int_in;
    return this;
  },

  interface_out(int_out) {
    this.info.interface_in = int_out;
    return this;
  },

  fragment(fragment) {
    this.info.fragment = fragment;
    return this;
  },

  counters(counters) {
    this.info.counters = counters;
    return this;
  },

  saveString(saveString) {
    this.info.saveString = saveString;
    return this;
  },

  build() {
    return new IpTableRule(this.info);
  }
});

class IpTableRule {
  constructor(info) {
    Object.assign(this, info);
  }

  toString() {
    var response = "";
    // response += this.chain      + "\t";
    // response += this.protocol   + "\t";
    // response += "--"    + "\t"; //opt? todo
    // response += this.source     + "\t";
    // response += this.destination + "\t";
    // response += this.port + "\t";
    // return response;
    for (var property in this) {
      if (property !== "" && property !== "saveString")
        response += this[property] + "\t";
    }
    return response;
  }
}

module.exports = rule;
```

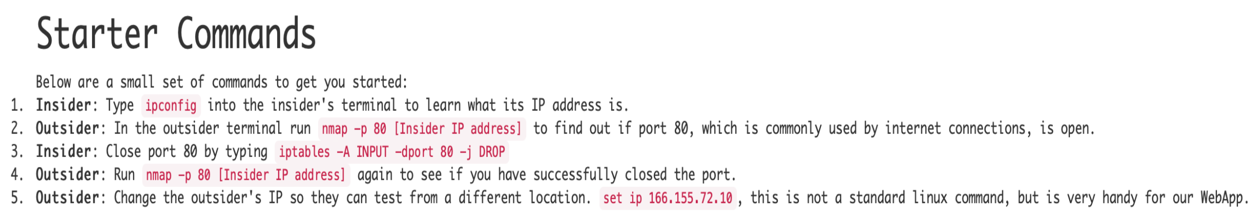
MessageManager.js:

```
const Message = require("./Message");
const util    = require("../util");

class MessagesManager {
  constructor(key) {
    this.key = key;
    this.msgs = [];

    this._upDownCurPlace = 0;

    this._loadHistory();
  }

  _loadHistory() {
    const history = JSON.parse(window.localStorage.getItem(this.key));

    if (history) {
      for (var i in history) {
        var m = history[i];
        this.msgs.push(new Message(m.author, m.message, m.timestamp));
      }
    }
  }

  _getUserMsgs() {
    var msgs = [];
    for (var i = 0; i < this.msgs.length; i++) {
      var m = this.msgs[i];
      if (m.author !== "bash" && m.author !== "console") {
        msgs.push(m);
      }
    }
    return msgs;
  }

  upDownUserMsg(keyCode) {
    var visited = 0;
    var userMsgs = this._getUserMsgs();

    for (var i = userMsgs.length - 1; i >= 0; i--) {

        visited++;

        if (keyCode === util.keyCodes.ARROW_UP) {

            if (visited > this._upDownCurPlace) {
                this._upDownCurPlace++;
                return userMsgs[i];
            }


        } else if (keyCode === util.keyCodes.ARROW_DOWN) {
            if (this._upDownCurPlace < 2) {
                // empty msg
                this._upDownCurPlace = 0;
                return;
            }
            if (visited === this._upDownCurPlace) {
                // return the previous
                this._upDownCurPlace--;
                return userMsgs[i + 1];
            }
        } else {
            // shouldn't be getting here!
            console.log("GTFO");
        }

    }

    // only case when it gets here is if we were already at the last place.
    // return first msg
    return userMsgs[0];
}

add(msg) {
    this.msgs.push(msg);
    this._upDownCurPlace = 0;
}

save() {
    window.localStorage.setItem(this.key, JSON.stringify(this.msgs));
}

delete() {
    this.msgs = [];
    window.localStorage.setItem(this.key, "");
    }
}

module.exports = MessagesManager;
```

Terminal.js:

```
const IpTable          = require("./IpTable");
const FileSaver        = require('file-saver');
const MessagesManager  = require("./MessagesManager");
const Message          = require("./Message");
const util             = require("../util");

class Terminal {
    constructor(elem, userName, cmdProcessor) {
        this.terminal       = elem;
        this._id            = elem.attr('id');
        this.elem_messages  = elem.find('.messages');
        this.elem_input     = elem.find('input');
        this.msgManager     = new MessagesManager(this._id);
        this.user_name      = userName;
        this.cmdProcessor   = cmdProcessor;

        this._init();
    }

    _init() {
        this._loadHistory();
        if (localStorage.getItem(this._id + "Ip") === null)
            this.ip = util.randomIp();
        else
            this.ip = localStorage.getItem(this._id + "Ip");

        this.elem_input.keyup((event) => this._listener(event));
        $('#' + this._id + 'File').change((event) => this._rulesUpload(event));
        $('#' + this._id + 'Download').click((event) => this._rulesDownload(event));

        this.changeUser(this.user_name);

        // iptables
        this.ipTable = new IpTable(this._id + "Rule");
    }

    _rulesUpload(event) {
        var f = event.target.files[0];
        var reader = new FileReader();
        reader.terminal = this;
        reader.onload = (function(reader)
        {
            return function()
            {
                reader.terminal.ipTable.resetRules();
                var contents = reader.result.split('\n');
                contents.forEach(function(line) {
                    if (line.trim().substr(0, 2) !== "##") {
                        reader.terminal.cmdProcessor.process(reader.terminal, line);
                    }
                });
                reader.terminal.cmdProcessor.process(reader.terminal, "iptables -L");
            };
        })(reader);
        reader.readAsText(f);
    }

    _rulesDownload(event) {
        const rules = this.ipTable.rules;
        var toSave = [];
        rules.forEach(function(rule) {
           toSave.push(rule.saveString);
        });
        var blob = new Blob([toSave.join('\n')], {type: "text/plain;charset=utf-8"});
        FileSaver.saveAs(blob, ((this._id === "i") ? "insider" : "ousider") + ".rules");
    }

    changeOtherTerminal(terminal) {
        this.otherTerminal = terminal;
    }

    changeUser(name) {
        this.user_name = name;
        this.terminal.find('.name').text(name + ":");
    }

    clearMessages() {
        this.elem_messages.html("");
    }

    _loadHistory() {
        if (!this.msgManager.msgs) return;

        this.msgManager.msgs.forEach((msg) => {
            this._append(msg);
        });
    }

    add(message) {
        this.msgManager.add(message);
        this._append(message);
    }
    _append(message) {
        this.elem_messages.append('<p>' + message.toString() + '</p>');
    }

    deleteMsgs() {
        this.msgManager.delete();
        this.clearMessages();
    }

    _listener(event) {
        var msg;
        switch (event.keyCode) {
            case (util.keyCodes.RETURN):
                msg = this.elem_input.val();

                msg = new Message(this.user_name, msg);
                this.add(msg);
                this.cmdProcessor.process(this, msg.message);
                this.elem_input.val("");
                this.msgManager.save();
                break;

            case (util.keyCodes.ARROW_UP):
            case (util.keyCodes.ARROW_DOWN):
                msg = this.msgManager.upDownUserMsg(event.keyCode);
                this.elem_input.val(typeof msg !== "undefined" ? msg.message : "");
                break;
        }
    }

    saveIp() {
        window.localStorage.setItem(this._id + "Ip", this.ip);
    }
}


module.exports = Terminal;
```

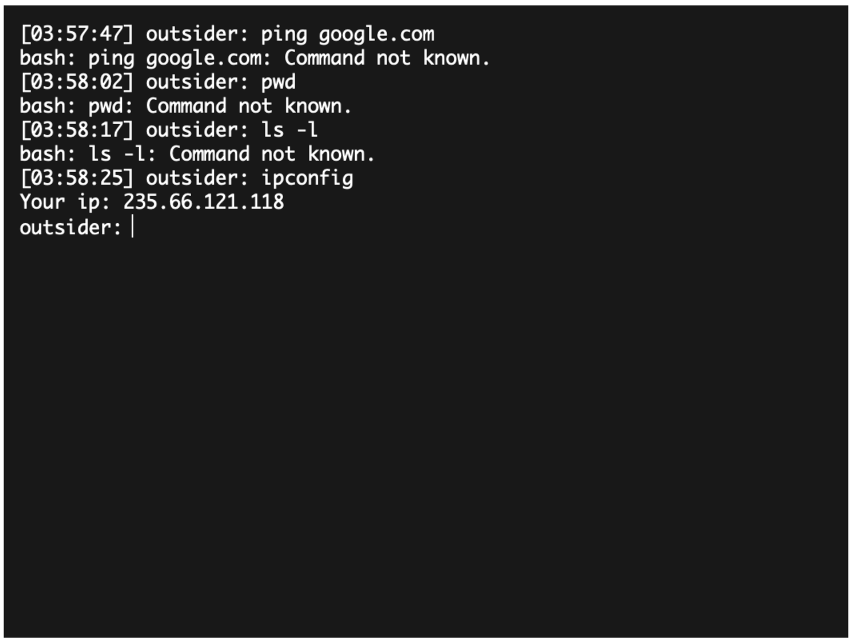
Appendix VII



Starter Commands

Below are a small set of commands to get you started:

1. **Insider**: Type `ipconfig` into the insider's terminal to learn what its IP address is.
2. **Outsider**: In the outsider terminal run `nmap -p 80 [Insider IP address]` to find out if port 80, which is commonly used by internet connections, is open.
3. **Insider**: Close port 80 by typing `iptables -A INPUT -dport 80 -j DROP`
4. **Outsider**: Run `nmap -p 80 [Insider IP address]` again to see if you have successfully closed the port.
5. **Outsider**: Change the outsider's IP so they can test from a different location. `set ip 166.155.72.10`, this is not a standard linux command, but is very handy for our WebApp.



```
[03:57:47] outsider: ping google.com
bash: ping google.com: Command not known.
[03:58:02] outsider: pwd
bash: pwd: Command not known.
[03:58:17] outsider: ls -l
bash: ls -l: Command not known.
[03:58:25] outsider: ipconfig
Your ip: 235.66.121.118
outsider: 
```