Deep Packet Inspection using Snort

by

Saad Hafeez
B.Eng., The Islamia University of Bahawalpur

A Report Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

Supervisory Committee

Deep Packet Inspection using Snort

by

Saad Hafeez
B.Eng., The Islamia University of Bahawalpur

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member
(Department of Electrical and Computer Engineering)

Abstract

Internet traffic analysis is of key interest to network designers, as efficient analysis leads to more efficient and fault-tolerant networks. Internet traffic can be analysed in different ways to perform tasks such as classification and filtration. Deep packet inspection is a means of filtering network traffic by monitoring a stream of packets and identifying strings of data that appear common. Based on information contained in the packet headers, or other protocol-specific parameters, we can distinguish the application of origin and even requests made by the application. In this report, Snort is used as a tool for deep packet inspection. Network traffic can be analysed by creating rules for web applications using Snort. Further, different policies can be implemented on Snort for deep packet inspection.

Table of Contents

List of Figures

Acknowledgments

I would like to sincerely thank my supervisor Dr. T. Aaron Gulliver for overseeing my project work and providing guidance throughout my Master's degree. I would also like to thank Dr. Mihai Sima, for being on the supervisory committee. I would like to thank my friends for their support, especially Salahuddin Jokhio, for helping me with his knowledge and experience during this project. Foremost, I would like to express my deepest gratitude to my mother and siblings for encouraging me throughout my life, without their support and appreciation, this work would not have been possible.

Dedication

This work is dedicated to my Mother.

Chapter 1 Introduction

1.1 Internet Communication

A communication link between two or more computers forms a computer network, commonly known as a network. The communication medium can either be wired or wireless. The best-known computer network is the internet which consists of many different networks. The internet is a hub for global information exchange that uses the Transmission Control Protocol/Internet Protocol (TCP/IP) suite for communication. Communication on the internet is done by sending and receiving packets. The packets are sent from sources to destinations using network devices known as routers.

Packets consist of two components, the header and the payload. The header contains the routing information which helps the packet reach the destination, whereas the payload contains the data for the destination. Two Internet Protocol (IP) versions (IPv4 and IPv6) are used to communicate and have different size packet headers, but both contain the same information about the packet. The packet header contains information that distinguishes the packets such as the source address, destination address, protocol type, Time To Live (TTL), IP version and type of service. The packet payload is the actual data that is transmitted through the communication channel.

1.2 Internet Traffic Classification

Internet traffic analysis is of key interest to network designers, as this analysis leads to more efficient and fault-tolerant networks. Internet traffic can be analysed in different ways to perform various tasks such as classification and filtration. The objective of this project is to examine a new approach to Deep Packet Inspection (DPI) for web traffic analysis.

Internet traffic analysis is one of the most important tasks when it comes to controlling traffic access to a network. Internet traffic classification is beneficial to network administrators, Internet Service Providers (ISPs), and government agencies [1]. Administrators can use classification for identification and control of network applications. It can be applied in an Intrusion Detection System (IDS) for detecting Denial of Service (DoS) or other types of malicious attacks. ISPs can utilize traffic classification to monitor network flows and diagnose the network. Traffic classification can also be used to allocate bandwidth for applications, which is one aspect of DPI. Traffic classification can also help ISPs in network design and management. Traffic monitoring is useful for governments to carry out Lawful Inspection (LI) of packet payloads to obtain information about the users [1].

Internet traffic monitoring is a complex and challenging problem because network traffic is dynamic, real-time and random (since the next packet received might be completely unrelated to the one just received). Several techniques are used to carry out traffic detection and analysis. In this project, Snort is used to perform traffic detection and

analysis by means of DPI [2]. Traffic analysis using Snort requires the creation of Snort rules containing content of web applications that need to be detected.

1.3 Problem Statement

The internet is the hub for information exchange which transports trillions of bytes a day. Traffic classification is required by ISPs and organizations to filter and monitor internet traffic. Port based detection methods are not able to detect certain packets entering a network which should be filtered [1]. DPI techniques can perform this task effectively with better accuracy than port based methods. Usually, DPI is employed on routers by creating rules for the traffic to be detected, but routers are expensive and small organizations cannot afford them. Hence, there is a need for a system to be employed on a smaller scale to keep the cost and implementation complexity low. Snort can be used for this purpose. Snort is a network traffic detection tool which is primarily used in intrusion detection systems. Snort works by creating rules for detecting network traffic, and can perform all the tasks needed for DPI. Snort can be implemented on a single computer so small organizations can use it to avoid the extra cost of routing services.

1.4 Objectives

The objectives of this project are to configure and implement Snort as a DPI tool. This process involves web application detection by creating rules in Snort. A rule in Snort is a set of commands (containing content of the payload to be detected), which helps Snort

to detect network traffic. Snort is configured for detailed analysis of network traffic including bandwidth calculation of each application. By calculating bandwidth, ISPs can charge users for their internet services. The main goal of DPI is to create and implement policies for users like blocking web applications. This project will focus on DPI by using Snort for traffic detection and analysis.

1.5 Contributions

The major contribution of this project is the design of a low-cost and effective system for DPI. Usually, DPI is performed on routers which are expensive devices and small organizations cannot afford them. This work proposes Snort as an effective tool to perform DPI which can be implemented on a single computer as well as an entire network without using expensive networking devices like routers. Snort is configured on a single computer here as an example to perform DPI. Snort is configured as an inline Network Intrusion Detection System (NIDS) mode to block and track network traffic. In inline mode, a bridge network is created and all the web traffic passes through Snort and the desired policies are implemented there. Snort is further configured as an open-app-id (a mode in Snort to detect network traffic using different protocols like HTTP and HTTPS), to obtain the bandwidth of web applications. Furthermore, rules have been created to analyse and block web traffic.

Chapter 2 Traffic Analysis

Network measurements and monitoring provide important information to users, Internet Service Providers (ISPs), researchers and government agencies. This information can be used to ensure the quality, performance and functionality of computer networks, their services and applications. There are different methods to classify internet traffic. These methods are discussed below.

2.1 Traffic Classification Methods

ISPs always perform some sort of network filtration. For the filtration process, network traffic needs to be classified according to the type of traffic. Figure 2.1 is used as an example to explain the classification process. In this figure, payload based classification

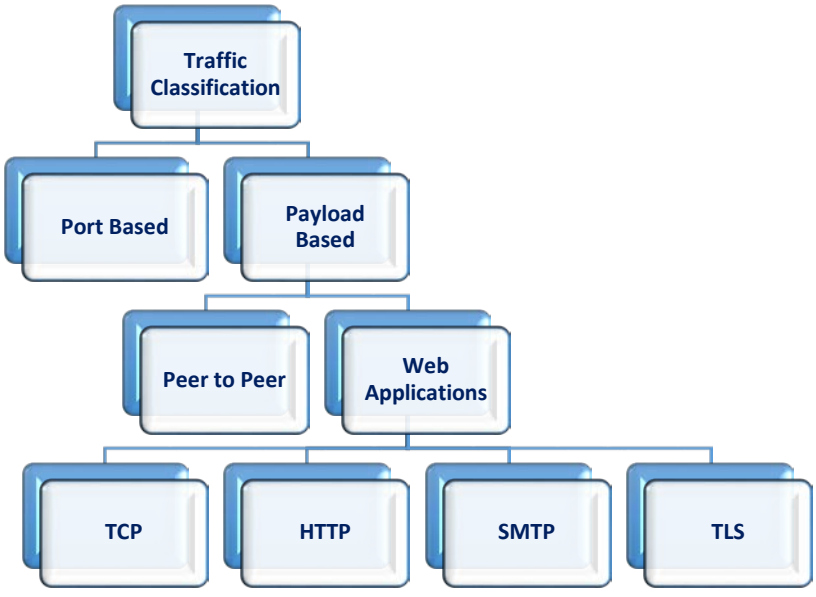

Figure 2.1: A hierarchical view of internet traffic classification.

is used to differentiate between peer to peer and web applications. Web applications can be further classified by the type of protocol like TCP, HTTP, SMTP and TLS. ISPs normally use Access Control Lists (ACLs) to allow or block wanted or unwanted Internet traffic. These techniques are discussed below.

2.2 Access Control Lists (ACLs)

ACLs are usually used on network monitoring devices which act as an entry point for traffic entering the network. The ACLs can be either port based or IP based. ACLs are sets of simple rules configured on networking devices.

2.2.1 Port number based packet classification using ACLs

Port number based ACLs are configured on monitoring devices so that the packets are classified based on port numbers used by well-known applications. The Internet Assigned Numbers Authority (IANA) is the body responsible for defining port numbers, for example Hyper Text Transfer Protocol (HTTP) uses port 80 and Simple Mail Transfer Protocol (SMTP) uses port 25. This classification is not suitable for analysis since Peer to Peer (P2P) traffic and intelligent applications can be disguised by using the port number of a well-known application. Hence, the majority of researchers believe that port-based classification cannot suffice for classification [1]. Another reason to avoid this technique is that several applications use dynamic port numbers.

2.2.2 Payload-based packet classification

Payload-based packet classification involves packet payload investigation also known as Deep Packet Inspection (DPI). The rule (signature) of the payload is used to classify packets which can yield accurate results. Rules are distinct strings in the packet payloads that can be used to differentiate them. Packets can be classified by matching rules to the contents of the payloads [2].

2.2.2.1 Deep Packet Inspection (DPI)

With the development of the internet, the accuracy of services identification by port number has declined. Moore and Papagiannaki stated that the accuracy of this method is at most 70 percent [11]. Thus, port number based services identification is not used independently in networks but plays the role of an additional inspection [11].

DPI is a means of filtering network traffic by monitoring a stream of packets and identifying strings of data. Based on information contained in the header and payload, we can distinguish the application of origin and even requests made by the application. The ability to sort traffic based on the web application has many practical uses [11]. In the process of inspecting traffic, the content of the payload is matched with the rules by deep inspection of the information at the application layer. The efficiency of DPI is dependent on the generation of rules which identify the network protocols and payloads.

Cell phone providers have smartphone application services which employ DPI, for example, My Rogers, which allow customers to access their account information.

Providers give the option to use this service on the network free of charge. Applications are detected by DPI and providers do not charge for the bandwidth that is consumed by that specific web application. Bell Mobility offers a number of hours of free viewing on their Bell TV application which is also done by DPI. In addition, many countries restrict access to particular web pages, for example, Facebook, YouTube and Twitter. For this to work effectively, all requests generated on the web page and its mobile applications must undergo a drop policy. Enforcement of this policy is accomplished with DPI by creating rules to drop or block the content of these websites.

Chapter 3 Snort

3.1 Snort

Snort is an open source Network-based Intrusion Detection System (NIDS) and Network-based Intrusion Prevention System (NIPS) which is used to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks [3]. It allows creating rules in order to detect malicious traffic and alert the user. Rules rely on distinct characteristics of network traffic in order to detect this traffic. These rules are used with Snort functions to perform protocol analysis, content searching, and content matching on network traffic [4]. The implementation of these rules on Snort is referred to as policies. Snort is able to raise alerts in real-time if a rule matches the content of a payload in the traffic. Snort has three operational modes as described below [7].

3.1.1 Sniffer mode

Sniffer mode only displays information about packet headers. Different commands can be used to display the headers of different protocols. For example, the *./snort –v* command will display TCP/IP packet headers [4].

3.1.2 Packet logger mode

Packet logger mode stores packets to the disk. This mode can be used to store the packet payloads in different directories based on the type of protocol or IP class. For

example, the *./snort –dev –l ./log* command will collect and place all packet payloads in the *log* directory [4].

3.1.3 Network Intrusion Detection System (NIDS) mode

NIDS mode can be used with stored packets as well as real-time traffic. In this case, network traffic is detected by implementing Snort rules on real-time traffic. This mode is configured using *snort.conf* (Snort configuration file) according to the output that is needed [3]. This mode can be further configured as an Intrusion Prevention System (IPS) to block network traffic by creating a bridge network. This mode is widely used for intrusion detection and prevention worldwide [4].

3.2 Snort Implementation

Snort is implemented such that all network traffic passes through it and policies are applied on this traffic. The policies are implemented by creating Snort rules which match the content that needs to be detected. If Snort is only used for NIDS than it is implemented within the network in a way such that a copy of all traffic is sent to Snort. This traffic is analysed by Snort and results are generated accordingly. This is typically only used for alerts and statistics of network traffic, and so is for intrusion detection not prevention. For IPS, Snort is implemented such that all the traffic passes through it before reaching the

destination. Figure 3.1 shows how Snort can be implemented in this mode. The Snort server is placed such that all traffic passes through it before reaching internal network.

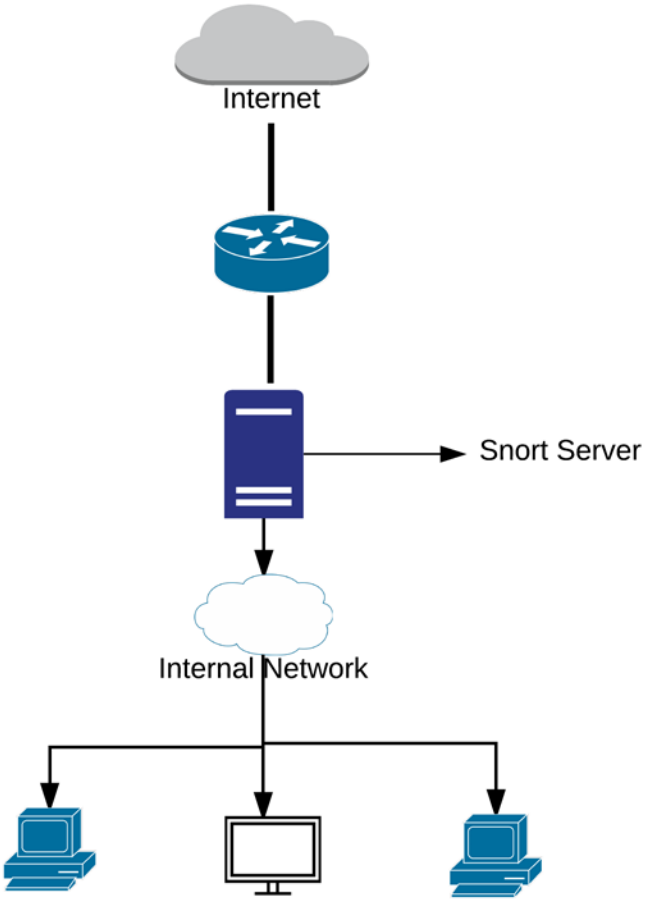

Figure 3.1: Implementation of Snort as an IPS.

The Snort server is the system on which Snort is installed and configured. In the implementation here, two Ethernet interfaces are created within Snort as a bridge network. All the traffic passes through Snort which acts as a first interface. It processes the network traffic according to the rules that are implemented and takes appropriate actions. If some traffic needs to be blocked, it drops that traffic and does not let it pass.

3.3 Snort Functionality

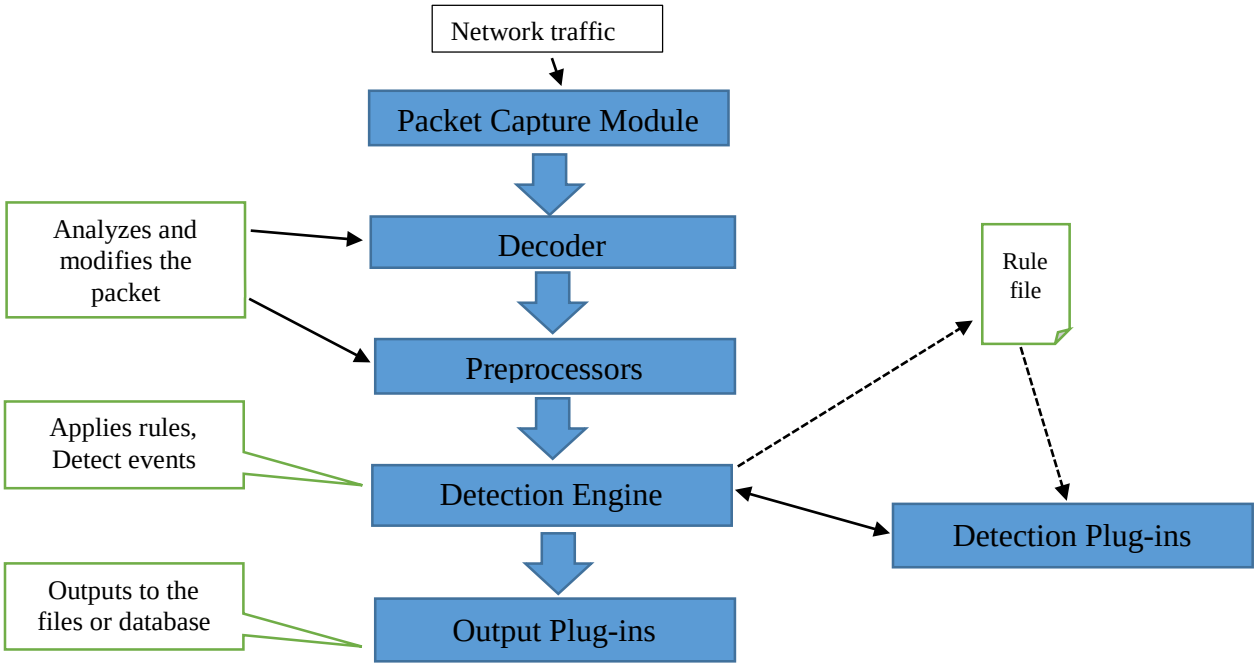

Figure 3.2: The functionality of Snort [10].

Figure 3.2 shows the basic functionality of Snort. The Packet Capture (PCAP) module is used for capturing incoming traffic packets. Next, the packet decoder identifies the protocol and classifies the packet according to the data-link specification of the packet. The preprocessors (e.g., SMTP preprocessor, POP preprocessor, FTP/TELNET preprocessor and session preprocessor), prepare the packets for processing by the detection engine. First, the preprocessors reassemble Internet Protocol (IP) fragments and Transmission Control Protocol (TCP) streams. Then, they check the packets for any traffic that is outside the scope of the detection engine [5]. The preprocessors can also

modify the packets so it will be easier for the detection engine to process the packets. For example, TCP anomalies such as data received outside a TCP window cannot be processed by the detection engine. The session preprocessor can detect this missing data and modify the packet to put it back in the TCP window so it can be detected [4]. The detection engine detects network traffic based on a set of defined rules [4]. Finally, reports are generated by the output plug-ins.

3.4 Snort Rules

A rule is implemented in Snort and policies are made according to this rule. Snort takes appropriate actions when the rule matches the internet traffic. Figure 3.3 shows the basic structure of a Snort rule which consist of a rule header and rule options.

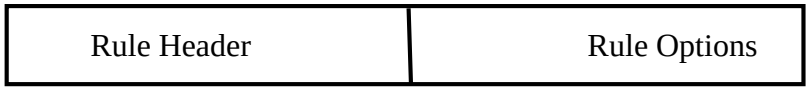

Figure 3.3: Snort rule structure [6].

Figure 3.4: Snort rule header [6].

Figure 3.4 shows the structure of a Snort rule header which consists of seven options. The Snort header provides the action to be taken with the address of the specified network

traffic. The rule is specified by the protocol, source address, source IP, direction of traffic flow, destination IP, and destination port [5].

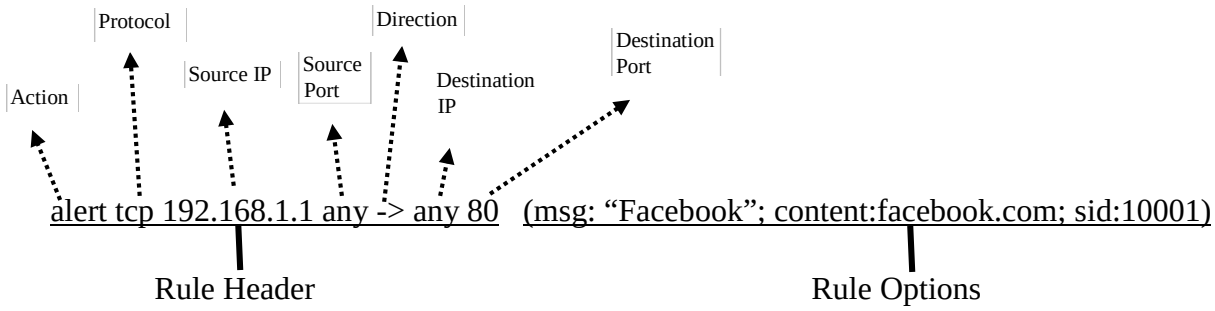

Figure 3.5: A sample Snort rule.

Figure 3.5 shows a sample Snort rule. The rule header contains information which can send an alert for a packet in a TCP stream whose source IP is 192.168.1.1 with any source port address towards any destination IP having port number 80 [8]. Rule options specify the rule name and rule id along with the content that needs to be detected. Snort takes the prescribed action when the rule header and rule options match the content of the packet [4].

Chapter 4 Methodology

In this project, Snort is used for DPI by configuring it in IPS mode. Rules are created to detect different web applications. These rules give alerts when matched with the traffic passing through Snort. For rule creation data capture, data filtering and data preparation must be done. This methodology is depicted in the diagram shown in Figure 4.1 and is discussed in detail below with an example.

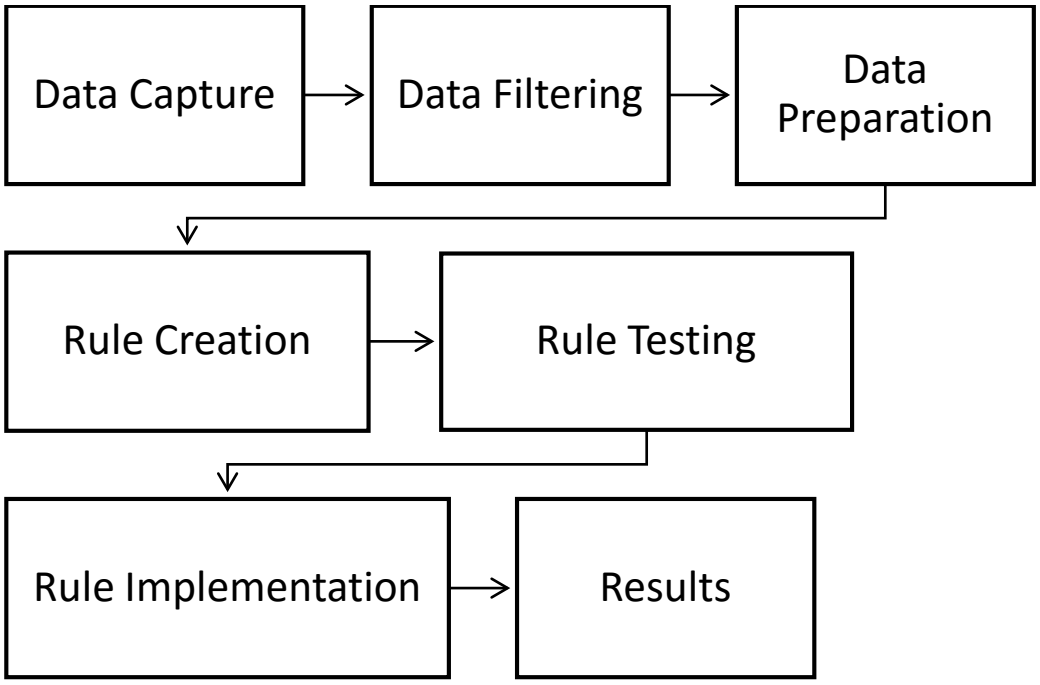

Figure 4.1: The methodology for DPI using Snort.

To create a rule for a web application, the first step is to get the content of the application which is achieved by capturing the web traffic of that application. In this report, Wireshark

is used for capturing traffic. Snort can also capture traffic but Wireshark is preferred because it is easy to read packets in Wireshark. Wireshark is an open-source program which displays packets both sent and received from a network interface in real-time. For example, Figure 4.2 shows the captured traffic of Facebook on *eth0* interface of the network. Wireshark provides details about each captured packet. After capturing the web traffic, it is filtered to remove unwanted traffic. Figure 4.3 shows a filter that is used to remove icmp, igmp, ntp, nbns. nbss and browser traffic. Many filters can be applied to get the desired results. The purpose of packet filtration is to get the traffic needed for rule creation. After filtering the raw traffic, the required traffic is in a PCAP file. Figure 4.4 shows a packet from a TCP stream which contains a Facebook server name. This content can be used to create a rule so that Snort will generate alerts when it find the same content in the network traffic. Figure 4.4 shows that this packet contains different server names, e.g. facebook.com, facebook.net, fb.com, fbcdn.net, and fbsbx.com. There can be a single server name or multiple server names in a packet. To get more accurate results in DPI, traffic can be captured on different platforms like IOS, Windows, Android and Linux. Accurate results can also be achieved by capturing different types of traffic like video, pictures and chat. Multiple rules can be created for the same web application.

In Figure 4.5, two rules have been created in Snort to detect Facebook traffic. The first rule finds “facebook.com”. By adding fast_pattern, Snort will search for this keyword first. In the second rule, there is PCRE (Pearl Compatible Regular Expressions) instead of content. The reason being that Snort is unable to process an OR statement, so PCRE is used to indicates that if any of these words are detected it is Facebook traffic. The next

step is to check the created rule by running it. If the rule has an error Snort will give an error message and not validate the rule. In Figure 4.6, such an error is shown where Snort does not validate the first rule as the word ‘side’ is not a rule option. Figure 4.7 shows that the error has been fixed as ‘side’ has been replaced with ‘sid’ which is the correct keyword for rule id. Snort will also detect any format errors in the rule. After validation, rules are tested on the PCAP files. In Figure 4.8, the rules in Figure 4.7 have been tested against the PCAP files and provide alerts on Facebook traffic which means that it is working and giving the desired results. Figure 4.8 shows the alerts for sid:10002 generated by Snort with source IP, source port, destination IP, destination port and alert time. After testing the rules, they can be implemented on real-time traffic to get alerts on Facebook traffic. This is the methodology for creating, testing and validating Snort rules.

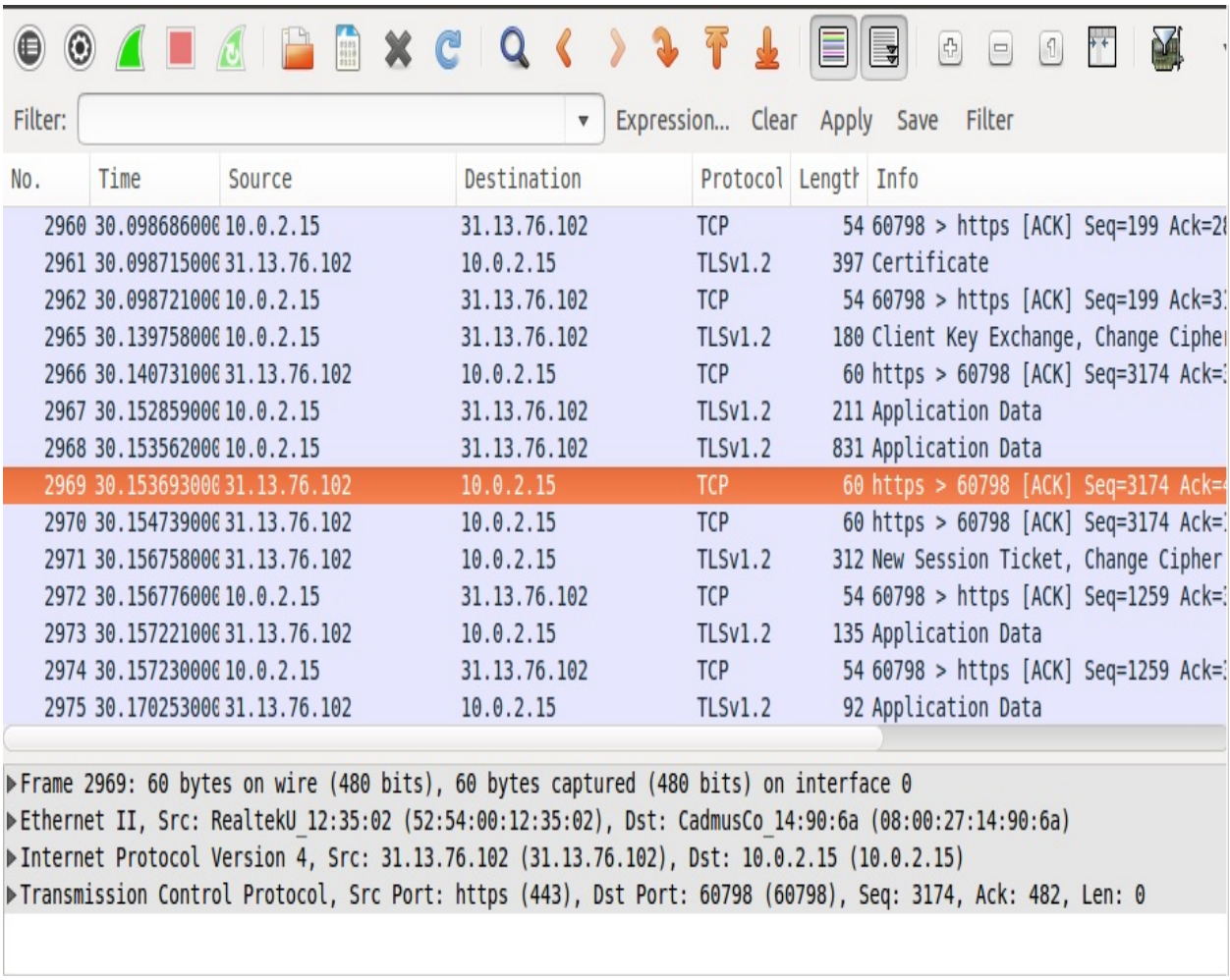

Figure 4.2: Facebook traffic captured using Wireshark.

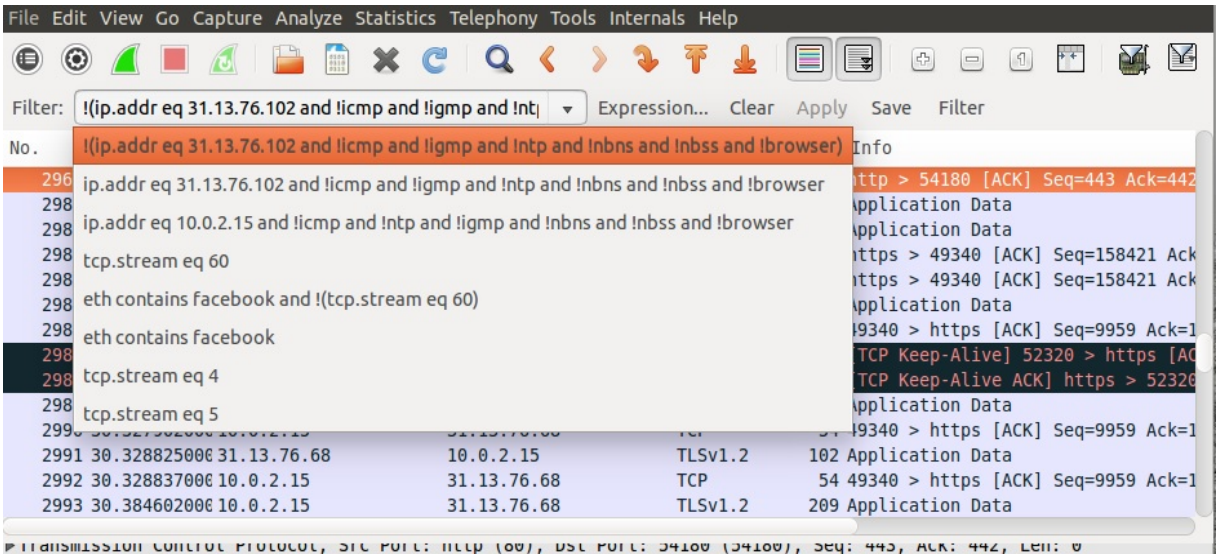

Figure 4.3: Filtration of a raw packet using Snort.

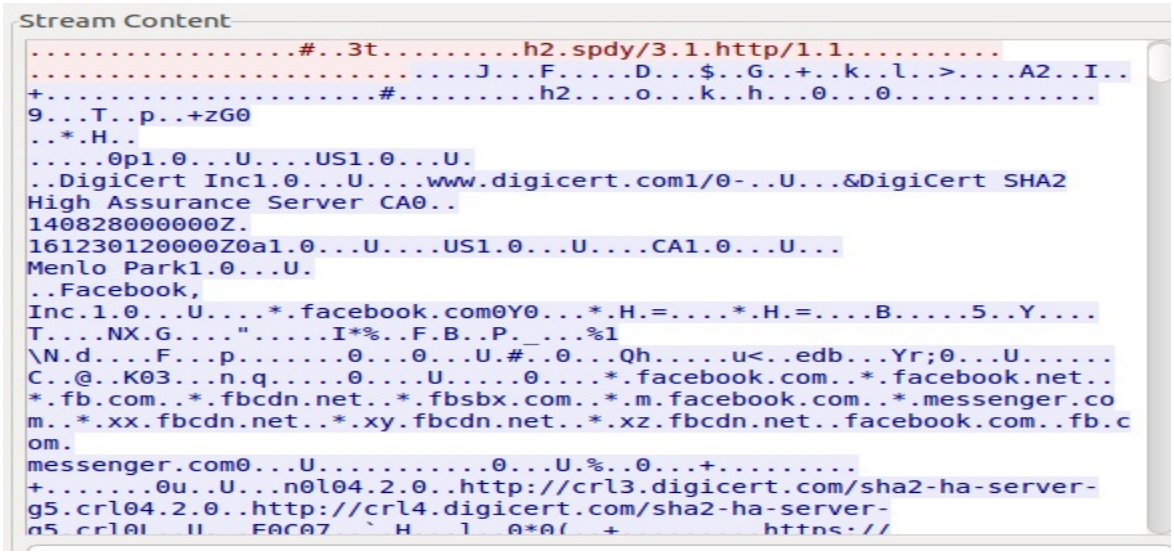

Figure 4.4: Payload of Facebook traffic used to create rule.

```
alert tcp any any -> any any (msg: "Facebook"; content: "facebook.com"; h
ttp_uri; fast_pattern; side:10001; rev:1;)

alert tcp any any -> any any (msg: "Facebook"; pcre:"/facebook.net|fb.com
|.fbcdn.net|.fbsbx.com/i"; sid:10002; rev:1;)
```

Figure 4.5: Rule to detect Facebook traffic.

```
++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
ERROR: /etc/snort/rules/local.rules(1) Unknown rule option: 'side'.
Fatal Error, Quitting..
saad@saad-VirtualBox:~$
```

Figure 4.6: Snort failed to verify the rule.

```
alert tcp any any -> any any (msg: "Facebook"; content: "facebook.com"; h
ttp_uri; fast_pattern; sid:10001; rev:1;)

alert tcp any any -> any any (msg: "Facebook"; pcre:"/facebook.net|fb.com
|.fbcdn.net|.fbsbx.com/i"; sid:10002; rev:1;)
```

Figure 4.7: Corrected Snort rule.

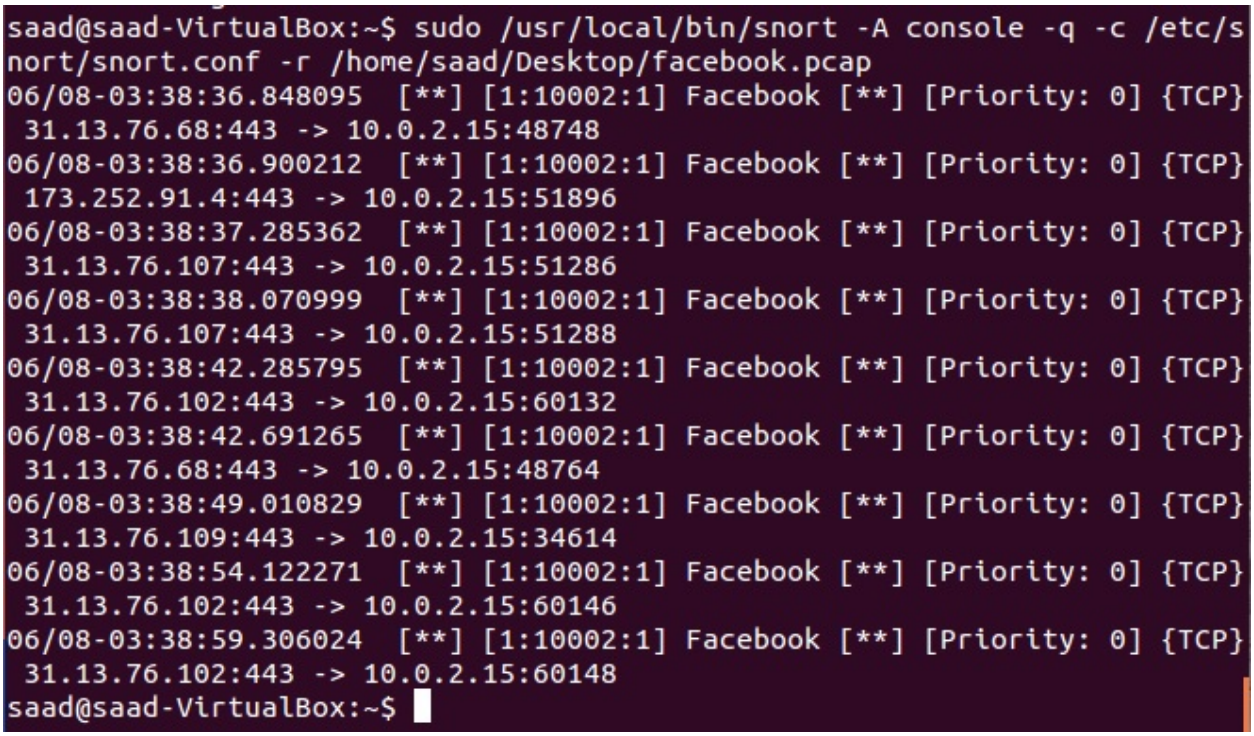

Figure 4.8: Rule alerts on Facebook.pcap in Snort.

Chapter 5 Results

Referring to Figure 3.1, the validated rules are implemented on the network interface to generate alerts on real-time traffic. Snort analyzes the ingress and egress of traffic on the network interface. Figure 5.1 shows the rules created for the detection of Facebook and Gmail traffic. Snort is configured on network interface *eth0*. After analysing Facebook and Gmail traffic, Snort generates the results. Figure 5.2 shows details of the traffic received, dropped, analysed and filtered. It also shows the number of packets Snort has processed as well as the running time of Snort and number of packets analysed. This shows that Snort ran for 1 minute and 55 seconds and analyzed 37447 packets per minute. The total number of received packets is 37794 and Snort analyzed 37448 of these packets which is 99.085%. The outstanding 346 packets were not analyzed by Snort because of errors in the packet structures. These packets can be analyzed manually.

Figure 5.3 provides a breakdown of the analysed traffic by protocol. This shows that 99.997% of the analysed packets are IPv4 packets and 0.003% are IPv6 packets. Further, 94.686% of the analysed traffic is TCP while 5.285% is UDP. Figure 5.4 shows the statistics of the alerts generated on the analysed traffic. It also shows the actions taken by Snort. Snort also generated a log file of processed traffic which can be used in open-app-id to calculate the bandwidth used by each web application. The bandwidth consumed by each web application is shown in Figure 5.5. Open-app-id contains some default rulesets to detect the bandwidth consumption of different types of the network traffic. It can be disabled but in this case it was enabled to detect the bandwidth consumed by each web application. Figure 5.5 shows the bandwidth consumption of Facebook and

Gmail traffic along with the other web applications detected by open-app-id. Figure 5.6 shows the alerts based on the rules generated by Snort. Obtaining statistics on each web application used is the main goal of DPI. In this example, rules were created to detect two web applications. In DPI, rules can be created and updated to detect each web application so that unknown traffic can be minimized.

In the next step, rules are created for security purposes to avoid a terrorism plot. Figure 5.7 shows the two rules created to detect content related to terrorism, e.g. make bomb, buy pistol, buy gun, and use gun. If anyone searches for this content on the internet, Snort will generate an alert with date, time, destination and source IP. These IPs can be further analysed by a security analyst to mitigate any future risks. Figure 5.8 shows the rule for “Terrorism Content Search” alerts and Gmail alerts. Figure 5.9 shows the rules created to block Facebook traffic. The basic difference is the word drop instead of alert in the Snort rule so it will drop the detected content. The results for the traffic blocked by Snort are shown in Figure 5.10. Nine alerts were generated by Snort and Snort blocked all 3305 related packets.

```
# Signatures to detect application
#Facebook Signatures
alert tcp any any -> any any (msg: "Facebook"; content: "facebook.com"; http_uri; fast_pattern; sid:10001; rev:1;)
alert tcp any any -> any any (msg: "Facebook"; pcre:"/.facebook.com|.facebook.net|.fb.com|.fbcdn.net|.fbsbx.com|.fb.me/i"; sid:10002; rev:1;)
#Gmail Signatures
alert tcp any any -> any any (msg: "Gmail"; content: "mail.google.com"; http_uri; fast_pattern; sid:10011; rev:1;)
alert tcp any any -> any any (msg: "Gmail"; pcre:"/mail.google.com|.Gmail.com/i"; sid:10012; rev:1;)
```

Figure 5.1: Rules to detect Facebook and Gmail traffic.

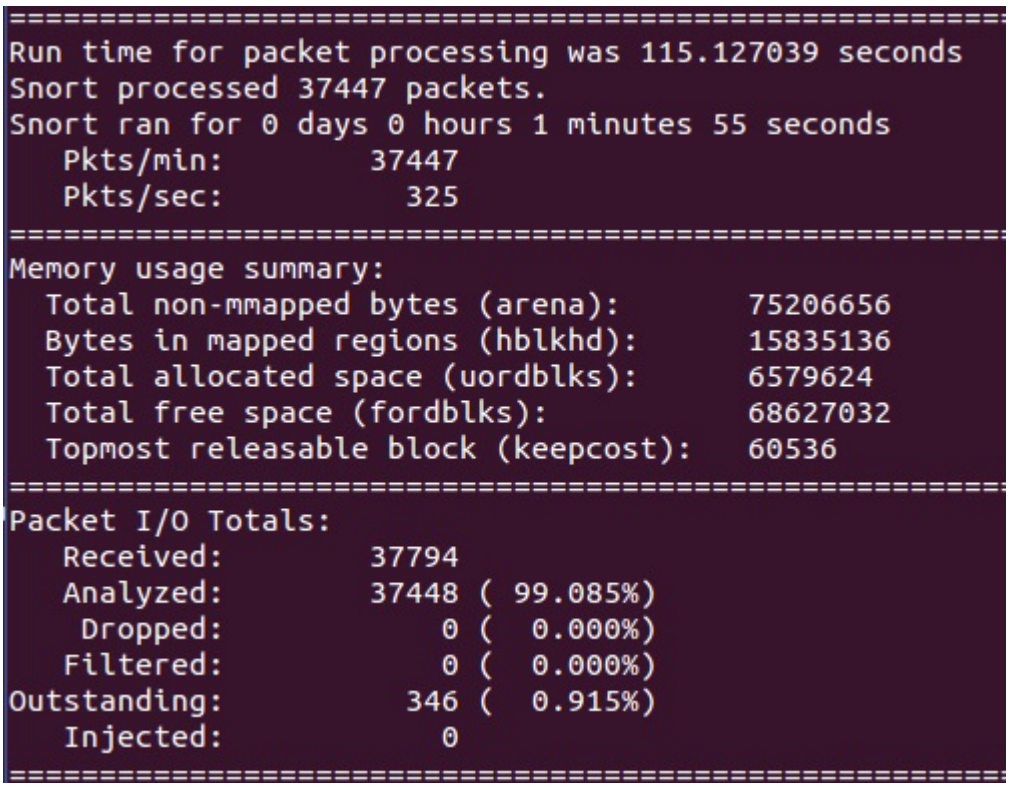

Figure 5.2: Packets processed by Snort.

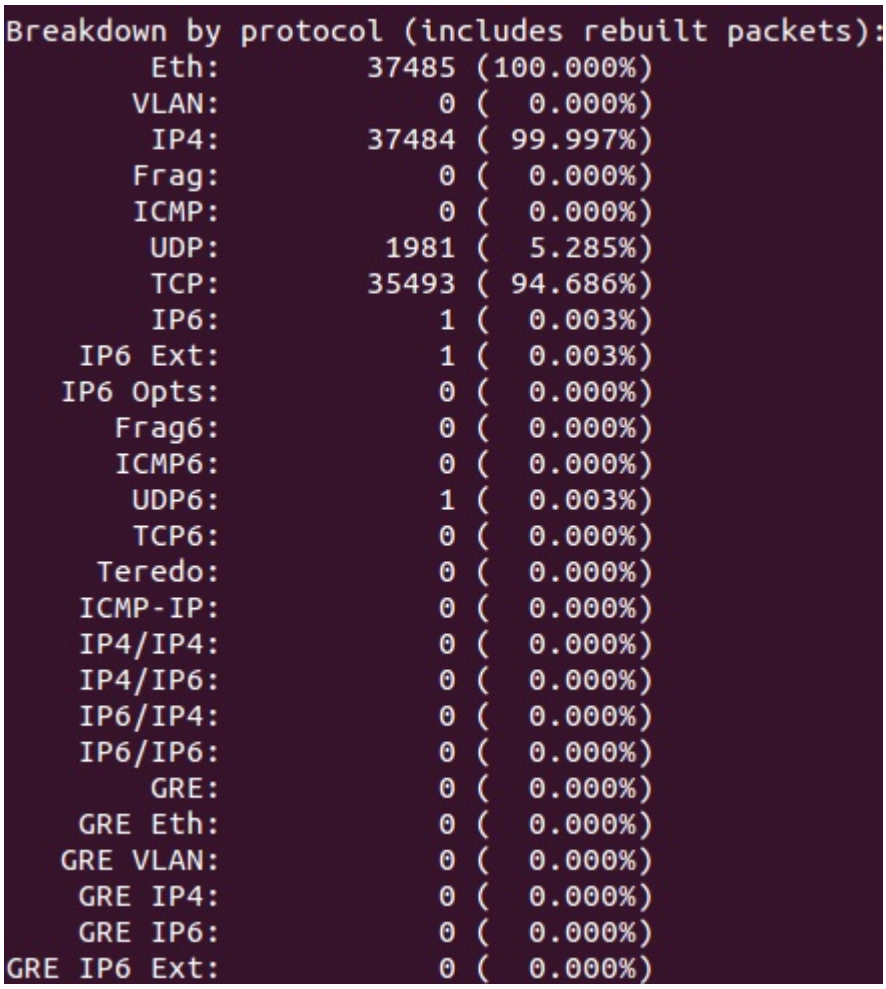

Figure 5.3: Classification of network traffic by protocols.

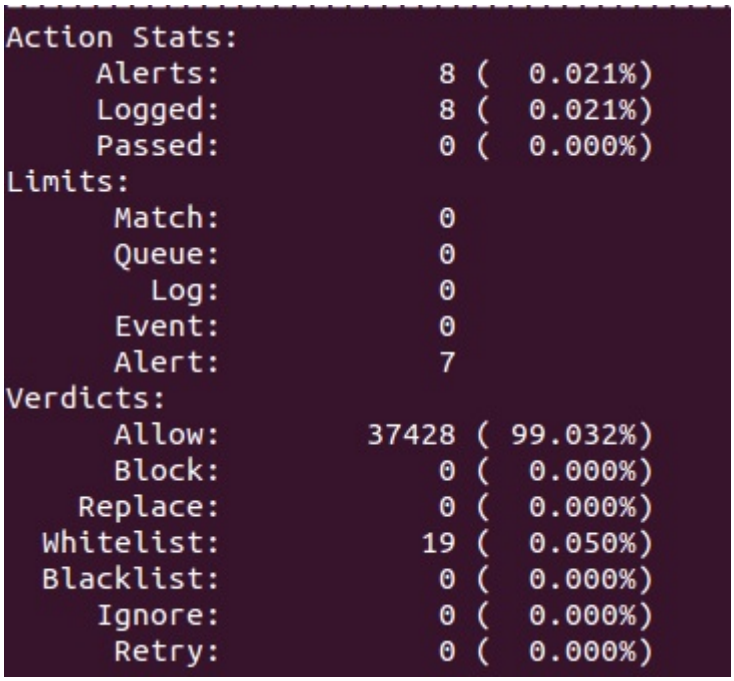

Figure 5.4: Statistics of the actions taken by Snort.

```
statTime="1469228580",appName="Google",txBytes="0",rxBytes="21265"
statTime="1469228580",appName="Wordpress",txBytes="0",rxBytes="7431"
statTime="1469228580",appName="HTTPS",txBytes="0",rxBytes="28696"
statTime="1469228580",appName="SSL client",txBytes="0",rxBytes="28696"
statTime="1469228580",appName="__unknown",txBytes="0",rxBytes="2886"
statTime="1469228580",appName="HTTPS",txBytes="0",rxBytes="43999"
statTime="1469228580",appName="Mozilla",txBytes="0",rxBytes="37326"
statTime="1469228580",appName="SSL client",txBytes="0",rxBytes="37326"
statTime="1469228580",appName="__unknown",txBytes="0",rxBytes="402148"
statTime="1469228640",appName="Google",txBytes="0",rxBytes="10558"
statTime="1469228640",appName="HTTPS",txBytes="0",rxBytes="10558"
statTime="1469228640",appName="SSL client",txBytes="0",rxBytes="10558"
statTime="1469228640",appName="__unknown",txBytes="0",rxBytes="266738"
statTime="1469228580",appName="Google",txBytes="0",rxBytes="3803680"
statTime="1469228580",appName="Wordpress",txBytes="54",rxBytes="6723870"
statTime="1469228580",appName="DNS",txBytes="26669",rxBytes="0"
statTime="1469228580",appName="Facebook",txBytes="270",rxBytes="2925212"
statTime="1469228580",appName="HTTPS",txBytes="432",rxBytes="13498151"
statTime="1469228580",appName="Mozilla",txBytes="108",rxBytes="4028"
statTime="1469228580",appName="SSL client",txBytes="432",rxBytes="13476509"
statTime="1469228580",appName="Doubleclick",txBytes="0",rxBytes="6293"
statTime="1469228580",appName="Atlas Advertiser Suite",txBytes="0",rxBytes="5288"
statTime="1469228580",appName="LiveRail",txBytes="0",rxBytes="8138"
statTime="1469228580",appName="Gmail",txBytes="702",rxBytes="5952398"
statTime="1469228640",appName="Google",txBytes="0",rxBytes="187999"
statTime="1469228640",appName="DNS",txBytes="98152",rxBytes="0"
statTime="1469228640",appName="HTTPS",txBytes="0",rxBytes="207084"
statTime="1469228640",appName="SSL client",txBytes="0",rxBytes="187999"
```

Figure 5.5: Bandwidth calculation of web applications.

```
07/22-17:04:22.949993  [**] [1:10002:1] Facebook [**] [Priority: 0] {TCP} 69.171.230.68:443 -> 10.0.2.15:56506
07/22-17:04:37.396015  [**] [1:10002:1] Facebook [**] [Priority: 0] {TCP} 31.13.76.102:443 -> 10.0.2.15:38152
07/22-17:04:48.170219  [**] [1:10012:1] Gmail [**] [Priority: 0] {TCP} 74.125.28.189:443 -> 10.0.2.15:58364
07/22-17:06:33.216254  [**] [1:10002:1] Facebook [**] [Priority: 0] {TCP} 31.13.76.102:443 -> 10.0.2.15:38178
07/22-17:06:38.778191  [**] [1:10002:1] Facebook [**] [Priority: 0] {TCP} 31.13.76.102:443 -> 10.0.2.15:38180
07/22-17:07:38.755500  [**] [1:10012:1] Gmail [**] [Priority: 0] {TCP} 74.125.28.189:443 -> 10.0.2.15:58396
07/22-17:08:15.457901  [**] [1:10012:1] Gmail [**] [Priority: 0] {TCP} 74.125.28.189:443 -> 10.0.2.15:58402
07/22-17:09:00.658354  [**] [1:10012:1] Gmail [**] [Priority: 0] {TCP} 74.125.28.189:443 -> 10.0.2.15:58408
```

Figure 5.6: Rules alerts on real-time traffic.

```
# Signatures to alert on content search for security reasons
alert tcp any any -> any any (msg: "Terrorism Content Search"; content: "terrorism"; sid:10021; rev:1;)
alert tcp any any -> any any (msg: "Terrorism Content Search"; pcre:"/make bomb|buy gun|buy pistol|use gun/i"; sid:10022; rev:1;)
```

Figure 5.7: Rules to detect terrorism content.

```
saad@saad-VirtualBox:~$ sudo /usr/local/bin/snort -A console -q -c /etc/snort/snort.conf -i eth0
07/22-16:33:02.600637  [**] [1:10012:1] Gmail [**] [Priority: 0] {TCP} 216.58.193.101:443 -> 10.0.2.15:55738
07/22-16:33:25.905642  [**] [1:10021:1] Terrorism Content Search [**] [Priority: 0] {TCP} 199.102.160.7:80 -> 10.0.2.15:52062
07/22-16:33:25.908641  [**] [1:10021:1] Terrorism Content Search [**] [Priority: 0] {TCP} 199.102.160.7:80 -> 10.0.2.15:52062
07/22-16:33:25.908704  [**] [1:10021:1] Terrorism Content Search [**] [Priority: 0] {TCP} 199.102.160.7:80 -> 10.0.2.15:52062
07/22-16:33:25.908736  [**] [1:10021:1] Terrorism Content Search [**] [Priority: 0] {TCP} 199.102.160.7:80 -> 10.0.2.15:52062
07/22-16:33:25.908765  [**] [1:10021:1] Terrorism Content Search [**] [Priority: 0] {TCP} 199.102.160.7:80 -> 10.0.2.15:52062
07/22-16:33:25.909622  [**] [1:10021:1] Terrorism Content Search [**] [Priority: 0] {TCP} 199.102.160.7:80 -> 10.0.2.15:52062
07/22-16:33:25.909665  [**] [1:10021:1] Terrorism Content Search [**] [Priority: 0] {TCP} 199.102.160.7:80 -> 10.0.2.15:52062
07/22-16:33:25.909695  [**] [1:10021:1] Terrorism Content Search [**] [Priority: 0] {TCP} 199.102.160.7:80 -> 10.0.2.15:52062
^C*** Caught Int-Signal
saad@saad-VirtualBox:~$
```

Figure 5.8: Alerts generated by Snort on "Terrorism Content Search".

```
# Signatures to drop or block internet traffic
#Facebook drop
drop tcp any any -> any any (msg: "Facebook"; content: "facebook.com"; http_uri; fast_pattern; sid:11001; rev:1;)
drop tcp any any -> any any (msg: "Facebook"; pcre:"/.facebook.com|.facebook.net|.fb.com|.fbcdn.net|.fbsbx.com|.fb.me/i"; sid:11002; rev:1;)
```

Figure 5.9: Rules to block Facebook traffic.

```
======
Action Stats:
     Alerts:             9 (   0.272%)
     Logged:             9 (   0.272%)
     Passed:             0 (   0.000%)
Limits:
      Match:             0
      Queue:             0
        Log:             0
      Event:             0
      Alert:             9
Verdicts:
      Allow:             0 (   0.000%)
      Block:          3305 (100.000%)
    Replace:             0 (   0.000%)
  Whitelist:             0 (   0.000%)
  Blacklist:             0 (   0.000%)
     Ignore:             0 (   0.000%)
      Retry:             0 (   0.000%)
===============================================
```

Figure 5.10: Traffic blocked by Snort.

Chapter 6 Conclusion and Future Work

Snort is an effective tool which can be used for deep packet inspection. All network traffic can be monitored by creating rules to detect web applications. Snort can also be used to detect content for security purposes and block web applications. Traffic analysis can be done using Snort. In this work, Snort was used on a single system but it can also be implemented on a network.

Snort is widely used as an intrusion detection system tool. In this work, Snort was used as a DPI tool. Using Snort will help reduce the need for expensive devices like routers. In addition, fewer resources are needed by using a single tool for multiple purposes. Snort allows for both deep packet inspection and intrusion prevention on a smaller scale.

Snort can be used as NIDS and DPI tools simultaneously. This work can be extended to develop a platform where both NIDS and DPI can be achieved together using Snort.

References

[1] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials,* vol. 10, no. 4, pp. 56-78, 2008.

[2] J. Erman, A. Mahanti, and M. Arlitt, "Internet traffic identification using machine learning," Proceedings of *IEEE Globecom*, San Francisco, CA, pp. 1-6, 2006.

[3] Snort Frequently Asked Questions, [Online]. Available: https://www.Snort.org/faq. [Accessed 04 07 2016].

[4] Snort Users Manual, vol. 2.9.8.2, The Snort project, 2016.

[5] K. Salah and A. Kahtani, "Improving Snort performance under Linux," *IET Communications,* vol. 3, no. 12, pp. 1883-1895, 2009.

[6] N. Khamphakdee, N. Benjamas, and S. Saiyod, "Improving intrusion detection system based on Snort rules for network probe attack detection," Proceedings of the *International Conference on Information and Communication Technology*, Bandung, Indonesia, pp. 69-74, 2014.

[7] R. Padamashani, S. Sathyadevan, and D. Dath, "BSnort IPS better Snort intrusion detection / prevention system," Proceedings of the *International Conference on Intelligent Systems Design and Applications*, Kochi, India, pp. 46-51, 2012.

[8] E. Azimi, M. B. Ghaznavi-Ghoushchi, and A. M. Rahmani, "Implementation of simple Snort processor for efficient intrusion detection systems," Proceedings of the *IEEE International Conference on Intelligent Computing and Intelligent Systems*, Shanghai, China, pp. 533-537, 2009.

[9] J. Xi, "A design and implement of IPS based on Snort," Proceedings of the *International Conference on Computational Intelligence and Security*, Hainan, China, pp. 771-773, 2011.

[10] S. Niccolini, R. G. Garroppo, S. Giordano, G. Risi, and S. Ventura, "SIP intrusion detection and prevention: Recommendations and prototype implementation," Proceedings of the *IEEE Workshop on VoIP Management and Security*, Vancouver, BC, pp. 45-50, 2006.

[11] Y. Lan, Y. Tang and S. Zou, "Research and implementation of mobile internet services identification system," Proceedings of the *International Conference on Computer Science & Education*, Cambridge, UK, pp. 401-405, 2015.