Firewall Rule Set Optimization

Submitted in partial fulfilment
for the requirements of the degree of

Bachelor of Science (Honours) in Computer Science

at Rhodes University

Innocent Makungo Mukupa

Grahamstown, South Africa
November 8, 2010

Abstract

A firewall is an intermediate system placed between two networks of different trust levels, mostly between the secure corporate network and the less secure public Internet. It is intended to keep the corporate network safe from intruders, hackers and other malicious entry into the corporate network. A firewall performs packet filtering by applying rules defined in a rule set that is configured on the firewall. This packet inspection against the rules is done sequentially to find a rules that match the packet for the firewall to apply an action of either *allow* or *deny.* Most firewalls exit the inspection process once a match is found, but the fact that the inspection is a sequential process negatively impacts on filtering performance.It is evident that not all rules in a given rule set on a firewall match a packet inspected.Therefore evaluating all the rules until a match is found introduces performance degradation.Network transmission speeds have increased tremendously.This increase coupled with increased complexity of the attacks, requires better packet filtering algorithms to maximize throughput and the benefit of modern firewall hardware devices. Recent works in this area have investigated ways of improving firewall performance based on different criteria.This research investigated ways of combining traffic characteristics with rule matching patterns to reduce the number of rules in a rule set. The researcher came up with approaches and used them to come up with an optimal rule set of less rules that reduces the inspection time.

ACM Computing Classification System Classification

Thesis classification under the ACM Computing Classification System (1998 version, valid through 2010):

C.2.0 [General]: Security and protection

C.4 [Performance of Systems]: Measurement techniques

D.0 [General]

General Terms: Security, Performance

Acknowledgements

To the most high God, Jah , king of kings and maker of heaven and earth who gave me life and blessed me with so much, praise be to thee!

I am very grateful to my supervisor, Mr Barry Irwin, whose involvement in this work and experience in the field of Information Security I have been privileged to learn a lot from, Barry thank you very much for your guidance. Thank you to the members of the Security and Networks Research Group (SNRG) at Rhodes University for their sharing and teaching spirit.

I am grateful to my parents, Mr Katandula W Mukupa and Mrs Elizabeth Chewe Mukupa for their love and support. Mum your teachings and instructions have shaped me into this man I am today. To my father, who always told me to work hard towards what I believed in and seek excellence, thank you very much. Thanks to all my brothers and sisters who have been so supportive to me through out. I love you all.

I am thanking my brother Prudential Chongo Mukupa who never hesitated to put my life and growth in his hands. Your lessons, love and financial commitment to my progress and pursuance of my independence has been heavenly.

It takes a certain type of man or woman to teach another something that will make them better. Living in this world is a challenge in itself and I have been able to grow and breakthrough those challenges thanks to the teachings of men and women labouring for the advancement of God's kingdom; Bishop Roy Wallace and other pastors of the Gospel Outreach Church in Hochland Park, Windhoek, Namibia. My humble thanks to Rev. Helmut Reutter, Pastors Jeff Musonda, Dickson Phiri, Mbinda Wina and other ministers of God's word at the Gospel Outreach Fellowship Centre in Lusaka Zambia for their life equipping gospel teachings that have taught me a lot and shaped me this far.

I wish to extend my thanks to River of Life Church in Grahamstown and Pastors John and Debbie Sloane in particular for their encouragement and spiritual feeding without which my life in Grahamstown would have been empty.

What would one be without friends? I wish to thank the Nawas, Elleanor and Watae for the encouragement and care showed to me during my study period. Mwape Mo

Chileshe, you keep a positive heart going when you crack jokes in moments of reflection. My friends Ian Namulya Mung'omba, Malama Chishala, Sydney Shibeene, Chilufya Ndalameta, Benard Kayinga, Alfred Banda, Jasmine Orlam, Biko Adams, Alton Zirimah, K.T.L Pohamba, Miss Nuule Vasti, Nimrod Njamba, Lahya, Vido C Vicentius and all of you guys that I have become family with, thank you for your prayers, good wishes and help that made the load of life easy for me to carry.

To Mpande Chizhyuka and Mabvuto Mumba, your running around for me made it possible to get to Rhodes. You cared and settled me well in Grahamstown. Your reward is from God you served by doing that for one of his children. Thank you so much for being there for me.

I wish to acknowledge the support of the following cooperating partners of the department of computer science at Rhodes University; Telkom SA, Comverse SA, Stortech, Tellabs, Easttel, Bright Ideas Projects 39 and THRIP through the Telkom Centre of Excellence at Rhodes University. Their material provision made this work possible and as such I am highly indebted to them for providing this learning environment in which I have been able to utilize my potential as a student.

Contents

List of Figures

Chapter 1

Introduction

A firewall is a logical security component deployed between networks of different trust levels. This is mostly between the public Internet of no trust and the internal network for a given business. Firewalls are also deployed within private trusted networks to segment networks and control resource sharing internally [34]. A specific case in point would be using a firewall to control subnets withing an organisation to avoid data contamination, privilege elevation, and access to certain data [15]. The primary function of a firewall is to block unauthorised traffic while permitting authorised traffic going in either direction.

There are various types of firewalls deployed depending on the role required in the network. There is increased need for packet filtering with the growth of the Internet and the proliferation of Internet based services. This has extended the types to firewall categories based on functionality such as *web application firewalls, proxy firewalls, host based firewalls, circuit level, dynamic and hybrid firewalls* [7]. Hybrid types are a combination of two or more implementations to improve on functionality.

A firewall performs packet filtering by applying a set of rules to a packet sequentially until a rule matching the packet is found in the rule set. Firewalls that perform deep packet inspection like *application firewalls*, go on inspecting even after the first match is found [27]. Firewall rule-set can have configuration flaws; at times different administrators write rules ending up with a set of generic rules defined that match packets that other rules are also set to match [7, 33]. These configuration changes due to adjustments in the security policy, do make it difficult to know which requests for important network services are allowed to pass through the firewall because of large rule sets [34]. Therefore, the need for rule set optimization is a critical one.

Finding ways of reducing this inspection time to increase a firewall's throughput has been investigated and continues to be given attention by the research community. Packet filtering as an aspect of network computing security has generally been described as an NP-Hard problem [19]. This is because of the many challenges posed by protocol complexities, filtering choices, addressing challenges like IP spoofing.etc [7]. There are other challenges like rule dependencies which are configuration based and pose dangers of opening the system to intended illegal traffic or locking out the system from needed services and traffic.

1.1 Problem Statement

It is evident that not all rules in a firewall rule set match every packet inspected. It has also been widely observed that network traffic is never static and its dynamic nature requires continued adaptation of rules in a rule-set to match up to traffic demands. Rule-sets grow to large numbers in depth and maybe written by different network administrators. This gives rise to rule over-shadowing problems among other inconsistencies in the firewall configuration.

Firewalls perform filtering by inspecting a packet against a set of rules written from the security policy. This inspection is sequentially done for both inbound and outbound packets. Rule-sets grow to long depths in terms of the number of rules in the rule-set depending on the security requirements of the network. Sequential packet inspection against the full rule-set until a match is found is resource intensive and negatively impacts on the filtering capability because unnecessary checking is done on rules that will not match the packet.

1.2 Research Goals

This work was motivated by the evidence that not all rules in a rule-set match a packet and also that traffic is never static. The increase in network speeds poses increased challenges on filtering performance of a firewall. There has been improvements in protocols designed for IP networks, also transport techniques keep improving and changing. The objectives of this research therefore were:

- to investigate firewall filtering capacity improvement from reducing the rule set size.
- to develop a tool that will aid network administrators in optimizing rule sets.

1.3 Thesis Layout

The remainder of this document is arranged as follows:

Chapter 2 : Discusses network theory as it relates to firewalling. A discussion on security, packets and protocols and their characteristics that help and complicate firewall performance is presented here. An overview of firewalls; what they are, how they filter traffic, types of firewalls and implementation technologies employed are given in this section. Then a detailed review of previous works done with regard to rule-set optimization focusing on the approaches and traffic aspects considered is given.This is to help build a good background of firewalls,rule sets and how packets interact with the firewall. A look and analysis of the techniques used in previous works and tools developed or proposed for rule set optimization is detailed here. This section ends by looking at security policies; how they are used to develop rule-sets and some complications experienced in packet filtering.

Chapter 3 : The researcher here discusses the project methodology used to conduct the research. This details the experimental setup; the firewall package used, the rule-sets creation and rule-set types used to perform the filtering. A way of manipulating rules and rule-sets; detailing rule syntax, numbering and use of ports in filtering is given in this section.This researcher explains in detail, the project environment from the operating systems used, tools, network configuration to traffic movement and how optimization was performed. Methods used to collect traffic statistics are given here and explained in the optimization basis section to give more insight into how packet inspection works.

Chapter 4 : Presents the design of the rule-set optimizer. This details the design considerations and implementation approach. This researcher explains the problems encountered with designing and implementing the tool. Finally, proposed extensions that can be done to proceed with the tool design and implementation in future are given at the close of this section.

Chapter 5 : Explains the tests done to measure performance comparisons on the default and optimized rule-set.The section starts by explaining the idea behind performing the tests done and then goes into the setup parameters used to conduct the tests. The metrics measured to determine performance differences are given, and an in-depth look at how these metrics relate to firewall throughput is presented here.

Chapter 6: This chapter presents the results of the measured metrics. These are based on the default or naive rule-set and the optimized rule-set. Analysis of the results and observed variations are discussed in this chapter. The review of the results is in relation to the hypothesis presented in the set out objective.

Chapter 7 Concludes the work done in this research and sums up the research findings with respect to the goals. It outlines the ways used in approaching the problem statement ending with a general overview of why rule-set optimization is critical considering the dynamic nature of traffic.

Chapter 2

Related Work

This chapter covers some previous work done in the field of firewalls with regard to rule-set optimization. A look at various novel works and approaches to limiting rule set size was done as reported here. Also the current and previous works on optimization tools are reviewed and their approaches analyzed and presented here. An overview of network principles and security in general as it relates to the research is presented to start the section. This consolidates the knowledge of firewalls and their place in the network.

2.1 Network Overview

The idea of creating computer networks is to share resources, exchange data, share hardware, collaborate on tasks, and distribute services and applications over geographical locations or within buildings. The devices on the network are referred to as nodes, analogous to knots in fishing nets. These nodes can be connected by various types of media.Any node on a network is identified by a an Internet protocol,IP, address [3]. The IP address is a numeric value that can also be represented in text form.

These networks are built in different arrangements called topologies ranging from bus (all nodes are connected along a single cable), star (all nodes connect to a central node), ring topology and mesh. These topologies are used to implement networks such as Local Area Networks (LAN), Wide Area Networks (WAN) and Metropolitan Area Networks (MAN) [39].

2.2 Security

Security in computing environments involves the daily tasks of making sure the services a network offers are available always. This requires that the infrastructure that hosts these services the network provides is available and secure all the time. These are tasks like power management, User management, Backups, Privilege management, IP Accounting among others [7]. This security is divided into Physical and Logical security.

- *Physical Security*: the different ways of restricting access to the network environment and resources physically by lock and key, security guards, dogs, closed circuit television, CCTV, and other methods used to stop physical access.
- *Logical security*: implementing logical ways of restricting resource use and manipulation using Intrusion Detection Systems, IDS, Firewalls, biometric identification, Authorization Authentication and Accounting, (AAA) and access control through limiting privileges among others [36]. Authorization determines whether a given entity is allowed to perform a given activity.Authentication checks if the device or user requesting a resource is who it is by providing corresponding credentials such as a password [42]. Accounting keeps track of the use of network resources by users. In firewall systems, a user can be a process or service trying to access an address that it is not allowed to. In this case AAA helped keep logs for accounting, check the requesting host address for authentication and authorization.

2.3 Packets and Protocols

The underlying objects firewalls deal with are packets for different protocols. For network transport to send any information across it has to be broken down into small pieces, each of which is sent separately.These pieces of data are called Packets and all data transfer across IP networks is done in the form of packets. The breaking down of these pieces allows many systems to share the network, each sending the packets in turn [3]. This network sharing is critical in packet filtering, it requires firewall configuration to ensure legitimate packets are not lost by denying them entry.

Packets are built in a layered structure, with each layer representing a protocol used for a particular connection wrapped around the packet. A packet has two parts: body and header [7]. The header contains protocol information relevant to that layer and the body carries the data for the protocol. This wrapping around of header information continues down the TCP/IP stack with each layer preserving the header and body of the previous layer in what is termed as encapsulation [42].

This packetization of data with each layer having the headers preserved is what makes filtering possible. It allows packets to be inspected by protocol type, address or service they are destined for. Ethernet packets for example, consists of the Ethernet header and Ethernet body. Ethernet address is also known as the Media Access Control (MAC) address [34]. You will generally not perform packet filtering based on information in the Ethernet header though it gives the following information:

- What kind of packet this is: packet name by transport protocol; Appletalk, Novell or some other kind. In this research it is an IP packet
- Ethernet address of the source node that put the packet onto this particular Ethernet network segment: this is the address of the machine the packet originated from if it is attached to this segment; if not, it is the address of the last router in the path from the source machine to the destination.
- Ethernet address of the packet's destination on the given Ethernet network segment: this could be the address of the destination machine if it is on this segment; or the next router in the path from here to the destination machine.

Ethernet packets encapsulate IP packets as their data body [13]. This is passed up the stack and at the network layer; the IP packet is opened to see the addresses. This research in rule set optimization focused on network layer filtering considering IP packets and the services associated with that. This is because at the IP layer, the IP Packet header contains interesting information pertaining to packet filtering:

- IP source address
- IP destination address
- IP protocol type
- IP options field

These values above are used in conjunction with Transmission Control Protocol information that is used to pair with Internet Protocol values. For communication to occur for example, a port on the source IP host must connect to a port on the destination host [38]. As such the following TCP header values were identified as being critical to packet filtering.

- TCP source port: this number specifies what client or process a packet is originating from on the source host.
- TCP destination port: this specifies the client or process a packet is going to on a destination host.
- TCP flags field: contains various flags that are used to show special kinds of packets especially during the process of setting up and tearing down TCP connection.

A general understanding about networking has been presented here to put into perspective, firewall decisions when filtering traffic. Rules are often explicitly written to block or allow traffic from specific IP addresses and network numbers. Also if forwarding is configured on the firewall, IP addresses are used to identify recipient nodes that might perform further packet inspection.

2.4 Firewalls

A firewall is a form of logical security implementation deployed to filter traffic between networks whose trust levels are different or within a network to filter packets between subnets. These devices have evolved from the initial design concept that started this technology as discussed here under.

2.4.1 History of Firewalls

The name Firewall is derived from the ancient design of building a wall between two or more buildings or rooms to stop the spread of fire if it broke out [38].The first generation of firewalls were made by Cisco Systems Inc.'s IOS software division around 1985. They were called Packet Filter firewalls. Late 80s into the 90s saw the second generation of firewalls called Circuit Level Firewall pioneered by Dave Presotto and Howard Trickey of AT&T Bell laboratories [38, 39]. They also worked on the third generation called Application Layer Firewalls and were publicized in 1991 by Marcus Ranum of AT&T Bell laboratories. Ranum's work became the first commercial product called SEAL by Digital Equipment Corporation. Check Point Software released the first commercial firewall product based on the fourth generation architecture in 1994 [30, 14]. The fifth generation firewall architecture, the Kernel Proxy architecture was worked on by Scott Wiegel, chief scientist at Global Internet Software Group Inc. in 1996. Cisco Systems Inc released the first commercial product, Cisco Centri Firewall, based on this architecture in 1997 [37].

Firewalls have since evolved through generations to the current stage that not only filters packets.Modern implementations incorporating functions like management of network bandwidth, enforcing requirements to use a web proxy server, protecting resources in the organisation, preventing malicious attackers from the Internet among others [30, 31]. The growth of networks, the desire by companies to share their intranets with customers and business partners has brought challenges and a greater need for network security [25].The innovations in firewall technologies resulted from Department of Defense research and funding projects [38].These advances and improvements have come by through research by different pioneering companies and individuals among them Cisco Systems Inc., AT & T Bell, and Digital Equipment Corporation as pointed out already.

Irrespective of the type of firewall being used, firewalls provide several services some of which are discussed below:

- *Traffic shaping:* This is a strategy to optimize performance and manage traffic on a network. This is done to make sure a network does not become overloaded. This takes traffic transport into consideration and is implemented on a firewall by techniques such as giving data packets different priority levels as they go through the firewall.
- *Network differentiation:* A firewall creates a boundary between networks by creating a clear distinction between them. This can be within the same company environment to divide departments
- *Content filtering:* Proxy level firewalls are normally configured to manage and control traffic by inspecting URL and page content. Proxy oriented firewalls can identify and block content that is seen suspicious.

- *Network Address Translation (NAT):* As a security feature and IP reservation mechanism. NAT makes a firewall at the border gateway secure the internal addressing from attacks as their addresses are not routed outside. This also reduces costs of acquiring public addresses from ISPs
- *Bandwidth management:* Some applications can be allocated more bandwidth than others. FreeBSD ipfw firewall for example has ALTQ a QoS mechanism allowing prioritizing flows.
- *Auditing*: Firewall logs are a good source of security planning and reviews. Object access and failures can be logged to a file and retrieved for audit purposes. FreeBSD ipfw and other firewalls offer this feature that captures the IP address of a source host, the ports or service they were targeting on the resident network.
- *Packet Redirection:* A firewall may be required to send traffic to another port or another host. These situations arise in environments that require sending certain traffic to servers for further processing.

The digital convergence and the desire for businesses to communicate, collaborate and share work using IP networks has brought more challenges in terms of information security [24]. This has given a further demand on the speed a network should avail to such services and applications. At the core of this speedy delivery of traffic to and from a network is an Intrusion Detection System, IDS. One IDS key component is a firewall which this research is focusing on.

2.4.2 Firewall Security

The firewall security focus of this research was network layer packet filtering using a rule set written based on a security policy. A flawed firewall configuration for example could leave the system open to attacks like Denial of Service, DoS, attacks [12]. Several issues need to be considered when dealing with packet filtering. Internet connectivity is growing with most enterprises shifting to the use of web based services for service provision [22]. This is seen to be sharpening their competitive edge as it gives them and their customers, rapid access to information.

Firewalls provide a mechanism for protecting these enterprises from the less secure Internet over which their customers or collaborating partners transfer packets destined for the corporate network [12, 19, 29]. They are deployed as first lines of defense at the network perimeter .Firewalls have grown to take up more tasks than merely filtering traffic to managing bandwidth, routing control, packet forwarding etc [15, 34]. This research focused on the Network Layer operation of a firewall and explored ways of improving its performance by optimizing the rule set. A corporate network policy is translated into rules that are defined as the rule base or rule set. These rules are configured on the firewall to provide the security and packet availability. The idea from a security point is ensuring that data, system integrity and confidentiality are maintained [2, 20, 19].

There has been a drastic growth of connected nodes on the Internet and in private networks [26]. This exponential digital growth in the last decade has not only improved network technologies but brought more complex security threats and demands on packet filters [8]. Research in the subject of improving firewall performance continues to be done by many network researchers some of whose work was evaluated and learned from to get a view of what has been done.This section presents various approaches and techniques of firewall rule set optimization. The chapter closes by looking at some tools and approaches proposed to aid rule-set optimization.

2.4.3 Firewall Categories

Firewalls are categories according to different approaches as their functionality and need relates to a given network. Below are the classes and examples for each category.

- Processing mode (Packet filtering, Application gateways, Circuit gateways, MAC layer firewalls and Hybrids)
- Development era (generations according to time).
- Intended deployment structure (Stand alone, self contained, Small Office/Home Office SOHO, Commercial grade)
- Architectural implementation (Packet filtering routers, Screened host firewalls, Dual-homed firewalls, and Screened subnet firewalls)

2.5 Firewall Types

This categorizes firewalls according to how they perform packet filtering. It considers levels of deployment and key functionality involved.

2.5.1 Packet Filter Firewall

These are based on first generation firewall technology. They analyze network traffic at the transport layer [30]. They examine each IP network packet to see if it matches one of the rules defined for allowing or denying data flows. The decision is based on the information they get from the packet's transport layer headers and the direction the packet is going into [37].They are therefore configured to check:

- Transport layer type (TCP, ICMP, UDP)
- Source port
- Destination IP address
- Source IP address
- Network interface the packet arrives on
- Destination port

Packet filters do the above by applying a rule set residing in the TCP/IP kernel that defines what action goes with which rule. They come under three subsets being:

Static Filtering:These firewalls require that filtering rules governing how the firewall decides which packets are allowed and which are denied are developed and installed. Hence the name static [3, 24].

Dynamic Filtering:Dynamic filtering firewalls allow the firewall to react to emergent event and update or create rules to deal with the event dynamically.

Stateful Filtering:These types of firewalls keep track of each network connection between internal and external systems using a state table [41]. Filtering decisions are made based on the kept information about a given connection. A state is kept by a rule that specifies keep state so that responses to sent packets are monitored especially for fragmented IP packets and UDP packets.

2.5.2 Circuit Level Firewalls

These are based on second generation firewall technology [38]. Their principle of operation is based on the fact that a packet is either a data packet or a connection request belonging to a connection or circuit between two peer transport layers [37]. Circuit level firewalls work by checking that each connection setup follows a handshake system for the transport layer protocol being used. They also store session identifiers and connection state in a table of valid connections and removing them once connection is terminated. Connection tear-down happens once transmission is complete.

Like packet filters, circuit level firewalls work by applying a rule set that is maintained in the TCP/IP kernel [14, 39]. They allow all network packets bound for that connection through the firewall as stated in the firewall server's routing table but do not do further security checks on the packets [37]. They also perform Network Address Translation and perform checks to see if a packet has been spoofed and whether the data being transported complies with the protocol definition.

2.5.3 Application Layer Firewall

Also called third generation firewall; these firewalls evaluate packets for valid data at the application layer before allowing a connection [34].

- Examines data in network packets at the application layer
- Maintains connection state and sequencing information
- Can validate passwords and service requests
- Most of them include proxy services for specific services such as HTTP or FTP which provide more checks and generate audit records about the traffic they transfer.

Application layer firewalls generally do not focus on the network packet information. This makes the risk higher if the network layer is not performing well [35]. They also employ proxies that require additional passwords or some validation of some sort. This brings more delays.

2.5.4 Dynamic Firewalls (Stateful Firewall)

A fourth generation firewall type allowing modification of the rule set. A virtual connection is established and the packet is allowed to travel the firewall server [30]. These provide support for UDP packets by associating them with a virtual connection [31]. The connection information is kept for a short period and the connection is terminated if no response packet is received within that short time. They are good for not allowing unwanted UDP packets into a network because the response packet must contain a destination address that matches the original source address [38].

2.5.5 Hybrid Firewall

Because of the need to do more than packet inspection, firewalls are being implemented as hybrid systems. These are mostly implemented by adding packet filtering to an application gateway. Cisco PIX firewalls are an example of such hybrid firewalls [38].

2.6 Security Policy

Packet filtering is done based on a set of rules written and configured on the firewall device. These firewall rules are written based on the organization's security policy [24, 41]. They need to be built on clear business ideas. This is because various business reasons will demand different communication to be allowed through a firewall. As such, the security policy must take care of the different subnet traffic needs so that when rules for the firewall are written, filtering matches the security considerations.Each protocol carries with it some risks and these risks differ from protocol to protocol therefore, protocol analysis is key when coming up with a security policy [41]. Cases of having subnets with different traffic considerations make writing security policies complex. This also grows rule sets to lengths that network administrators do not understand any more, especially if rules are written by different administrators [43].

2.7 Firewall Filtering Complications

There are various factors that come into play when dealing with packet filtering. These are mostly requirements and values on which packet filtering is performed. Useful as they may be, some of them pose challenges and security risks in firewall functionality.

2.7.1 Protocols

Packet filtering is one complex if not hard aspect of network security [15, 30]. For example, you can permit or deny protocols by port numbers; it is hard to allow some operations while denying others in the same protocol. You also can never be sure that the packets coming in on that port is actually the protocol you wanted to allow [8]. Therefore, characteristics of protocols, services and implementations greatly affect how effective a firewall will be.

2.7.2 IP Fragmentation

IP fragmentation is one huge complication in firewall performance. IP packets are fragmented to smaller sizes to meet Maximum Transmission Unit (MTU) requirements for the links they are transported on [5]. They are then reassembled into the original IP packet at the destination. A Fiber Distributed Data Interfac (FDDI) frame for example, is much bigger than an Ethernet frame. A router between the FDDI ring and an Ethernet one may have to split an IP packet that fists into a single FDDI frame into multiple fragments that fit into smaller Ethernet frames [7]. This is a problem from the packet filtering point at firewall level [7]. Only the first of the IP fragments contains protocol headers TCP or UDP) from the original packet that are vital for making filtering decisions. IPFW firewall supports IP fragmentation using the *frag* keyword on a rule, which applies the filtering rules to the starting packet that has necessary protocol data and allow or deny the rest of the packets that are part of that *established* transmission by using *keep state [14]*. This is a dangerous action in that the decision on the remaining packets is based on the assumption that the first packet was allowed or denied entry. The figure below explains packet and protocol layer relationship.

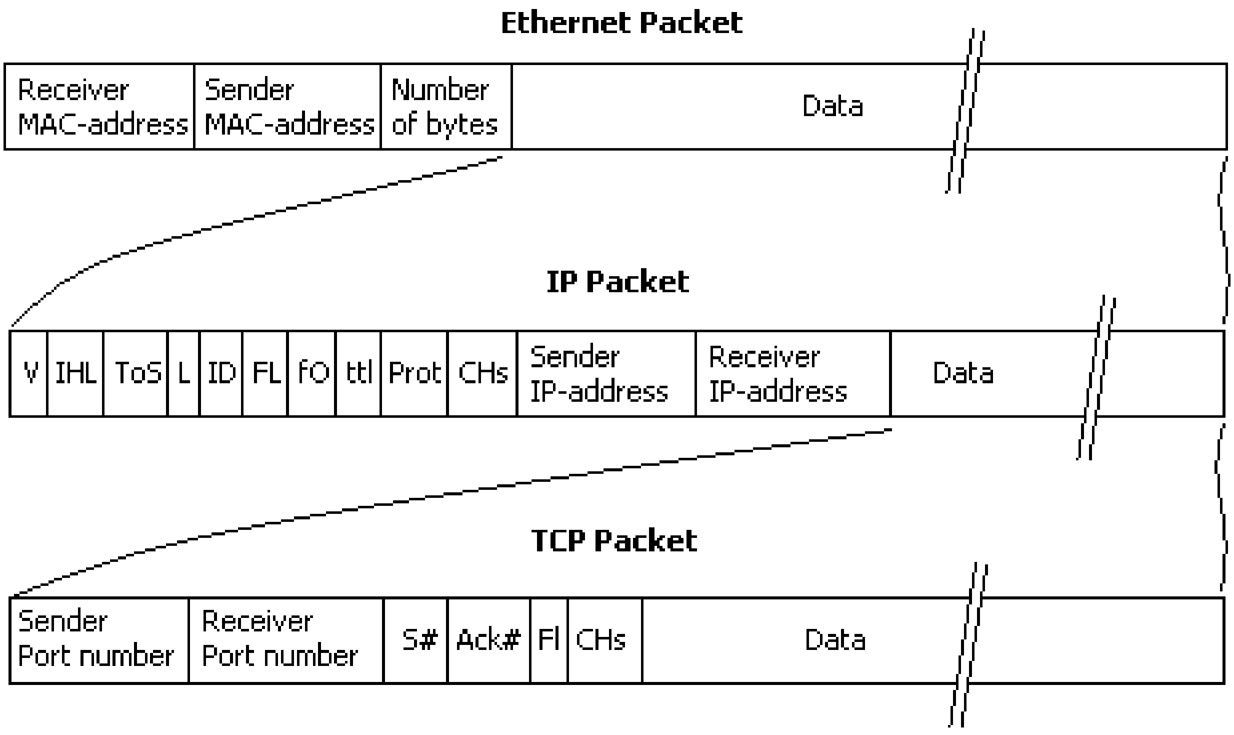


Figure 2.1: Packet Layer Relationship (Adapted from [3])

2.7.3 IP Source Filtering

This is another potential source of abuse by attackers. Basically allowing filtering to be based on IP sources and destinations ties the firewall to trust those trusted addresses. Unauthorized packets can get through purporting to be from those addresses listed as trusted ones in the firewall. To counter this, an enhancement of Internet Protocol called Path Enhanced IP (PEIP) has been developed [6]. PEIP is designed to detect forged source addresses, their original source and even filter them.Another solution to this is to only allow in response packets to sent out packets for which dynamic rules have been established [5].

2.7.4 Filtering Criteria

Packet filtering can be performed based on Protocol, IP address, Interface, and Port number or service name [3]. Filtering by hostname is possible but must never be done.

This is because hostnames can be easily changed intentionally to mask bad blacklisted sources and these get fed to your network's DNS server.

2.8 Summary

The above architectures have their own advantages and disadvantages when evaluated based on performance and security. Based on performance, Packet Filers provide the highest performance, then Circuit Level, Dynamic Packet Filter firewalls and Application Layer firewalls [30, 38]. This is a reverse in terms of security. Application Layer Firewalls make packets go through more protocol layers inspection and detail [25]. Application Layer Firewall therefore are seen to be more secure than Dynamic Packet filter firewalls that are viewed more safer than Circuit Level firewalls and Packet filter firewalls [39]. Application Layer firewalls are more process intensive and hence slower but considered to be the most secure.

A firewall is just one part of the network security system of a private network. These firewalls do not protect the network from viruses, insider attacks and previously unknown attacks [35].This is because most firewall technology works on a 'catch up' and 'protect from known threats' technology [37]. Therefore keeping the network secure demands continued updates of the firewall rule set and other security policies [31].The fact that a network is dynamic and not all rules in the firewall rule base are matched at all times is the motivation for this project which is seeking to find ways of optimizing the firewall rule set to improve throughput at layer 3, Network Layer.

IPFW firewall employs some of the filtering implementations discussed above. It can perform dynamic or stateful filtering by specifying keep state and check state on a rule. It is thus considered to be a hybrid type network layer firewall.

Defining rules to filter traffic need to be well understood. Key issues to consider in that exercise are essentially based on the security policy. Protocol implementations, filtering choices, desired services and choices of filtering must be well dealt with to have in depth of what is being allowed or denied and how it impacts on the overall security of the network [7, 8].

2.9 Rule-set Optimization Techniques

Increased firewall complexity breeds more vulnerability and may reduce availability of network services and applications an enterprise uses [15, 2]. The increase in network size, bandwidth, and processing power of networked hosts continues to increase the demand for optimizing firewall operations to improve performance [22]. The following techniques are some of the many approaches previous works have considered in optimizing firewall rule sets.

2.9.1 Rule set clean up

This approach analyzes the rule set to identify inconsistencies and redundancy in the rule set and removes them. Rules that make no unique contribution to the firewall behavior are removed. These are either Redundant or Shadowed rules [33]. Redundant rules never match packets because there are more preceding rules matched first. Unused rules that have the log option but have no logs showing that they have matched packets. They are therefore candidates for removal when optimizing though their removal may affect firewall behavior later [28]. Remove rules for unused Network groups; if the organization does not have a mail server, SMTP rules can be taken out of the configuration [33]. This should also be done for unused Network objects. Remove unused service objects and disable them on machines. A blend of this depends on the specific organizational network needs and should be tailored well to achieve better packet filtering that improves throughput [26].

2.9.2 Rule set re-ordering

This technique relies on the network statistics logged on the firewall or written to a file in a network database. Jukka Zitting *et al* [13] and Hunt [22] suggested listing most used rules in decreasing order of usage by hit count and percentage hit count. These rules can then be moved towards the beginning of the rule set to improve performance. Optimize the rule set by ordering rules based on the rule usage data and rule order dependencies that does not alter the firewall behavior [28]. This moves the most used rules toward the beginning of the rule set until they are very close to the source of an order dependency. The complexity with this method is the dependencies problem [16] and the traffic pattern changes [24].

2.9.3 Rule Grouping

It is evident that a major part of the network traffic matches a small subset of the firewall rules [20, 14, 21, 33]. This therefore calls for selecting these rules and calling them by groups depending on their usage [12]. This scheme divides the filtering policy into two layers of rules, (a) most active rules-those performing the most packet matching and (b) inactive rules-perform much less matching. Rules are checked and if two or more rules are found to have the same matching action, they are merged [33]. This reduces the rule set size and consequently the search time for the filtering algorithm because less rules are inspected by the firewall to deny or allow a packet.

2.9.4 Rule Frequency re-ordering

The number of times a rule is triggered is recorded and used to determine matching patterns and arrangement of the whole rule set [19]. Ehab Al-Shaer *et.al.* proposed an adaptive way of dynamically optimizing firewall rule sets using actively calculated statistics. This looks at other filtering categories other than the traditional IP header. Subrata *et.al.*[2] and Ehab Al-Shaer *et.al.*[19] presented a technique that uses Internet traffic characteristics to optimize firewall filtering policies. They have called this method statistical matching. It has however been found challenging to implement with fears of performance degradation [34].

2.9.5 Rule Editing

Firewalls have thousands of rules and hundreds of IP addresses to take care of. The typical approach is to scan through all these rules in a linear fashion until a match is found for the packet being inspected [15, 29, 34]. In trying to change this scenario, first remove the configuration errors or anomalies in the implemented rule set [21]. This can be done by aggregating rules, rule checking against matched packets, rearranging them, and other algorithm specific ways e.g. Pre-calculating values though it uses up memory [5]. This helps remove these four classed anomalies;

1. Shadowing : rules not matching packets because earlier rules are matched.
2. Redundancy : rules not matched by packets
3. correlation : rules performing the same action.

4. irelevance: rules for unavailable services and or protocols.

Rules can be edited from being too generic to more specific.This requires religiously observing dependencies in order not to open the firewall up to allow undesired traffic [17]. Most techniques used to classify packets use the behavior of filtering rules without taking into account, traffic behavior in their optimization algorithms [19].

2.9.6 Go To Function

Modern firewalls come with a feature allowing skipping from one rule or rule set to another rule in a rule set [43]. The go to function is used to switch the search and match flow from the default one (next rule in the list) to the one specified in the go to command. In IP Tables it is called "jump" [15, 35]. This causes the search algorithm to skip all rules following until it reaches the specified one called a "target". IPFW has this "skipto" functionality i.e. skip the following rules until the one specified after the skipto [14]. This works in a firewall more like a break or jump in a normal program. This function or command makes optimization possible in that, not all rules in a list are checked for matching with a given packet as Layer 3 firewall is concerned.

2.9.7 Dynamic Filtering

Hunt [22], Subrata [2], and Hamed [19], in their publication proposed ways of creating firewalls that adapt to traffic changes dynamically. These traffic aware firewalls are proposed to work on previous traffic patterns to detect new transmissions and react. Andrea *et al* [28] suggested creating such an adaptive way of optimizing rule sets by using rule ranking so that less priority rules are pushed to the bottom of the rules set as they are not expected to match most of the traffic. This is argued to be a good way of getting matches fast and reducing the time as most firewalls exit inspection once a match is found.

2.9.8 Dynamic Rule-Ordering

A dynamic way of re-ordering rules was suggested by Ehab Al-Shaer and Hazem Hamed [19]. Their suggested technique utilizes Internet traffic characteristics to optimize firewall filtering rules. The technique timely adapts to traffic changing conditions using actively calculated statistics to dynamically optimize the ordering of packet filtering rules. This takes into account, rule importance in traffic matching and dependency of rules on other rules to perform the optimization. Just like [2] and [22], they have considered traffic characteristics to come up with a way of making the firewall traffic-aware as it starts inspecting new flows. They have pointed out how hard the optimization problem is to solve in polynomial time and have presented a heuristic approximation algorithm that achieves near-optimal results for the most common firewall policies.

2.9.9 Early Packet rejection

The default deny rule is placed at the end of a rule set and it matches all undesired traffic. Adel *et al* [20] proposed a technique that analyzes the firewall rule-set in order to come up with a sub set of rules that can reject maximum number of undesired packets as early as possible. They have however acknowledged that this is an NP-complete problem and have as such used an approximation algorithm that pre-processes the firewall rule-set off-line and generates different close to optimal solution [20]. A solution that seen to be more appropriate in this approach is one that has the least overhead cost, which is dynamically selected based on network traffic statistics [11].

2.10 Rule-set Optimizing Tools

There is work done in the research and design of tools that can be used to optimize firewall rule-sets. These tools have taken different design approaches and incorporate different ways of looking at traffic in relation to rule-sets and security policies.

2.10.1 Athena FirePAC [1]

This is a proprietary tool owned by Athena Security Inc. It operates on off-line network traffic to perform firewall rule set checks on Cisco, Check Point and Netscreen firewalls. Its functionality is centered on removing vulnerabilities, non-compliance and errors in the firewall configuration. It is configured to perform comparison functionality to see how packet flows have been affected by specific rule and object changes [33]. Its auditing feature finds misconfiguration, redundant rules,overshadowed rules, and unused objects that

[1]http://www.athenasecurity.net/athenafirepac.html

can be removed to optimize the rule-set. It does not perform rule removal automatically but recommends which rules can be removed and rule re-ordering based on audited traffic flows. Therefore, it is more of a monitoring tool and a good one for use by administrators to proceed with rule-set optimization.

2.10.2 OPTWALL [2]

This is a hierarchical traffic aware framework for firewall rule-set optimization proposed by Mehmud *et al* [1] of the University of Pittsburgh. OPTWALL divides a rule-set into multiple rule sets to reduce the packet matching time to a rule. The proposed tool offers adaption schemes that should dynamically change priority of a rule based on the traffic. This adaptation is based on a heuristic solution that takes initial filter determinations of hit count per rule, hit count of related rules and a random measure of how often those rules match traffic. Those values are then used to calculate the cost of a rule. They have presented a 35% improvement in operational cost of a heavily loaded firewall [1]. This proposed solution has not been produced yet but does show that optimizing a rule set offers a speed up in packet inspection time.

2.10.3 Policy Advisor [3]

This tool was suggested by Ehab Al-Shaer in his research paper [4]. It takes into consideration, rule set design to recommend optimization techniques. Their focus was on correctness and usability of firewall rule-sets using this tool than the computation complexity and optimization of the algorithm. It inspects a rule-set for shadowing anomaly, correlation anomaly, redundancy anomaly, and generalization anomaly. Their anomaly detection algorithm works by applying relationships to rules in a set and compares the fields specified for matching actions to recommend an optimization action. The administrator can then perform those actions from a graphical interface. This too uses offline analysis to process recommendations on the rule set.

[2]http://www.chautari.org/forums/index.php?showtopic=12303
[3]http://www.mnlab.cs.depaul.edu/projects/SPA/

2.11 Observations

Other methods used explore the subject in a similar way as outlined above though their naming conversions may be different. Adel El-Atawy *et al* [20] propose a technique that uses dynamic statistics collected on firewall policy rules and uses these results to construct a set of rules for denying or allowing traffic. Most research has presented and acknowledged that this is an NP-Complete problem [26, 33]. They used an algorithm that processes the firewall policy offline and comes up with optimal solutions. The solution with the least processing cost is dynamically selected based on network traffic statistics. This can be likened to [2].

Heuristic based algorithms have been employed in finding the shortest time of matching a packet by doing the least search [16]. These are a combination of most of the techniques discussed above. Disjoint rule set creator for rule set based optimization [2], Direct Acyclical Graphs [16], Filtering Trees [20] among others, all use one or more of the above techniques as a single implementation or in a hybrid form. Most of these methods are based on the standard performance evaluation benchmarks set by the Internet Engineering Task Force in [?].

2.12 Summary

An overview of network concepts from addressing, protocols, security, to packet filtering and considered filtering complications have been discussed in this chapter. Traffic filtering, complex has it is, can be done at different levels being application, proxy and network levels. All filtering work conducted in this research focused on network layer firewalling unless explicitly stated.

Firewall rule set optimization means reducing the rule set size to achieve optimal inspection time . This takes many approaches like rule grouping, merging, skipping, and rule editing among others.Work to find better ways of optimizing a rule set while maintaining the firewall security semantics continues receiving attention. Hardware improvements and increased network speeds are calling for better filtering approaches. As such, overwhelming firewalls with undue processing must be avoided to get better performance by reducing the inspection time. It is evident that traffic traversing a network does not match all defined rules in the rule set. This therefore increases the processing time unnecessarily and drastically impacts on firewall performance with regard to throughput and consequently security. Rule sets grow to large depths and therefore optimizing them is necessary. Making these rules react more dynamically to ever changing traffic with varying characteristics will answer most of the firewall performance problems. This has however been described as an NP-Hard task to accomplish [19, 33]. The optimization approaches this research has taken combines packet filtering functionality with traffic characteristics.Then taking the approaches into designing a tool that can be integrated in a firewall to optimize the rule set and give the much sought after increase in throughput.

Chapter 3

Project Methodology

Conducting investigations regarding rule set optimization requires an environment in which all parameters necessary would be blended to come up with conclusive results. As such, this section describes the experimental setup. It give the hardware, software, tools and other components used in performing the research. It begins with a description of the firewall package used; detailing its features and implementation architecture that made it useful in this research. A look at the tools used to perform tests and results gathering are given at the end and the reasoning behind using them explained in the respective sub-sections.

3.1 IPFW FreeBSD Firewall

The firewall used for all testing done in this research was *ipfw* a FreeBSD IP packet filter and traffic accounting facility. It is included in the FreeBSD install as a separate run time loadable module. The system dynamically loads the kernel module when the *rc.conf* statement *firewall_enable="YES"* is used. It is also available in other operating systems, open source and proprietary;adapted as WIPFW for Microsoft windows operating systems, Apple in their Mac OS operating systems series have implemented ipfw [9]. Linux implements it as a variation called Netfilter, a framework for providing firewalling tools - iptables and nftables [35]. IPFW was used because of the built in features discussed below that make it efficient and adaptive:

- ***NAT support:*** offers in-kernel Network Address Translation using the various options available for the parent Nat command; *nat nat_ number config nat-configuration*. NAT in ipfw helps in masking the internal addressing of a network as the firewall uses a public well known address when filtering outbound traffic and mapping responses for the internal addresses.

- ***Traffic Logging:*** A packet that matches a rule with the *log* keyword creates a message that will be logged to the specified log file. The rest of the packets update counters of the accounting module depending on specified options in *etc/rc.conf* and in the kernel. This was a source of information critical for rule set optimization.

- ***Kernel mode control:*** IPFW is built into the FreeBSD kernel as a utility. This makes it faster in responce processing,good communication, easy manipulation and configuration as opposed to implementations that are loaded from external installation directories **.**

- ***IP accounting:*** The ability to log traffic offers a good way of accounting for object access and failures. This is the source of general statistics in log files and counters. This makes it possible to see what ports or services malicious traffic is targeting. It also works as an internal control system for user behavior on the network.

- ***Traffic shaping:*** This is implemented using dummynet, a traffic shaper, packet scheduler and network emulator. Dummynet is a IPFW subsystem that can be used to delay or drop packets just like some network links or queuing systems**.**

- ***Dynamic rule support***: upon a match, the firewall will create a dynamic rule. This rule's default behavior is to match bidirectional traffic between source and destination IP/port using the same protocol. Dynamic rules have a limited lifetime which is refreshed every time a matching packet is found. This keeps the host safe from flood attacks from fake TCP packets and helps fight DoS attacks.

- ***Stateful Firewall***: this is a way of keeping connections for protocols requiring responses to their communication. This too creates dynamic rules and offers support for stateless protocols once the *check-state*, *keep-state* and *limit* option is enabled on a rule. *ipfw add check-state, ipfw add deny tcp from any to any established, ipfw add allow tcp from my-net to any setup keep-state [12].* This will allow the firewall to create dynamic rules only for those connections that start with a regular SYN packet coming from the inside of the network thereby protecting from spoofed requests [14].

3.2 Firewall Rule set

A rule set is a group of rules programmed to allow or deny packets.The decision to allow or deny is based on the values contained in the packet. The firewall rule-set processes both packets arriving from the public Internet, as well as the packets originating from the internal network [14, 35]. Every service based on TCP/IP i.e.: telnet, www, mail, etc. is predefined by its protocol i.e SSH, HTTP, SMTP e.t.c and privileged (listening) port.Packets destined for a specific service, originate from the source address using an unprivileged (high order) port and target the specific service port on the destination address. The above parameters (ports, addresses and protocols) are used as traditional selection criteria to create rules which will pass or block packets [5, 15, 34]. Collectively, they are referred to as the 5-tuple {Source IP Source Port Destination IP Destination Port Protocol} The table below shows an example filtering policy written in rules structure by adding an action to the 5-tuple.

1	2 , 3	4 , 5	Action
Protocol	Source Address:port	Destination Address:port	Action
1 tcp	* . * . * . * :any	172.128.16.41:25	allow
2 tcp	146.231.125.30:any	* . * . * . * :21	deny
3 tcp	* . * . * . * :any	172.128.16.*:21	deny
4 tcp	146.231.125. *:any	* . * . * . *:21	allow
5 tcp	* . * . * . * :any	172.128.16.*:22	allow
6 tcp	146.231.125. *:any	* . * . * . * :80	deny
7 tcp	* . * . * . * :any	172.128.16.40:80	allow
8 tcp	* . * . * .* :any	172.128.16.42:53	allow
9 udp	* . * . * .* :any	172.128.16.42:53	allow

Figure 3.1: Sample Filtering Policy

3.2.1 Packet Inspection

When a packet enters the firewall it is compared against the first rule in the rule-set and progresses one rule at a time moving from top to bottom of the set in ascending rule number sequence order. When the packet matches the selection parameters of a rule, the rules' action field value is executed and the search of the rule-set terminates for that packet [14, 15, 35]. This is referred to as 'first match wins' search method. If the packet does not match any of the rules, it is caught by the default rule, number 65535 *(deny all)* which denies all packets and discards them without any reply back to the source [14, 28]. It is important that no reply is sent for those packets so that attackers sending such malicious packets do not get to know the transmission status and addresses replying to their packets.

3.2.2 Rule Manipulation

Firewall rules in ipfw can be added in two ways [7]. Using the *ipfw* command at run time or writing them in a file to be loaded at boot time. Adding or deleting from the active firewall internal rules while the firewall is running can be convenient for reacting to observed trends without service interruption [29]. The drawback with this approach is that once the firewall is stopped, all rules that were changed, added or deleted are lost. Rule sets used in this research were of the second method of writing rules in a file for better administration and manipulation of rules. This file is read by *ipfw* and rules loaded at boot time.

3.2.3 Rule Set Types

Two types of rule sets have been named according to how they are written and perform packet filtering [24]. That classification defines a rule set as either Inclusive or Exclusive. The difference between these types is the way the firewalls they create do packet filtering.

Exclusive Rule sets

Exclusive rule sets create firewalls that allow all traffic through except for the traffic matching the rule-set [8]. Inclusive firewalls allow traffic matching the rules through and deny everything else.

Inclusive Rule sets

Inclusive rule-sets create firewalls that offer much more control of both outgoing and inbound traffic. This is the type used for all rule sets implemented in this research. Because of more control, inclusive firewalls reduce chances of allowing unwanted traffic to pass through significantly [8].

3.2.4 Used Rule-set

This research utilized inclusive rule sets for the firewall tests performed. It was a mix of stateful and stateless network firewalling because; a single implementation ends up oping the system to attacks. Stateful firewalling for example is a good implementation that tightens security further. It keeps track of which connections are opened up through the firewall and only allows traffic that matches the connection or opens a new one. This can open the network to Denial of Service (DoS) attacks if lots of new connections are opened and their state maintained [7]. Stateful firewalling treats traffic as a bi-directional exchange of packet for a single session. This allows any packet that ipfw stateful filter will be certain to be part of an active session to pass through even if it is a different protocol.

The inclusive firewall used included rules to allow free movement of special internally used packets that are operating system critical and if blocked would lock out the system from any interaction. FreeBSD being a Unix system is designed to use the loopback interface lo0 and IP address 127.0.0.1 for internal operating system communication [18].

3.2.5 Rule Syntax

Firewall configuration is an involving task considering that the security policy must be understood and broken into a rule set [35]. Rule sets grow to large numbers of rules and network traffic trends keep changing [31]. For the general syntax of ipfw firewall rules, refer to appendix C.

3.2.6 Rule Numbering

Ipfw rules like most firewall rules in production environments should be numbered as a good practice for easy administration [15]. Ipfw numbers rules automatically from 00001 to 65535 by adding 100 to the previous rule if a number is not set manually on the rule. This complicates manipulation of rules as the the numbering is not flexible to track. The last rule 65535 usually has the default deny action but may be different depending on the kernel build option [15]. This rule matches all packets and that fall through without matching earlier rules.This makes sure unwanted packets do not enter the network. It is not mandatory to have all the rules from 00001 to 65535 because security needs and services differ from network to network. Rules can also show duplicate numbers in the same rule set [4].

3.2.7 Ports

Ports are the end points to logical connections and specify the way a client program communicates with a server on a network [32]. Ports represent named services and applications. e.g HTTP port 80 RIP port 520. Ports 0 to 1023 are reserved as well known ports for privileged services and applications. Random and dynamic ports are assigned from 1024 to 65535 [3]. All these ports are used by clients when initiating connections to hosts that they need to communicate with. Refer to [32]for more details on assigned port numbers.

Firewalls use these ports combined with the protocol, address, interface and other selected values to perform their filtering [27]. It is therefore cardinal that these ports are well known when writing rules to avoid locking the network from receiving legitimate traffic.Also to avoid opening the network to malicious traffic meant to be denied entry.

Using the syntax discussed above, rules in a rule set are then written based on the security policy of the organization. This must be well reviewed to get the best of the filtering and not allow what is meant to be denied. Refer to Appendix A for the example *ipfw* rule set.

3.3 Test Environment

In this section, the environment in which the tests were performed is presented. The researcher gives details of the testbed layout, starting with a brief explanation of the virtual machines networking followed by how the virtual machines used were configured. A description of the FreeBSD *ipfw* kernel module follows on to detail how *ipfw* firewall works with traffic traversal, traffic statistics logging and extraction.

3.3.1 Environment Setup

This research was performed in a visualized network environment using virtualisation software, various tools and test parameters. The firewall host operated in *bridged mode* configuration connecting two networks configured on its two interfaces. One interface *em0* was configured with a private IP address and the other interface *em1* was configured with the public IP simulating a public Internet address. These two interfaces were used to create the bridge:

```
# ifconfig bridge create

bridge0

# ifconfig bridge0

#ifconfig bridge0 addm em0 addm em1 up
```

The interfaces em0 and em1 were configured with IP addresses and only added as members of the bridge and setting the bridge up. The client hosts connected to the firewall for test purposes detailed in the following sections. The Linux host was also used for exporting graphics from the firewall machine using secure shell.

```
mungole@ubuntu:~$ sudo ssh -X -vvv root@192.168.46.134

[sudo] password for mungole:
```

The Bridged firewall in the diagram below was assigned the IP 192.168.46.134 on the bridge interface shown for remote access as earlier mentioned.

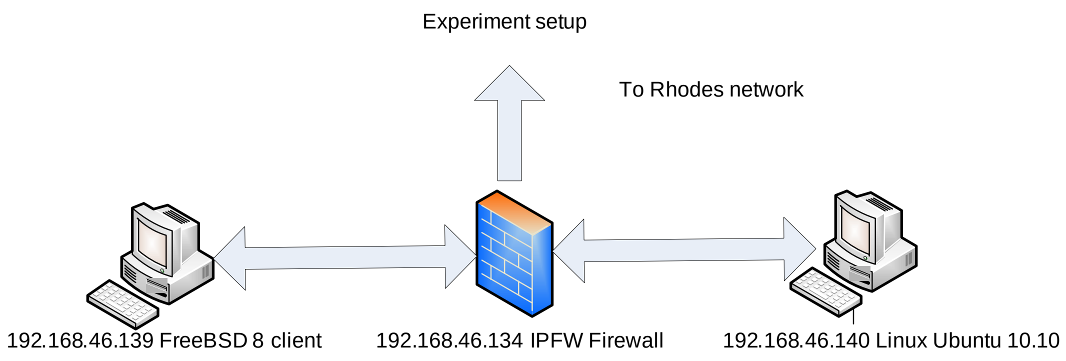

Experiment setup
To Rhodes network
192.168.46.139 FreeBSD 8 client
192.168.46.134 IPFW Firewall
192.168.46.140 Linux Ubuntu 10.10

Figure 3.2: Test Environment

3.3.2 VMWare

Due to the advances in virtiualisation, the researcher conducted all experimental work using virtual machines in VMWare workstation version 7 [23]. VMWare workstation is a software that allows setting up complex testing networks and networked applications running on different platforms all on a single physical machine. For this implementation, two operating systems were used Linux Ubuntu 10, two FreeBSD 8, one being the firewall and the other two as clients.These operating systems were isolated in secure virtual machines that co-existed on a single piece of hardware - the lab PC. This is because virtualisation maps the physical PC resources to the virtual machines resources giving each VM host, its own CPU,memory,disks and I/O devices, etc [5].

Tests done in this research used data sources that contained multiple protocols. Clients specified application layer entities which were not associated with a direct physical interface but did receive traffic.

3.3.3 Virtual Machines Configuration

The three virtual machines run FreeBSD 8, (IPFW firewall and a client) and Linux Ubuntu 10 as their operating system. These hosts, were connecting to the Internet using Network Address Translation (NAT), for installation of testing tools and updates.

IPFW Host

The ipfw firewall host had graphing tools installed among them RRDTOOL, GNUPLOT, MRTG, IPFW-Graph. The use of more than one tool was to explore and see which one presented results best.IPFW-Graph was used to get the results and then graphed them using another tool for presentation purposes.This host also had installed, packet crafting and injecting tools like Nemesis, TCPReplay, Bittwist and Hping2 to craft and send different protocol based packets through the *bridge* on the firewall. The firewall type was set to simple in rc.conf to make the firewall act like it was protecting a full networking. A file name to load rules from was specified so as to avoid loading default rules generated by the ipfw module. Refer to the Appendix section for the rule files used. These tools were used to send packets from the firewall in *Bridged* mode - the DUT assumes it is forwarding packets to a host connected to the bridge interface.The firewall host had various services enabled on the rules and in the host itself to support different traffic types some of which are listed here under:

- FTP
- DNS
- SSH
- NFS
- IPv6 support
- SMTP
- HTTP
- ICMP
- Samba

FreeBSD and Linux Host

The services above were also enabled on the hosts to allow flow of various packets to the clients through the firewall. This made the firewall to inspect the client traffic against the configured rule set and take appropriate action taken depending on the rules. These two clients were Linux Ubuntu 10 and FreeBSD 8. Clients had packet generating and crafting tools installed and worked as the untrusted network sources sending packets that the firewall was filtering.

3.3.4 Tools

A variety of tools were tested along the project life time and a few taken as candidates for traffic generation. Tools were selected based on their capabilities in creating packets of variable length, different datagram protocols and also the general behavior of their crafted packet in traversing the firewall for firewall testing. Their packets were used to check both rule sets traffic expected to be allowed or denied. This was necessary to isolate issues with certain protocols.

Nemesis [1]

This is a command line based tool authored by Jeff Nathan for crafting and injecting custom packets for Unix and windows based systems to test firewall and other devices. Nemesis provided a good source of the many packets it creates and injects; *ARP, ICMP,IP,OSPF,RIP,TCP,ETHERNET,IGMP* and *UDP* packets. It was used in IP and Ethernet injection modes; those modes offer a chance to craft and inject any packet.

Hping2[2]

This tool works on Unix based systems.It is a command-line based tool that supports sending *TCP, UDP,ICMP* and *RAW-IP* protocol. It proved useful in firewall testing and supports fragmentation which is one of the features filtering testing considers a challenge.

Bittwist [3]

Bittwist is a libcap based utility built to complement *tcpdump.* It allows packet regeneration from capture (.pcap) files onto the live network. It simulates network traffic well for firewall testing and simplified the process. It runs on FreeBSD, Linux and windows. It has features that allows sending packets at custom set speed or line rates in Mbps. This helped testing the filtering capability on the rule-sets with regard to connection handling and tear-down.

TCPReplay[4]

Developed by Aaron Turner, this Unix based tool gives the ability to use previously captured traffic in pcap format to test the firewall. It allows for header rewriting and traffic

[1]http://nemesis.sourceforge.net/
[2]http://www.hping.org/
[3]http://bittwist.sourceforge.net/
[4]http://tcpreplay.synfin.net/wiki/tcpreplay

classification before playing the traffic back onto the network. The header rewriting feature was not used as test cases demanded same packet structure over naive and optimized rule-sets of firewall.

3.4 Traffic Traversal

This could be said to the critical part of this research. Everything from firewall placement, statistics collection, tool functionality to the actual optimization of the rule set depended on the kind of traffic used and how it moved in the virtual network and the firewall in particular. This part explains how the test setup provided traffic movement for the filtering tests and optimization conducted.

.

3.4.1 Captured Packets (.pcap)

The approach used to generate test traffic was sending traffic from pcap capture files over a bridged interface on the firewall. This enhanced the statistics and tested the firewall's filtering performance using the unoptimized and optimized rule-sets. The pcap files used were taken from the network security web sources[5] and [6].Use of these packet files improved testing requirements of using huge amounts of real network traffic of various protocol composition as needed for firewall filtering testing. The bridged configuration gave two unique traffic flows (i) protected -> unprotected (ii) unprotected -> protected which is vital for conclusive inspection investigation as both directions apply the rule set. Shown below is the movement of traffic in the test environment. It should be noted that crafted packets were only used to test for false positives and negatives to check how optimization handled allowed and denied traffic.

[5]http://wiki.wireshark.org/SampleCaptures#Sample_Captures
[6]https://www.evilfingers.com/repository/pcaps.php

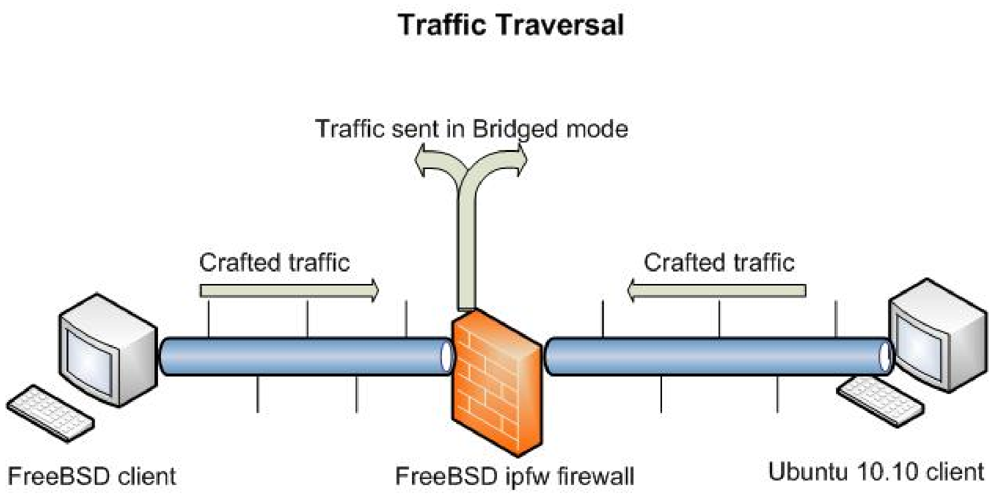


Figure 3.3: Traffic Traversal

3.4.2 Traffic Activity Logging

Traffic logging is a mechanism of recording firewall traffic related activity. It is very vital as it provides a way of detecting attacks and is the best source of information about what happened when an attack succeeds. It is also a sure way of knowing whether a firewall is functioning as expected from its configuration[28]. The firewall tested here,*ipfw*, has this as a kernel logging module that gathers traffic filtering patterns, for rules with the log key word set. General traversal of packets through the firewall can be viewed by issuing the command *ipfw show* at the command line. The logging module is pre-configured but needs to be enabled and the *log limit* defined by adding the following lines to the configuration file , */etc/sysctl.conf* :

```
net.inet.ip.fw.verbose=1

net.inet.ip.fw.verbose_limit=10
```

This enables logging and limits log amounts to 10 per rule that has the keyword log. Limiting the log amount is cardinal to prevent Denial of Service attacks from *syslogd* flooding[15]. The count of 10 was used to give good counters of how often packets of a given type came through to make good optimization decisions. This in the real sense impacts on storage as the log file grows with each rule logging its statistics. This limit was adjusted to the default 5 after collecting enough log information to proceed with optimization. The best practice in production environments is to keep log files in more than one place for security purposes.

3.4.3 Traffic Statistics

To capture traffic statistics, logging was enabled to take stock of network activities going through firewall. They were logged to a file created in on the firewall host in */var/log/ipfw/ipfw.log.* Traffic logging provided log counters for packets matching rules that had the *log* keyword specified. The logs captured critical data and fields that were vital for optimization decisions and techniques as outlined below:

1. The date of packet receipt
2. Time of packet receipt in HH:MM:SS.F for hours, minutes, seconds and fractions of a second.
3. Interface name the packet was processed on. e.g Bridge0
4. Group and rule number of the rule matched
5. The action applied. Allow, Deny and those for default deny as the global setting.
6. The addresses, source IP and port and destination IP and port. e.g 192.168.125.131:80 -> 192.168.128.4:1722
7. The protocol carrying the traffic. e.g tcp , udp.
8. The packet length in bytes.
9. Match count per rule

Packet matching statitics for all rules are kept by the ipfw accounting facility dynamically. These can then be viewed by issuing viewing comands:

#ipfw -t list: this lists all the rules with a time stamp of when last the rule was matched.

ipfw show: shows all rules with the packet matching counters associated with each rule.

There are other commands available for viewing traffic and rule related statistics available in ipfw [15].

3.4.4 Packet Inspection

Ipfw firewall enforces the policy according to '*first-match*' semantics: for each new IP connection, the firewall checks the rules one by one, according to their order in the rule set, until it finds a rule that matches the new connection.

3.4.5 Rule Action

The first rule that matches the connection determines the firewall's action: if the first matching rule has an action of *'ALLOW'* then the firewall will allow the connection to continue, and if the rule's action is *'DENY'* then the firewall will discard all the packets belonging to the connection. Once a packet matches a rule, rule set checking for that packet ceases, and the packet is either *ALLOWED* or *DROPPED* according to the action specified in the matched rule.If no rule matches the connection then the firewall uses a default action, which is usually *DENY*. All denied packets were dropped and logged to see how well the firewall performed on handling illegal packets.

3.4.6 Statistics Extraction

Logged traffic statistics were extracted by reading the log file. The counters were also viewed using the *ipfw show* command to see how rules performed. The sample output in Appendix B shows the result obtained after issuing the *ipfw show* command to see if any traffic was going through the firewall.

During statistics collection, counters and logs were reviewed after inspection of pcap files was completed.The output was then saved in text files and log counters reset and log file cleared for logs to start new counts. These text files were then used to make comparisons of traffic flows and general inspection patterns of the firewall over the naive rule sets and used to perform optimization.

3.5 Optimization

Optimization means reducing the rule set so that packet inspection is only done using fewer applicable rules out of a given rule set size. Traditional rule sets are normally large in the number of rules they contain. Most network layer firewalls like ipfw terminate inspection once a mtch is found. However, this is still a performance degrading situation as rules are evaluated sequentially thus inspection termination depends on the position of the rule being matched. The accounting facility of ipfw provides such statistics related to traffic matching patterns and was used to evaluate traffic trends and all packet to rule set relations. There are various anomalies a large rule set intoduces that affect the filtering of the firewall as earlier stated. Redundancy, rule overshadowing, corelation, irelevance and other anomalies are bought about by the changing traffic patterns and network topologies and general security requirements. There are times when a service an enterprise needed becomes obsolete or is just no longer needed. Most times rules for those services are other objects are not removed. These increase the processing load for the firewall as it searches through them even if they will not apply to traffic being inspected.

In this research, optimization was done based on the gathered statistics to come up with an optimal rule set that offers less inspection time. From the statistics, it was noticed that not all rules matched packets. Even some of those that matched packets, performed actions similar to other rules because of being too generic or their matching frequency was too low.

3.6 Optimization Approaches

To achieve the set out objective of reducing the rule set, many techniques were employed after collecting statistics from log files and counters after testing with different *pcap* files. This was an incremental approach in that the rule set was revised after observing changes in the matching patterns. The approaches used to reduce the initial rule set of 37 rules to 13 rules are presented here next. It should be noted that, the initial rule set was written based on a generic Internet security policy and most of it was adapted from the ipfw recommended rule set for network firewalls. Refeer to the appendices for the default and optimized rule sets.

3.6.1 Removing Redundant Rules

Rules observed not to match any packets over a period of time introduced processing overheads and as such were removed to improve performance.

3.6.2 Merging rules

Rules seen to be performing the same matching were merged by writing single replacement rules. This can happen when rules are defined too generic matching the same protocol for example if there no other special fields added to the rule.

3.6.3 Disabling rules

Having started with a generic rule set written for a fully operational network offering most services that filtering was configured for, tests showed that not all services received packets. Having such rules active in the rule set brings performance degradation because they are evaluated for packets they don not match. These were disabled on the assupmption that the services they were written for may become available later. This was done as principle that ipfw will skip such rules and when their intended traffic does show that it is being dropped in the logs, then they can be enabled.

3.6.4 Changing Rule Priority

Some rules were expected to be matched more than others after reviewing the matching patterns of the packets over time. This was done by placing such rules at the beginning of the rule set.

3.6.5 Removing Overshadowed Rules

These are rules that never match packets either because earlier rules match traffic or there are higher priority rules matching packets earlier. This is a fair decision though it is a challenging thing to do because these rules might later be needed if their predecessors are removed or new traffic that matches them comes through. These rules were removed to optimize the rule set.

3.6.6 Using Skipto function

This function allows setting next rule to be checked if the current rule is not matched. It was based on matching frequencies and rule associative properties. It also works in checking other criteria needed as an addition security check to the previous rule matched.

3.6.7 Rule Re-sequencing

Based on matching frequencies, rules are ordered with the highly matched one being more favored. This was based on expectations that similar packet flow patterns will come through.

3.7 Summary

Packet filtering requires rules developed from a security policy that defines how the organization uses it's computer network and how they interact with other networks of different trust levels. Firewalls could be implemented locally in the network segmenting subnets that should not have access to certain types of data across subnets. These rules are sequentially evaluated against a packet until a match is found. The testing environment described here used FreeBSD ipfw and other tools to transmit traffic and get statistics used in optimizing the rule set. The optimization was done based on these statistics. Care must be taken to ensure that optimizing does not open up the system to traffic that should be denied or lock it from receiving expected traffic.

Chapter 4

OptAid :Rule set Optimizer

Having observed traffic behavior and conducted optimization, this researcher proceeded to the second objective of coming up with a tool that can be used to aid network administrators in optimizing firewall rule sets. Design attention focused on the principle of the operation of firewalls, the traffic characters, transmission mechanisms, protocol functionality among others.

4.1 Design

The tool was designed to analyzes a firewall's filtering patterns based on the collected traffic statistics. These statistics were to be collected and used to determine a recommended action to the administrator. OptAid functionality design features were to provide optimization advice and the second key feature was to adapt a rule set for traffic from the unoptimized rule set. The first feature was to be achieved by implementing algorithms that manipulate collected statistics for the tool to:

- recommend rule re-sequencing
- suggest rules to be added to skip commands
- show rules not matching packets
- suggest dependency ordering
- allow for viewing rule counts dynamically through a graphical interface

The second feature of coming up with a sub rule set was to manipulate the most recent traffic statistics and use them to reduce the rule set dynamically - NP Hard[19]. This sub rule set is then used to inspect inbound traffic. This is to cater for long transmission flows for sessions lasting greater than or equal to a defined threshold. This dynamic algorithm should be triggered by the threshold and perform the following:

- check rules that matched the starting packets of the flow from the statistics
- compile a rule set based on those matched rules
- apply that rule set to the rest of the flow
- terminate once the close connection packet is received. TCP FIN

4.2 Implementation

OptAid was to be implemented using the functionality given in the design section. The diagram below illustrates the overview operational concept of the tool.

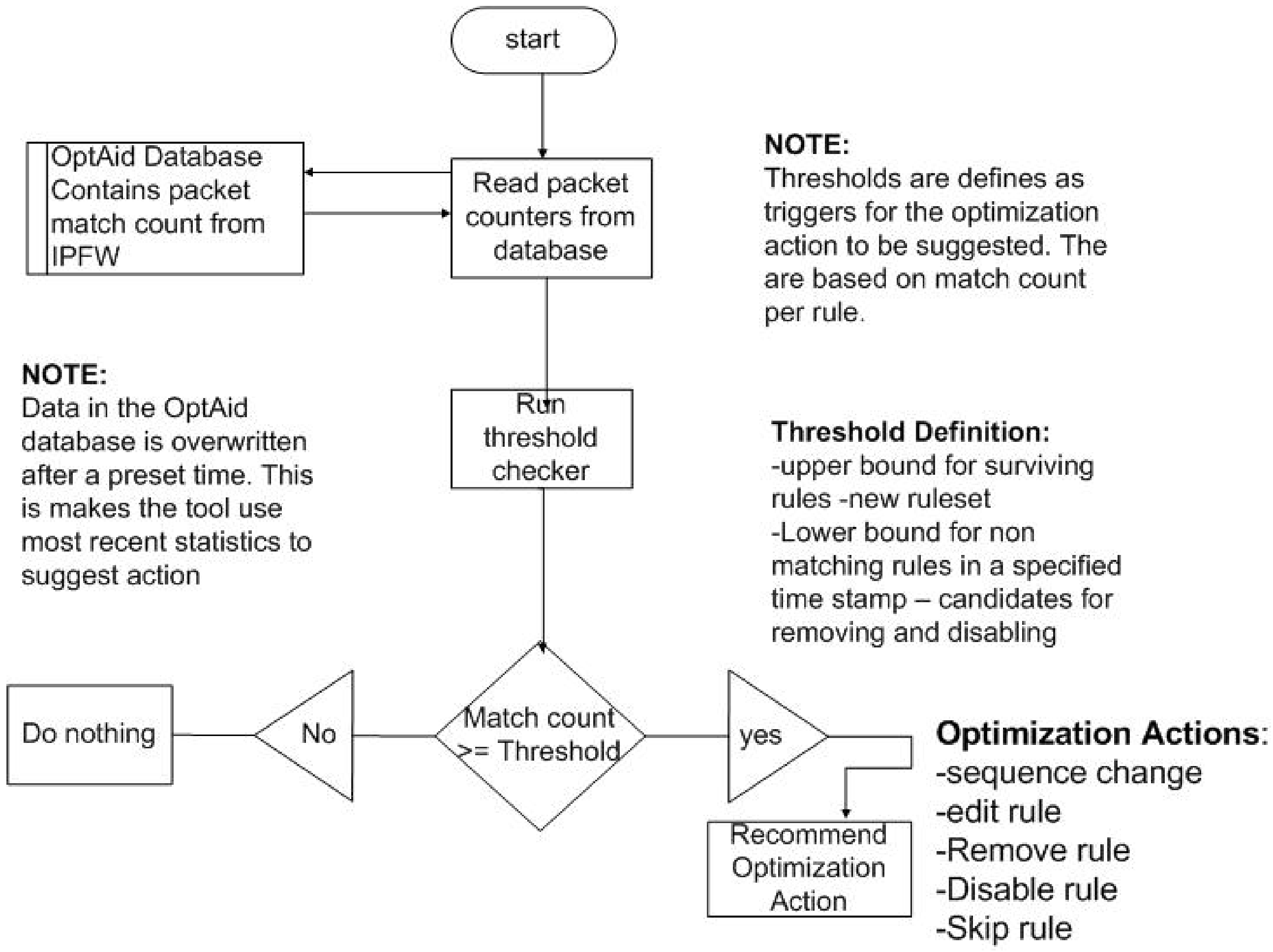

start
OptAid Database Contains packet match count from IPFW
Read packet counters from database
NOTE:
Thresholds are defines as triggers for the optimization action to be suggested. The are based on match count per rule.
NOTE:
Data in the OptAid database is overwritten after a preset time. This is makes the tool use most recent statistics to suggest action
Run threshold checker
Threshold Definition:
-upper bound for surviving rules -new ruleset
-Lower bound for non matching rules in a specified time stamp – candidates for removing and disabling
Do nothing
No
Match count >= Threshold
yes
Recommend Optimization Action
Optimization Actions:
-sequence change
-edit rule
-Remove rule
-Disable rule
-Skip rule

Figure 4.1: OptAid Tool Design

The tool uses data stored in an a database section that is updated by statistics from the ipfw counters of packet matching patterns. When the user starts the tool from the programs menu, it invokes a reader method that reads count statistics from the database and passes them to the threshold checker. The threshold checker runs and compares which rules have met the set triggering thresholds for recommending actions to be done on the rule. The recommendations are presented to the administrator in a graphical interface.

4.3 Issues Faced

The design and implementation of OptAid did not go far because of issues that needed to be addressed and well understood before proceeding. These problems had functionality and performance implications and as such needed to be well highlighted and dealt with. Key issues experienced and are currently receiving attention to resolve them are:

1. Statistics aggregation vs. dynamic nature of traffic: traffic being dynamic by nature changes patterns so often. This poses challenges when collecting statistics on which to base recommendations. Using an offline database that updates periodically could make the recommendations invalid as traffic changes.

2. Defining triggering counter thresholds for ever changing traffic: when thresholds are defined for recommending actions, there is no guarantee that they will reflect what will happen next. This means actions might be of little use at optimizing stage especially using an offline mechanism relying on periodic updates.

3. Should rules be discarded completely: if a recommendation to discard rules is given, should a rule be taken out of the rule set completely. This proved to be a complex problem as traffic coming next might need removed rules.

4. How much overhead does all this introduce: ascertaining how much processing overheads the sending of statistics to the OptAid database was not easy. Also how often those updates had to be run could not be well defined because of network traffic dynamics.

With respect to the second consideration of creating a sub rule set dynamically from the bigger set for traffic flows, there are inherent problems that became apparent and could not be sorted by the time of this reporting. Some of these problems are outlined here:

1. Processing complications: with regard to picking rules for oncoming traffic, what to use to determine what flow is oncoming was not resolved. Any dynamic way of coming up with a sub rule set must take care of this. This research is continuing exploring this factor.

2. What flows to adapt for: questions need to be asked relating to which flows to adapt for; the most recent flow, or those with dynamic state kept?

3. Issues with rule numbering for re-sequencing: once it is decided to re order rules based on matching frequency, this may complicate processing the rule set and affect general filtering negatively. This is because, frequently, some rules will have trigger thresholds higher than others. These are expected to be matched faster by placing them upfront. This overwhelms the firewall with rule set processing instead of packet filtering.

4. How does this sub-rule set get applied to traffic: if this sub rule set is created, how can it be used to general traffic arriving or leaving the firewall in real-time? This is yet to be understood.

4.4 Summary

It has been learned from the design and implementation that suggestions that packet filtering and optimizing a rule set dynamically in particular is NP-Hard. This is because the many issues that have to be considered bordering on performance and efficiency of the chosen algorithms to be implemented to do this. It is also generally hard to determine oncoming traffic flows given the dynamic nature of traffic. This makes choices of processing modes hard; offline processing for example, though can be argued to work well for suggesting actions on rules, it can be done in vain as traffic changes each time and statics become old in a very short time.

Chapter 5

Performance Benchmarking Tests

Firewalls have become increasingly complex, evolving from traditional filtering capabilities to offering application-aware processing of several Internet protocols. This is becuase of the upsurge in Internet applications that have to be controlled through firewalls. They are now being implemented as hybrid platforms for offering next-generation application-aware inspection capabilities that include:

- ***Network security*** – Application-aware content inspection, access protocols including IPSec, 802.1x, RADIUS, intrusion prevention capabilities and DDoS attack mitigation
- ***Web security*** – Intelligent HTTP/URL content inspection to fend off "buffer overflow" attacks, virus and spyware prevention for the web, phishing attacks, and to validate protocol compliance by ensuring the requests are not malformed
- ***Email security*** – Protection from spam, virus and phishing attacks, which overwhelms networks significantly with wasteful traffic
- ***Next-generation*** – Support for IPv6, quality-of-service, voice (e.g., SIP) and video streaming.

This has given rise to complex firewall testing mechanisms and approaches so that all such issues covered in a given firewall implementation are catered for. The focus of the work conducted in this research was on network layer packet filtering. As such performance improvement tests done centered around the following network layer related metrics detailed below.

5.1 Performance Tests

These tests were done to determine the throughput of the firewall using a default and optimized rule set packets traversing the firewall. It should be noted that there is no standard testing methods and tests prescribed for firewall testing. However, there are recommended guidelines proposed by the IETF Network Working Group in[10]. Tests conducted are case specific; dependent on what metrics the experiment is measuring as being performance contributors [40]. The tests conducted here measured the time it took for a packet of given size to be inspected by the firewall using the naive and optimized rule-sets. The timing observations were then recorded and are discussed in the context of the metrics presented in section 5.2

5.1.1 Setup Parameters

As a test case, packet size and test duration were used to compare filtering performance of both rule sets. These two provided a good comparison basis given that all other metrics are somewhat dependent on them.

- Packet Size - the number of bytes in the packets going through the firewall. This is regardless of any link layer or protocol specific headers or check-sums.
- Test Duration - time in seconds taken for a packet of given size in bytes to be transmitted. The contents of the pcap file in this case are sent through and get inspected on a rule set.

5.1.2 Procedure

As earlier explained in previous sections, the test environment sent IP packets to the firewall at a constant rate. The packet files played through firewall *Bridged interface* were

.pcap files of different protocols and sizes downloaded from web network sources [1]

The tests were performed using different packet sizes to see the difference in firewall performance under the same rule-set and general firewall environmental variables. This involved using the default naive rule set and the optimized one to see the gain in filtering time on the same packet sizes.

5.2 Measured Metrics

5.2.1 Connection establishment

This test defines a single TCP connection between two end hosts.It measures the time it takes for a TCP connection establishment (3-way handshake) to be set up between the DUT and the client seeking to transmit packets to it.

5.2.2 Forwarding Rate

A measure of packets forwarded expressed in either bits per second or packets per second. This is based on the offered load and covers IP packets, header and payload.

5.2.3 Connections per second

This measured the rate at which new TCP connections were initiated per second through the firewall. Processing time is directly dependent on how many requests a firewall is servicing per second given the amount of requests sent to it in a second. This a good indicator of how much processing gain optimizing gives. It also helps isolate any bottlenecks on the network and provides a good basis for tuning the firewall to operate optimally.

[1]http://wiki.wireshark.org/SampleCaptures#Sample_Captures

5.2.4 IP Throughput

Measured the maximum offered load at which no packet loss is detected for legitimate traffic. The unit of measurement used was bits per second and packets per second based on the pcap files played in the firewall. Clients crafted protocol specific packets and sent them to the firewall. The less rules used the higher the throughput.

5.2.5 Connection Tear-down

This measured the time it took for the firewall to react to transmission complete signals from end nodes. A heavily loaded rule set showed long time processing in connection closures because of so many dependencies like fragments being checked.

5.2.6 Legal Traffic

The firewall was tested to see how well it handled legitimate traffic. This was done to see if limiting the rule set would deny the network of necessary traffic. It also measured if any critical services that network and the firewall host itself might need for operation will be available. This might be termed as the false negatives test.

5.2.7 Illegal Traffic

Illegal traffic handling tested the firewall's ability to block what was configured to be blocked. The essence of having a firewall is to block unwanted traffic from entering a network. Therefore testing to see if any illegal traffic is finding its way through is critical. It not only checks how effective optimization is but also how secure the system is after optimizing. If it is found that illegal traffic is being allowed through the firewall, the rule set should be checked for configuration flaws. This can happen due to rule set misconfiguration or specifying rules that are too generic. It can also be as a result of certain rules not having complementary or child rules that perform extended inspection especially for firewalls that perform deep packet inspection.

5.3 Summary

Performance testing is the best way to determine the significance of any approach taken.Metrics and tests measured for performance gain investigation and benchmarking tests carried out to test firewall on both rule sets have been described. Performance metrics used here are not exhaustive of the many that can be employed to test firewalls for other security requirements. The researcher performed the ones explained above as they related to the investigation of determining an increase in throughput by limiting the firewall rule set size. The log file shows dropped or denied packets, their source and destination IP and what port they were attempting to establish connections on. This accounts for failed connections, and gave a way of seeing how the filtering actions were working.

Chapter 6

Performance Test Results

The test environment, approaches used to perform rule set optimization and the tests conducted to measure performance have been discussed in preceding chapters. This section presents the results of those tests and how they relate to the metrics explained earlier. It starts by giving the high level view of how packets were inspected by the firewall by showing matching patterns captured using a tool. Then comparative results for tests done on both rule sets are presented and analysed in the following parts leading to the summary. They represent performance tests carried out using both the default and optimized rule sets in an augmented approach as reported graphically here under.

6.1 Results

The tests conducted on the rule sets were augmented in that a single transmission done measured metrics that were adopted as performance variance determinants in this research. Figure 6.1 reported here under shows rule set packet matching patterns using *Ipfw-Graph.* Ipfw-Graph captures the bytes going through *ipfw* firewall and graphically displays allowed, denied and all packets but does not show the time taken to inspect. *Ipfw-graph* was run from the start of transmission to the end observing the time when the end of the packet size was reached manually. The views below show matching propotions of denied and allowed packets as an example view of how traffic was being treated by the firewall. Below is the output from *ipfw-graph.*

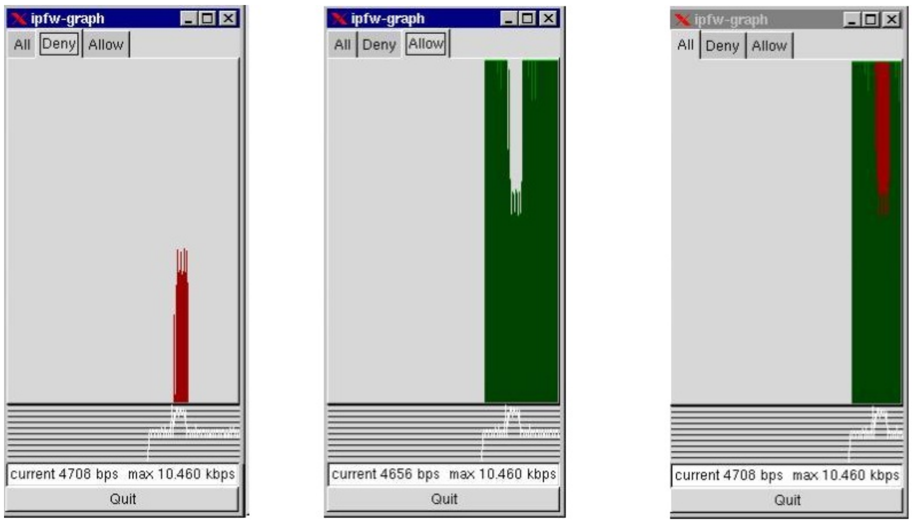

ipfw-graph
All
Deny
Allow
current 4708 bps max 10.460 kbps
Quit
ipfw-graph
All
Deny
Allow
current 4656 bps max 10.460 kbps
Quit
ipfw-graph
All
Deny
Allow
current 4708 bps max 10.460 kbps
Quit

Figure 6.1: Ipfw-graph Reporting Format

The snapshots seen above were taken on ipfw-graph output showing a concentration of filtering statitics.

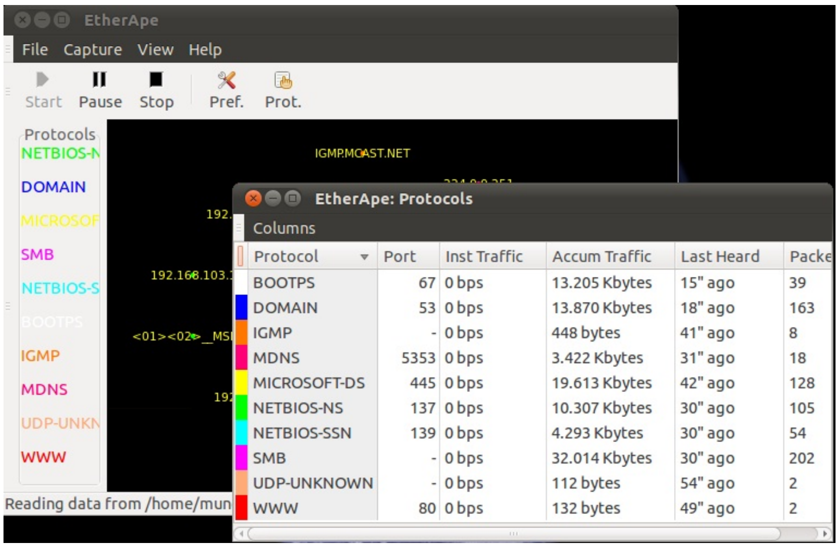
Deny: denied packets are shown in red under then **Deny** button.

Allow: allowed packets are showed in gree under the **Allow** button.

All: shows a combination of both allowed and denied packets.

As can be seen, there is no timing captured, therefore manual timing on a packet of 1500 Bytes was done on both rule sets. This file was a multi-protocol composed pcap downloaded from [1] as can be seen from the packet replay snapshot in Figure 6.2 below showing partial output of the long protocol list that the 1500 Byte packet contained.

[1]http://wiki.wireshark.org/SampleCaptures#Sample_Captures



Protocol	Port	Inst Traffic	Accum Traffic	Last Heard	Packe
BOOTPS	67	0 bps	13.205 Kbytes	15" ago	39
DOMAIN	53	0 bps	13.870 Kbytes	18" ago	163
IGMP	-	0 bps	448 bytes	41" ago	8
MDNS	5353	0 bps	3.422 Kbytes	31" ago	18
MICROSOFT-DS	445	0 bps	19.613 Kbytes	42" ago	128
NETBIOS-NS	137	0 bps	10.307 Kbytes	30" ago	105
NETBIOS-SSN	139	0 bps	4.293 Kbytes	30" ago	54
SMB	-	0 bps	32.014 Kbytes	30" ago	202
UDP-UNKNOWN	-	0 bps	112 bytes	54" ago	2
WWW	80	0 bps	132 bytes	49" ago	2

Figure 6.2: Protocol Composition

Figure 6.3 shows a comparison of filtering timings for both rule sets on a source packet of the same size. This time taken was manually recorded from start of transmission to completion as viewed in *ipfw-graph*. This was done because *ipfw-graph* does not provide a timer view for the transmission against packet size. The manual timings were then graphed for both rule sets to show the performance comparison between the two rule sets. The results graphed below are based on a 1500 Bytes pcap file and sums up the results reporting by showing the bytes against time taken to inspect them on both rule sets.

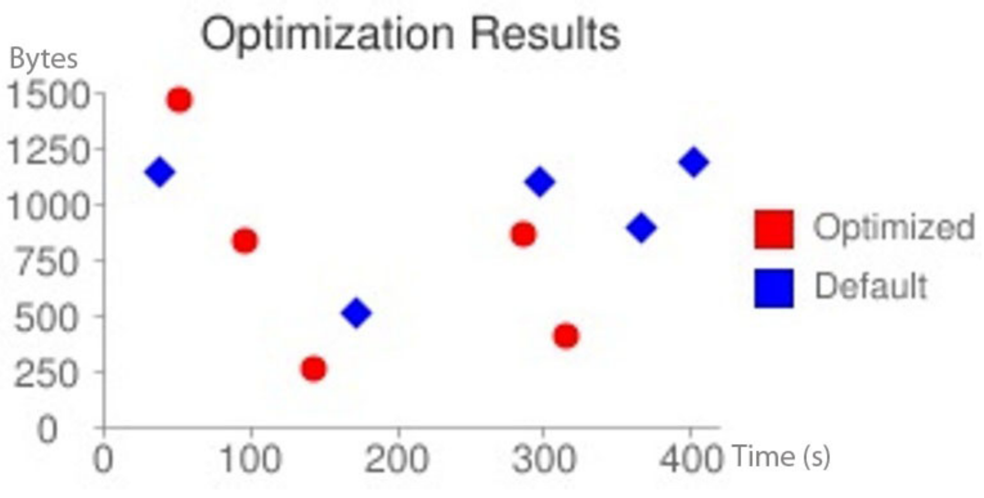


Figure 6.3: Performance Results

6.1.1 Default rule-set (Blue)

The default rule set contained 37 rules. Other test environment factors were kept the same; packet size, services enabled, processor capacity, memory, I/O devices and any other underlying hardware. The graph above presents the time taken to fully inspect a 1500 Bytes *.pcap* file of various protocols.

6.1.2 Optimized Rule-set (Red)

The same *.pcap* file used on the default or un-optimized rule set was used to test the optimized rule set that contained 13 rules. Other environment variables mentioned inthe previous section were kept the same as they were when testing using the default rule set. Refer to section 3.6 on how optimization was performed.

The results in the graph above show that the optimized rule set finished inspecting the 1500 Bytes early enough at around 320 seconds as opposed to the naive rule set that went on beyond 400 seconds.

6.2 Results Discussion

From the results presented in 6.3, it can be observed that optimizing a rule set offers a speed up in inspection time.This result takes into account all measured metrics as already mentioned in section 5.2 and explained here next. Having kept all other factors constant when testing with both rule sets, the throughput contributors or measures discussed here next are purely based on the rule set size. On both rule sets, legal and illegal traffic handling tests were conducted.

6.2.1 Connection establishment

Connection establishment took longer in the case of the default rule set because it applied more rules to each packet and checked them until a match was found. This delayed new connection establishment and contributed to the firewall taking longer to finish inspecting the file. It can then be advanced here that the optimized rule set did perform connection establishment faster. Packets inspected are sent over a connection established between two end points, source and destination. That being the case, connection establishment is a indeed a factor in filtering speed up and does depend on how many rules packets are being evaluated through.

6.2.2 Forwarding Rate

Packets in the pcap sent through the different rule sets with other variables kept same differed in forwarding date. The default rule set took longer inspecting packets, hence forwarding was affected. The optimized one performed better because packets became available for forwarding earlier than on the default rule set. This is the case in production networks where the firewall protecting a network inspects inbound and outbound packets at the border and forwards them to receiving nodes.

6.2.3 Connections per second

The faster the inspection the faster the connection tear-down and consequently more connection were processed per second on the optimized rule set. This contributed to carrying through the file 1500 file size faster than on the default rule set. This is closely related to connection establishment.

6.2.4 IP Throughput

The bits per second sent through were less on the default rule set and better on the optimized rule set. The general observation on the timings shows faster completion on the optimized rule set. This is because more bits are carried through per second than on the default rule set. This signifies the need for better filtering mechanisms using less rules.

6.2.5 Connection Tear-down

Closing connections happens when the transmission of a flow is complete.The optimized rule set closed connections much quicker hence the shorter time taken to complete inspecting as it gave room for new connections to be established faster.

6.2.6 Legal Traffic

Both rule sets gave positive results, the logs showed no legal traffic dropped after running pcap files on both naive and optimized rule sets. It should however be noted that, this needs serious caution and continuous checking to be sure that nothing is falsely denied and all services are functioning as expected.

6.2.7 Illegal Traffic

The effectiveness of handling illegal packets by not allowing false positives was tested and both rule sets reported good results. They blocked traffic explicitly set to be denied entry. This is another test that should be done often to ascertain the safety of the network. Dangers now exist with protocol encapsulation and other tunneling mechanisms that might shield denied traffic.

6.3 Summary

Given a network environment with all variables kept at bay and traffic changing as it does, there can be filtering performance gained by optimizing a rule set by the many approaches available and those that be seen necessary. Optimizing reduces the number of rules in a rule set thereby reducing the search space for packet to rule evaluation. This is however an involving exercise and needs to be done with caution to avoid closing out the network from communicating or from being open to attack.

Chapter 7

Conclusion

This section sums up the work done in this research. A look at the work presented in each chapter is taken and the set out goals revisited. Packet filtering and rule set optimization are discussed in a summary context to present how they ingredients into investigating performance speed up. Filtering challenges are highlighted alongside some issues faced in coming up with an optimization aiding tool. The chapter closes by focusing on some considerations being explored to expand on the work done in this research.

7.1 Summary

The concept of firewalls has been presented and described from its inception and history in the beginning chapters. The existing types were discussed in the context of their implementation, stages of filtering, how much inspection they each performed and how they filter packets.

The researcher reviewed previous works done on rule set optimization starting from the basics of how firewalls have evolved, how they perform filtering and rules are developed from security policies. This gave a better understanding of core concepts before going on to with the advanced material. A look at some tools that have been proposed and some implemented was taken, focusing on their operation principle and optimization mechanisms used.

An environment in which to perform this research was set up using chosen operating systems and tools. This provided the test bed in which tests on developed rule sets were conducted. Based on the results obtained from log files and packet counters, the original rule set of 37 rules was used to come with a sub rule set that contained 13 rules. Filtering tests were then performed using both rule sets (default and optimized) rules sets using the same packet sources of the same protocol composition and their inspection time noted.

Metrics measured in checking performance gain were chosen as contributors to the overall throughput. They do not represent an exhaustive list of tests that can be done as there is no recommended way of testing firewalls. Tests done for firewall performance depend on the factor being measured.

This research used a single testing approach that measured packet size against inspection completion time and discussed filtering improvements around adopted metrics taken as throughput contributors being:

- Connection establishment
- Forwarding rate
- connections per second
- IP throughput
- Connection tear-down
- Legal traffic handling
- Illegal traffic handling.

Investigating performance gain was achieved and can be seen from the presented results in the results chapter. Work on designing and implementing the optimization guide tool (OptAid) was not completed due to much work and issues that needed to be solved before proceeding. There were several issues encountered that hampered progress on this goal. There is however, a design presented that will be worked on in future to deliver the tool. Also possible extensions to improve the design have been outlined for further investigation.

7.2 Problem statement and Goals Revisited

The problem domain this research set out to address is based on the evidence that not all rules in a firewall rule set match every packet inspected. Considering the dynamic nature of traffic, better filtering approaches that measure up to modern traffic demands have to be worked out. Rule sets grow large in number and are often changed by different network administrators according to network demands. This introduces anomalies and other rule inconsistencies among them rule over-shadowing, rule redundancy and other problems in the firewall configuration.

Firewalls inspect packets against a set of rules sequentially to find one that matches a packet and applies the action set on it. This sequential evaluation of the rule set to find a matching rule is a major negative contributor to firewall performance as all rule inspected do not match a packet they are evaluated for. Most firewalls exit on the match of a rule but that does not help alleviate the performance drawback as the position of the rule in a rule set determines how fast it will be matched.

Given the improvements in both hardware, network transmission techiniques, protocol designs and network speeds, there is need for better packet filtering mechanisms. Firewalls are key components in network security as they are deployed between two networks of different trust levels to filter packets or within networks to limit data and access to other network resources. From other reviews and the findings of this research, it is evident that not all rules in a rule set match a packet and network traffic is not static. Firewalls are therefore posed with increased challenges on their filtering performance.

Based on the problem statement discussed here, this research set out two objectives with the first one being the basis for the second one:

- to investigate if reducing the firewall rule set size offers a speed up in packet inspection time.
- to develop a tool that will aid network administrators in optimizing rule sets.

From the results obatined and anlysed in the results section, the first objective of determining whether optimization offers inspection improvements was met. It was found that a smaller rule set finished inspecting a the same size of packets faster then the original, unoptimized rule set. Work on the designing of the tool as the second objective was slowed by issues faced with the design of the tool. Functionality demands that were set to be considered for the design need further investigation before the actual implementation is done.

7.3 Conclusion

Packet filtering optimization has been investigated by many researchers. However, there is still need for better directions and approaches to be taken given the continued changes in network technologies. Improved filtering mechanisms will enable firewalls that perform their filtering based on rules to use the least rules to keep up with increasing speeds of modern networks. Packet filtering is generally an NP Hard problem because of the complexity and mostly non-deterministic nature of traffic flows. The work done in this research has demonstrated that firewall rule set optimization does noticeably increase throughput. An optimized rule set means less rules are checked against a packet and as a result matches are found faster thereby giving a speed up in filtering performance. Better filtering mechanisms through rule set optimization bring about better Quality of Service. Ways of implementing this have been explored and tested in the optimization approaches used with early packet matching for frequently matched rules, rule merging for similar ones and editing those observed to be too generic as some of the approaches. Automating this in a tool so that the rule set is adjusted dynamically was embarked on to add efficiency to the firewall - NP Hard. It has however been investigated and work is continuing on the tool, OptAid. The idea of the tool is to aid network administrators in optimizing rule sets; dynamic rule set adaptation is a possible extension on the functionality of the tool. The tool removes the need for the network administrator to manually reconfigure rules. Manual configuration is not only a cost on time but may introduce flaws considering the number of rules being adjusted. Typical rule sets in production networks average above a thousand and are often written by different administrators as the security policies and topologies changed over time. The growth of the rule set not only increases configuration complexity but adds the obvious processing overheads and use of memory on the firewall device. From the work done in this research, rule matching frequency for firewall rules can be qualified as a critical property for optimization. This researcher proposes using this property to create a dynamic way of adjusting the rule set by augmenting these statistics dynamically using a threshold mechanism to pick a sub rule set from the original rule set.

7.4 Future Work and Extensions

Having explored and investigated performance gain through optimization fully, the rule set optimization guide tool OptAid will be developed after resolving pending issues of performance consideration in the design and implementation.

The ideal extension will be adding a feature that creates a sub rule set for oncoming traffic and applies it to perform filtering. There are pending problems with rule selection and choosing which flows to adapt for. Once these are worked on and processing versus performance trade offs are well understood and seen viable. This will be a good feature and a breakthrough in packet filtering.

References

[1] Acharyaý, S., Ablizý, M., Millsý, B., Znatiý, T. F., Wangü, J., Geü, Z., and Greenbergü, A. Optwall: A hierarchical traffic-aware firewall. *Network and Distributed Security Symposium* (2007). Accessed 18 June 2010.

[2] Acharyaý, S., Wangü, J., Geü, Z., Znatiý, T. F., and Greenbergü, A. Traffic-aware firewall optimization strategies. *IEEE International Conference* (June 2006). Accessed 15 August 2010.

[3] Adolfo Rodriguez, John Gatrell, J. K. R. P. *TCP/IP Tutorial and Technical Overview.* IBM Redbooks, 2001.

[4] Al-shaer, E. S., Hamed, H. H., Al-shaer, E. S., and Hamed, H. H. Design and implementation of firewall policy advisor tools. Tech. rep., 2002. Accessed on 23 June 2010.

[5] And, M. L. Firewall security: Policies, testing and performance evaluation. Accssed 14 June 2010.

[6] Arkko, J., Vogt, C., and Haddad, W. Enhanced route optimization for mobile ipv6. RFC 4866 (Proposed Standard), May 2007. Accessed on 24 June 2010.

[7] B. Chapman, E.D. Zwicky, S. *Building Internet Firewalls.* O'Reilly & Associates, 2000.

[8] Chapman, D. B. Network (in)security through ip packet filtering. In *In USENIX Security Symposium III Proceedings* (1992), USENIX Association, pp. 63–76. Accessed 8 June 2010.

[9] Computers, A. *MacOSXServer 2009*, 2009.

[10] D Newman B Hickman, S Tadjudin, T. M. Benchmarking methodology for firewall performance, 2003. Accessed on 18 April 2010.

[11] Ehab S Al-Shaer, H. H. H. Modeling and management of firewall policies. *IEEE Network and Management* (April 2004). Accessed 14 May 2010.

[12] Ehlert, S., Zhang, G., and Magedanz, T. Increasing sip firewall performance by ruleset size limitation. In *PIMRC* (2008), pp. 1–6. Accessed 12 May 2010.

[13] Eronen, P., and Zitting, J. An expert system for analyzing firewall rules. Accessed on 19 March 2010.

[14] FreeBSD. *Firewalls.* FreeBSD Document Project, February 2010, ch. 30, p. 777. Accessed 16 May 2010.

[15] FreeBSD, F. *FreeBSD Handbook 2010.* FreeBSD Document Project, February 2010.

[16] Fulp, E. W. Optimization of network firewall policies using directed acyclic graphs. Accessed 4 July 2010.

[17] Grote, A., Funke, R., and Heiss, H.-U. Performance evaluation of firewalls in gigabit-networks. In *Proc. 1999 Symposium on Performance Evaluation of Computer and Telecommunication Systems. Chicago, Society for Computer Simulation* (1999). Accessed 27 Augsut 2010.

[18] Hall, B. B. *Beej's Guide To Network Programming Using Internet Sockets.* USENIX, 2005.

[19] Hamed, H., and Al-Shaer, E. Dynamic rule-ordering optimization for high-speed firewall filtering. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security* (New York, NY, USA, 2006), ACM, pp. 332–342. Accessed 24 April 2010.

[20] Hamed, H., El-atawy, A., and Al-shaer, E. Adaptive statistical optimization techniques for firewall packet filtering. Accessed on 11 April 2010.

[21] Hazelhurst, S. A proposal for dynamic access lists for tcp/ip packet filering. Accessed on 13 July 2010.

[22] Hunt, R., and Verwoerd, T. Reactive firewalls - a new technique. *Computer Communications, Elsevier, U.K. Vol 26,No 12,* (2003). Accessed 19 Augsut 2010.

[23] Inc, V. *VMware Workstation 7.0.* VMware Inc, 3401 Hillview Ave, 2009.

[24] Karen Scarfone, P. H. *NIST: Guidelines on Firewalls and Firewall Policy.* 2009.

[25] KATIC, T., AND PALE, P. Optimization of firewall rules. IEEE Information Technology Interfaces.

[26] KOLEHMAINEN, A. Optimizing firewall performance. *Seminar on Internetworking* (2007). Accessed 6 May 2010.

[27] LAVIGNE, D. *BSD Hacks.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2004.

[28] MAIOLINI, G., NICOTRA, A., TORNARI, P., AND BAIOCCHI, A. Automated framework for policy optimization in firewalls and security gateways. *Information Assurance and Security* (2009). Accessed on 16 March 2010.

[29] MARMORSTEIN, R. A tool for automated iptables firewall analysis. In *Freenix Track, USENIX Annual Technical Conference* (2005), pp. 71–82. Accessed on 3 April 2010.

[30] MÁRTON ILLÉS, T. B. The evolution of the firewall. June 2006.

[31] PACO HOPE, YANEK KORFF, B. P. *Mastering FreeBSD and OpenBSD Security.* O'Reilly, 2005.

[32] REYNOLDS, J., AND POSTEL, J. Assigned numbers. Accessed on 7 April 2010.

[33] SECURITY, A. Firewall cleanup and optimization. Accessed 28 May 2010.

[34] SHIMONSKI R.J, SHINDER D.L, D. S. T. *Best Damn Firewall Book Period.* Syngress, 2003.

[35] STEVE SUEHRING, R. Z. *Linux Firewalls, Third Edition.* Sams Publishing, 2005.

[36] STEWART, J. M., TITTEL, E., AND CHAPPLE, M. *CISSP: Certified Information Systems Security Professional Study Guide.* SYBEX Inc., Alameda, CA, USA, 2008.

[37] SUNSHADOWZ. Types of firewals. http://www.sunshadowz.com/articles/firewalls, May 2010. Accessed 11 August 2010.

[38] SYSTEMS, C. *Evolution of the firewall industry.* Cisco Press, 2002.

[39] TRAINING, I. Firewalls: Overview. vol. 11/23/2009. Accessed 17 May 2010.

[40] WACK, J., CUTLER, K., AND POLE, J. *Guidelines on Firewalls and Firewall Policy.* U.S. Government Print Office, January 2007.

[41] WHITMAN, M. *Principles of Information Security 2nd Edition.* Course Technology, 2004.

[42] William R. Cheswick, Steven M. Bellovin, A. D. R. *Firewalls and Internet Security : Repelling the Wily Hacker.* Addison Wesley, 2003.

[43] Zhao, L., Shimae, A., and Nagamochi, H. Linear-tree rule structure for firewall optimization. In *CIIT '07: The Sixth IASTED International Conference on Communications, Internet, and Information Technology* (Anaheim, CA, USA, 2007), ACTA Press, pp. 67–72. Accessed 30 July 2010.

Appendix A

Default Rule set

The default or naive rule set here under contains rules that were seen necessary to secure the network and offer services. They were continually reviewed after testing the filtering capacity using files and the packet matching statistics used to come up eith the next rule set.

```
############# default ruleset######### #

!/bin/sh

#Flush all the rules, quietly

ipfw -q -f flush

#rules command prefix

cmd="ipfw -q add"

pif="bridge0"

#Localhost rules

$cmd 100 allow all from any to any via lo0

# Prevent any traffic to 127.0.0.1, localhost spoofing

$cmd 110 deny log all from any to 127.0.0.0/8

$cmd 120 deny log all from any to 127.0.0.0/8

#Testing rules.These rules allow ALL traffic to pass disabling any subsequent rules
```

```
#$cmd 140 allow log logamount 500 tcp from any to any

#$cmd 150 allow log logamount 500 udp from any to any

#Start dynamic state filtering

$cmd 200 check-state

#Filter out fragmented packets that

$cmd 210 deny all from any to an frag in via $pif

#Block ACK packets that are not matched dynamically

$cmd 220 deny tcp from any to any established in via $pif

#Outbound web server ports, port 80 and 443 (SSL)

$cmd 230 allow tcp from any to any 80 out via $pif setup keep-state

$cmd 240 allow tcp from any to any 443 out via $pif setup keep-state

#nameserver ports

#DNS SERVICE DISABLED

# traceroute out allowed for diagnostics

$cmd 270 allow udp from me to any 33434-33525 out via $pif keep-state

$cmd 271 allow udp from any to any 33434-33525 in via $pif keep-state

#inbound traceroute for diagnosis if needed

$cmd 280 allow log icmp from any to me icmptypes 3,11 in via $pif keep-state

#outbound ping

$cmd 290 allow icmp from any to any out via $pif keep-state

#DHCP requests from all of the 192.168.46.0 subnet.

$cmd 300 allow udp from any 68 to 192.168.46.0/24 67 out via $pif keep-state

$cmd 310 allow udp from 192.168.46.0/24 67 to any 68 in via $pif keep-state

#Allow ssh out to any host

$cmd 400 allow tcp from any to any 22 out via $pif setup keep-state
```

```
#Only Linux, 192.168.46.140 is allowed ssh to the firewall(me)

$cmd 410 allow tcp from 192.168.46.140 to me 22 setup keep-state

#NTP rule, for setting of time automatically

$cmd 420 allow udp from any to any 123 out via $pif keep-state

$cmd 421 allow tcp from any to any 123 out via $pif setup keep-state

#inbound ping for diagnostic purposes

$cmd 470 allow log icmp from any to me icmptype 0,8 in via $pif keep-state

#Samba rules

$cmd 520 allow tcp from any to any 137,138,139 out via $pif keep-state setup

$cmd 530 allow udp from any to any 137,138,139 out via $pif keep-state

#Microsoft Active Directory Service rules - needed for login

$cmd 560 allow tcp from any to any 3268 out via $pif keep-state setup

$cmd 570 allow udp from any to any 3268 out via $pif keep-state

#Outgoing SMTP to send messages if needed

$cmd 610 allow tcp from any to any 25 out via $pif keep-state setup

# Rules to allow IMP mail to function

$cmd 660 allow tcp from any to any 8444 out via $pif keep-state setup

# Allow RTSP streaming media

$cmd 680 allow tcp from any to any 554 out via $pif keep-state setup

$cmd 681 allow udp from any to any 554 out via $pif keep-state

$cmd 682 allow tcp from any to any 554 in via $pif keep-state setup

$cmd 683 allow udp from any to any 554 in via $pif keep-state

#RSTP uses UDP ports to communicate, and need to be open.

$cmd 684 allow udp from 192.168.46.0/24 6970-32000 to me in via

$pif keep-state
```

```
#Monitor requests

$cmd 700 allow tcp from any to any 8180 out via $pif keep-state setup

#Deny rules to filter out bogus packets

#Multicast packets to ignore

$cmd 700 deny tcp from any to 224.0.0.0/4 in via $pif setup keep-state

$cmd 710 deny ip from any to 224.0.0.0/4 in via $pif keep-state

#External ICMP redirect requests

$cmd 720 deny log icmp from any to any icmptype 5 in via $pif keep-state

#Prevent spoofing attacks, one should never see any traffic to and from me

$cmd 730 deny log ip from me to me in via $pif keep-state

#Prevent ping echo attacks

$cmd 740 deny log icmp from any to me icmptype 0,8 in via $pif keep-state

#Deny any TCP setup requests from the outside world

$cmd 750 deny log tcp from any to any setup in via $pif keep-state

#Default deny rule

$cmd 10000 deny log logamount 500 all from any to any

############ End of rule set #########################
```

Appendix B

Optimized Rule Set

This is the reduced size set of 13 working rules. As can be seen, most rules have been removed from the default rule set. Others are disabled/removed but showing in the rule set. This was done after it became apparent that some services did not receive or send traffic. So they are inactive and not being evaluated against any packets.

```
############# Optimized ruleset######### #

!/bin/sh

#Flush all the rules, quietly

ipfw -q -f flush

#rules command prefix

cmd="ipfw -q add"

pif="bridge0"

#Localhost rules

$cmd 100 allow all from any to any via lo0

# Prevent any traffic to 127.0.0.1, localhost spoofing

$cmd 110 deny log all from any to 127.0.0.0/8

$cmd 120 deny log all from any to 127.0.0.0/8

#Testing rules.These rules allow ALL traffic to pass disabling any subsequent rules
```

```
#$cmd 140 allow log logamount 500 tcp from any to any

#$cmd 150 allow log logamount 500 udp from any to any

#Start dynamic state filtering

$cmd 200 check-state

#Filter out fragmented packets that

$cmd 210 deny all from any to an frag in via $pif

#Block ACK packets that are not matched dynamically

$cmd 220 deny tcp from any to any established in via $pif

#Outbound web server ports, port 80 and 443 (SSL)

$cmd 230 allow tcp from any to any 80 out via $pif setup keep-state

$cmd 240 allow tcp from any to any 443 out via $pif setup keep-state

#nameserver ports #DNS SERVICE DISABLED

#outbound ping

$cmd 290 allow icmp from any to any out via $pif keep-state

#DHCP requests from all of the 192.168.46.0 subnet.

#$cmd 300 allow udp from any 68 to 192.168.46.0/24 67 out via $pif keep-state #

$cmd 310 allow udp from 192.168.46.0/24 67 to any 68 in via $pif keep-state

#Allow ssh out to any host

$cmd 400 allow tcp from any to any 22 out via $pif setup keep-state

#Only Linux, 192.168.46.140 is allowed ssh to the firewall(me)

$cmd 410 allow tcp from 192.168.46.140 to me 22 setup keep-state

#Outgoing SMTP to send messages if needed #

$cmd 610 allow tcp from any to any 25 out via $pif keep-state setup

# Rules to allow IMP mail to function

$cmd 660 allow tcp from any to any 8444 out via $pif keep-state setup
```

```
#Monitor requests

#$cmd 700 allow tcp from any to any 8180 out via $pif keep-state setup

#Deny rules to filter out bogus packets

#Multicast packets to ignore

#$cmd 700 deny tcp from any to 224.0.0.0/4 in via $pif setup keep-state

#$cmd 710 deny ip from any to 224.0.0.0/4 in via $pif keep-state

#External ICMP redirect requests

$cmd 720 deny log icmp from any to any icmptype 5 in via $pif keep-state

#Prevent spoofing attacks, one should never see any traffic to and from me

$cmd 730 deny log ip from me to me in via $pif keep-state

#Prevent ping echo attacks

$cmd 740 deny log icmp from any to me icmptype 0,8 in via $pif keep-state

#Deny any TCP setup requests from the outside world

$cmd 750 deny log tcp from any to any setup in via $pif keep-state

#Default deny rule

$cmd 10000 deny log logamount 500 all from any to any

############ End of rule set #########################
```

Appendix C

Packet Counter results

The sample output shown here was taken to test ipfw accounting module and see that the traffic sent through the bridge interface was being checked by the firewall. Most rules were set to just count to as to give a picture of packet movement. This was done before the default and optimized rule sets were created.

```
# ipfw show

00005 640322 1808654976 count ip from any to any via lo0

00010 1756 141490 count ip from any to 127.0.0.0/8

00015 1756 141490 count ip from 127.0.0.0/8 to any

00020 0 0 count tcp from any to any frag

00025 0 0 check-state

00035 640330 1808655308 count ip from any to any out keep-state

00040 14 552 count icmp from any to any

00045 0 0 count tcp from any to any dst-port 21 in setup keep-state

00046 0 0 count udp from any to any dst-port 21 in setup keep-state

00050 991008 943447488 count tcp from any to any dst-port 22 in setup keep-state

00060 0 0 count udp from any to any dst-port 53 in setup keep-state

00061 0 0 count tcp from any to any dst-port 53 in setup keep-state
```

```
00065 0 0 count tcp from any to any dst-port 80 in setup keep-state

00070 0 0 count tcp from any to any dst-port 443 in setup keep-state

00075 0 0 count tcp from any to any dst-port 110 in setup keep-state

00080 0 0 count tcp from any to any dst-port 143 in setup keep-state

00090 0 0 count tcp from any to any dst-port 2222 in setup keep-state

00100 1632979 2752301886 count ip from any to any

00100 0 0 count tcp from any to any dst-port 49152-65535 out setup keep-state

00500 1632979 2752301886 allow ip from any to any

00999 0 0 deny log ip from any to any 65535 0 0 deny ip from any to any
```

The output above shows sample output of counters per rule by number, packets matched, the packets size in bytes and the rule itself.

Appendix D

Rule Syntax and use

ipfw firewall rules are written according to a standard syntax that follows network requiremets for packet matching.

The general syntax of an ipfw rule is:

Syntax = CMD RULE_NUMBER ACTION LOGGING SELECTION STATEFUL

CMD: each rule must be prefixed with ***ipfw add*** for it to be added to the internal table.

RULE_NUMBER: each rule should be numbered, though it si not mandatory because rules are numbered automatically. Ipfw rule numbering scheme makes administering a rule set complex as rules are incremented by 100 from a previous number and makes administering a rule set complex.

ACTION: A rule is associated with one of the following actions:

- allow | accept | pass | permit : specified to allow the matching packets into the network. These words mean the same thing.
- check-state : check if the incoming packet matches an established state in the dynamic rules table. Then apply the action defined on the rule that generated the dynamic.
- deny | drop : discards packets matching the rule with this action. The search terninates once this rule is matched.

Logging

- **log** or **logamount :** writes a record to the log file through ***syslogd*** for a packet matching the rule with the log keyword.

Selection: This gives the characteristics of a packet on which action determination is based. There are general functional features applicable as discussed here next:

- **udp | tcp | icmp**: This is a mandatory requirement and can be in any other protocol as listed in /etc/protocols.
- **from src to dst**: ***to*** and ***from*** are used to match originating and destination IP addresses. Other words like ***any*** and ***me*** specify special matching with ***any*** matching any IP address and ***me*** being the host IP of the firewall.
- **port number**: are used to specify services that use port numbers that will be matched.Corresponding service names available from /etc/services can be used instead instead on numeric port values.
- **in | out:** matches incoming or outgoing packets depending on which one is coded for a rule. It is mandatory to specify one of them as part of the matching criterion.
- **via IF**: Matches packets going through the interface specified by name. The interface is always checked as part of the match process.
- **setup**: identifies the session start request for TCP packets.
- **keep-state**: When specified, the firewall creates a dynamic rule that matches bidirectional traffic between source and destination IP/port using the same protocol.
- **limit {src-addr | src-port | dst-addr | dst-port}**: provides same functionality as keep-state differently.It limits N connections that have the same set of parameters.

Adapted from the FresBSD handbook, [15] ipfw section.

Appendix E

CD Contents

The compact disk accompanying this thesis contains all the items used to conduct the research in folders named under. Those not available on it were intentionally left out due to space limitation but the sources have been provided.

1. Project proposal
2. Literature review
3. Poster presentation
4. Final presentation
5. Thesis
6. References
7. Data
8. Softwares
9. Operating Systems