DYNAMIC TRAFFIC DRIVEN ARCHITECTURES AND ALGORITHMS FOR SECURING NETWORKS

by

Subrata Acharya

M.Sc. Computer Science, Major: Computer Engineering, Texas A&M University, USA, 2004

B.Eng. Computer Science and Engineering, University College of Engineering, India, 2001

Submitted to the Graduate Faculty of
the Department of Computer Science in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2008

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF COMPUTER SCIENCE

This dissertation was presented

by

Subrata Acharya

It was defended on

August 08, 2008

and approved by

Dr. Taieb Znati, Department of Computer Science

Dr. Rami Melhem, Department of Computer Science

Dr. Alexandros Labrinidis, Department of Computer Science

Dr. Ehab S. Al-Shaer, Department of Computer Science, Depaul University

Dissertation Director: Dr. Taieb Znati, Department of Computer Science

DYNAMIC TRAFFIC DRIVEN ARCHITECTURES AND ALGORITHMS FOR SECURING NETWORKS

Subrata Acharya, PhD

University of Pittsburgh, 2008

The continuous growth in the Internet's size, the amount of data traffic, and the complexity of processing this traffic gives rise to new challenges in building high performance network devices. Such an exponential growth coupled with the increasing sophistication of attacks, is placing stringent demands on the performance of networked systems (*i.e.* Firewalls). These challenges require new designs, architecture and algorithms for the optimization of such systems.

The current or classical security of present day Internet is "static" and "oblivious" to traffic dynamics in the network. Hence, there are tremendous efforts towards the design and development of several techniques and strategies to deal with the above shortcomings. Unfortunately, the current solutions have been successful in addressing some aspects of security. However, as a whole security remains a major issue. This is primarily due to the lack of adaptation and dynamics in the design of such intrusion detection and mitigation systems.

This thesis focuses on the design of architectures and algorithms for the optimization of such network systems, to aid not only adaptive and real-time "packet filtering" but also fast "content based routing (differentiated services)" in today's data-driven networks. The approach proposed involves a unique combination of algorithmic and architectural techniques that aims to outperform all current solutions in terms of adaptation, speed of operation (under attack or heavily loaded conditions), and overall operational cost-effectiveness of

such systems. The tools proposed in this thesis also aim to offer the flexibility to include new approaches, and provide the ability to migrate or deploy additional entities for attack detection and defense.

Keywords Computer Networks, Network Security, Distributed Denial-of-Service Attack, Firewall, Hierarchical, Optimization.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

PREFACE

First and foremost I would first like to thank Dr. Taieb Znati, my adviser and guide, who has stood by me through this effort, without whose support and constant guidance, it would not have been possible to see the finish line. I would also like to thank Dr. Rami Melhem, Dr. Alexandros Labrinidis and Dr. Ehab S. Al-Shear for agreeing to be on my thesis committee and for their helpful support in the shaping of this document.

My research group members and department colleagues and friends must also be thanked for their help. They were ever ready to provide a helping hand, discuss ideas and topics that would prove helpful to me. It was a pleasure working with them. Special thanks to my dear friends Anandha, Brian, Bryan, Christine, Hammad, Hui, Ihsan, Lory, Mehmud, Michel, Michal, Octavio, Peter, Roxana, Swapna, Weijia and Yaw. I wish that I have such great friends forever in my life.

I would like to thank Bob, Terry, Russ, Chris, Loretta, Kathy, Nancy, Kathleen, Keena and Karen for helping me resolve multiple issues with the department and for patiently answering my many questions about procedures in the department, helping me with with all technical details.

I would like to thank my parents, Mr. Basanta Kumar Acharya and Mrs. Gayatri Acharya for their immense patience and never ending love and support. I would also like to thanks my sister, Mrs. Susmita Acharya, my brother in law, Dr. Ranjan Kumar Dash for their unfaltering love and support. Special thanks to my little sister, Ipsita Acharya and cute nephew Animesh Dash, who keep up and spirits and made me laugh in these trying

times.

Amongst all, I would like to thank my mother, Mrs. Gayatri Acharya for giving me enough and more of her love, care and support, whether it was to discuss my work or to generally talk and regain my confidence. Many a time, just having her in my thoughts helped me to live life positively. My existence would not have been possible without her. I dedicate my Ph.D. and my life to her.

1.0 INTRODUCTION

The continuous growth in the Internet's size, the amount of data traffic, and the complexity of processing this traffic gives rise to new challenges in building high performance network devices. Such an exponential growth coupled with the increasing sophistication of attacks, is placing stringent demands on the performance of networked systems (*i.e.* Firewalls). These challenges require new designs, architecture and algorithms for the optimization of such systems. This thesis focuses on the design of architectures and algorithms for the optimization of such network systems, to aid not only adaptive and real-time "packet filtering" but also fast "content based routing (differentiated services)" in today's data-driven networks. The approach proposed involves a unique combination of algorithmic and architectural techniques that aims to outperform all current solutions in terms of adaptation, speed of operation (under attack or heavily loaded conditions), and overall operational cost-effectiveness of such systems. The tools proposed in this thesis also aim to offer the flexibility to include new approaches, and provide the ability to migrate or deploy additional entities for attack detection and defense.

1.1 BACKGROUND AND MOTIVATION

Data communication networks are today an indispensable infrastructure for industrial and academic institutions. Internet has undoubtedly become the largest public data network, enabling and facilitating both personal and business communications worldwide. The volume of traffic traversing the Internet, as well as corporate networks, is expanding exponentially everyday. As social dependence on such information systems continues to grow rapidly,

a similar growth in threats is concurrently taking place. Traffic anomalies and attacks are commonplace in today's networks. Attacks span the spectrum from computer worms and individual, localized intrusions aimed at gaining access to information and system's resources, to co-ordinated and distributed attacks aimed at disrupting services and disabling critical infrastructure.

Furthermore, the number and frequency of these attacks has been increasing noticeably [1], as the knowledge and tools required to carry out devastating attacks are readily available on the Internet. Figure 1 illustrates the exponential increase in the number of security incidents over the past twenty years. As these attacks proliferate and grow in scope and sophistication, different institutions find themselves under growing pressure to place significant restrictions on open Internet access in the form of firewalls, selective application deployment, and mandatory proxies. Firewalls, constitute the cornerstone of most network defense systems and have been proven to be an effective solution to monitor and regulate traffic.

While network defense systems have been designed in recent years to address the problem, they are not geared towards extremely challenging environments requiring the support for high performance applications and open access policy for collaboration. Most of today's Internet service providers still rely on "*offline*" traffic analysis and manual detection to deal with security threats and *Denial of Service (DoS)* attacks. As such, most of the Intrusion Detection Systems (IDSs) tend to remain "*reactive*" and "*non real-time*" in nature and are "*non adaptive*" to the dynamically changing network environment. The existing IDSs also lack flexibility to deal with the ever-evolving characteristics of the attacks, in terms of diversity and intensity. More recently Artificial Intelligence(AI) based approaches have been explored to solve this problem [49, 50]. While the use of machine learning based approaches holds promise, the schemes still remain "*offline*" in nature due to potentially prohibitive high overhead.

A practical defense against intrusions and *DoS* attacks for high-performance collaborative

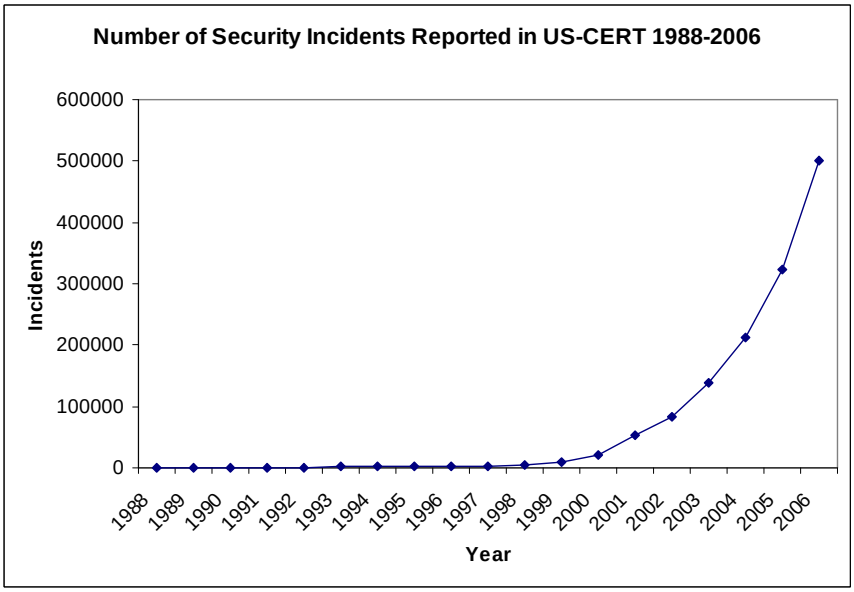

Figure 1: Threat trend (1988 - 2006)

environments must be "*proactive*", "*real-time*", and "*adaptive*" to the changing network environment. The necessary design goals that an IDS must meet are as follows:

- Accurate attack detection in real-time with minimal false alarms,
- Dynamic, collaborative, and resource efficient defense to counter and mitigate attacks, and
- Reliable delivery of legitimate traffic even under attack conditions.

These goals can be achieved using a "*dynamic*", "*proactive*", and "*data-driven*" approach. Thus, the primary focus of this research is to provide *real-time* optimization in packet filtering to achieve the above goals. Furthermore, the thesis also aims to offer flexibility to include new approaches, and would aid the ability of networked systems to migrate or deploy additional entities for detection and defense.

A detailed introduction about firewall technology and an extensive background of their

design and development of firewalls is discussed in Section 1.2. Details on the firewall security policies are discussed in Section 1.3. Section 1.4 discusses the issues and challenges in managing present day firewalls. This section concludes by presenting the challenges towards efficient managing and optimizing enterprise firewalls. The thesis problem and challenges are discussed in Section 1.5. We conclude in Section 1.6 with the contributions of the proposed research.

1.2 BACKGROUND OF FIREWALLS

1.2.1 Taxonomy of Firewalls

A firewall is a combination of hardware and software used to implement a security policy governing the flow of network traffic between two or more networks. The most typical idea of a firewall is a dedicated system or appliance that sits in the network and segments an "*internal*" network from the "*external*" Internet. In its simplest form, a firewall acts as a security barrier to control traffic and manage connections between internal and external network hosts. The actual means by which this is accomplished varies widely, and ranges from packet filtering and proxy service to stateful inspection methods. A more sophisticated firewall may hide the topology of the network it is employed to protect, as well as other information, including names and addresses of hosts within the network. The ability of a firewall to centrally administer network security can also be extended to log incoming and outgoing traffic to allow accountability of user actions and to trigger alerts when unauthorized activities occur. Standard security practices dictate a "*default-deny*" firewall rule-set, in which the only network connections which are allowed are the ones that have not been explicitly allowed earlier. Unfortunately, such a configuration requires detailed understanding of the network applications and endpoints required for the organization's day-to-day operation.

Firewall technology emerged in the late 1980s during the time Internet was a fairly new technology in terms of its global use and connectivity. The original idea was formed in re-

sponse to a number of major Internet security breaches, which occurred in the late 1980s. There are several classifications of firewalls depending on where the communication is taking place, where the communication is intercepted and the state that is being traced. The overall classification of firewalls is depicted in Figure 2.

In general, firewalls can be categorized as either *Desktop or personal firewalls* and *Network firewalls*. The primary difference between the two types of firewalls simply depends on the number of hosts that the firewall protects. *Network firewalls* are classified primarily into three types, namely, *Packet-filter firewalls*, *Circuit-level firewalls* and *Application-level gateways*. Most current networks operate on hybrid versions of the above types of firewalls [39].

1.2.1.1 Personal firewalls Personal firewalls are designed to protect a single host from unauthorized access. Modern personal firewalls now integrate capabilities of antivirus software monitoring, behavior analysis, and intrusion detection to protect the device. Some of the commercial personal firewalls include BlackICE and Cisco Security Agent. Examples in the small-office/home-office market include Trend Micro's PC-cillin, ZoneAlarm, and the Symantec personal firewall. Personal firewalls are geared towards small-office/home-office users as they provide end user protection and control of policies and do not cater to enterprise network security requirements. In the case of enterprise users the need to centralize policy control is very critical to minimize administrative burden.

1.2.1.2 Network Firewalls This classification of firewalls are designed to protect whole networks from attacks. Network firewalls come in flavors, either they are a dedicated appliance or a firewall software suite installed on top of a host operating system. Cisco PIX, the Cisco ASA, Juniper's NetScreen firewalls, Nokia firewalls, and Symantec's Enterprise Firewall are some examples of network firewalls. The most prevalent network firewalls include Check Point's Firewall-1 NG or NGX Firewalls [4], Microsoft ISA Server, Linux-based IPTables [2], and BSD's pf packet filter. This firewall type helps to provide enterprise users the maximum flexibility and protection in a system. Over the years, network firewalls have included many new features such as in-line intrusion detection, prevention as well as remote-

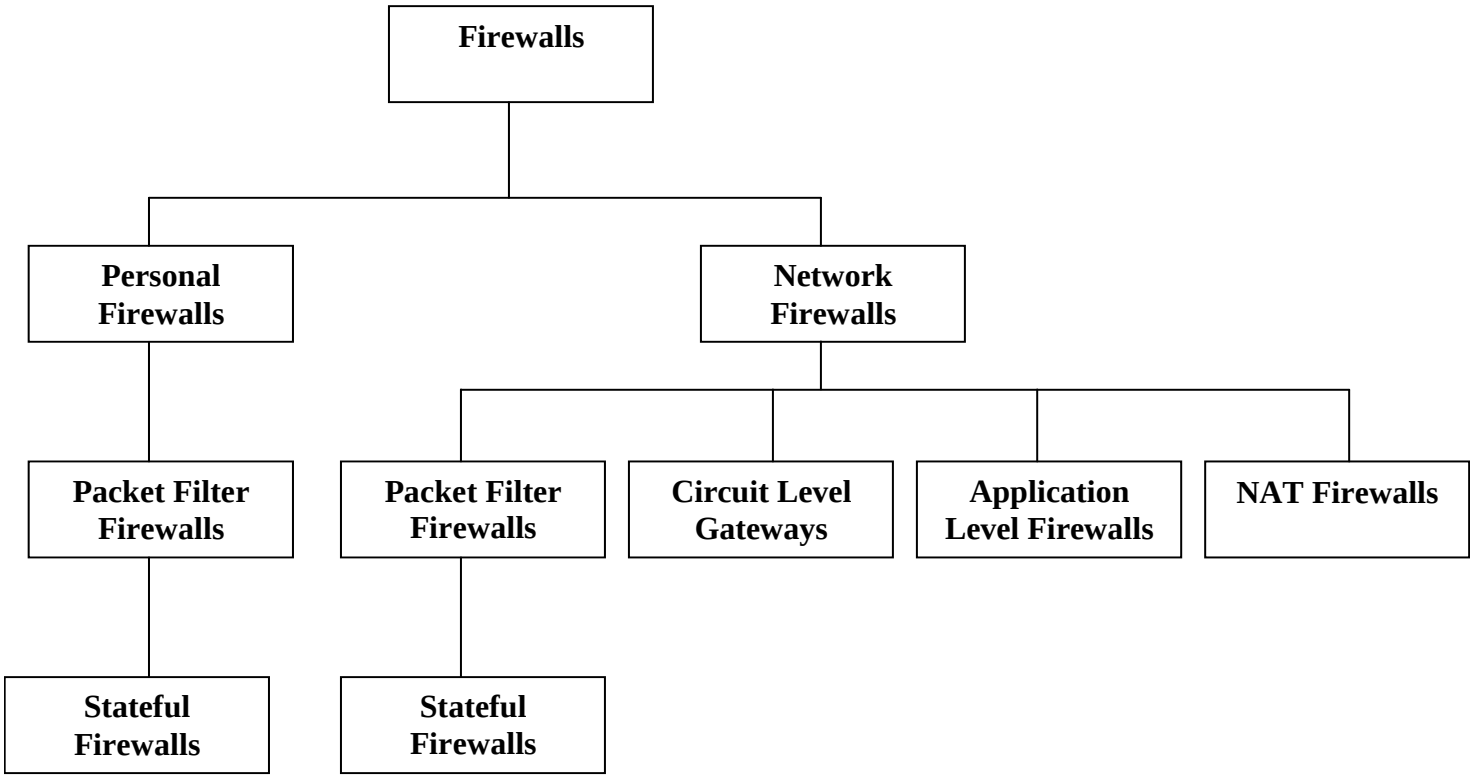

Figure 2: Firewall Taxonomy

access-user VPNs, and deep packet inspection capabilities. The firewall is able to identify traffic requirements not just by scanning Layer 3 and Layer 4 information, but by delving into the application data in order to make informed decisions to handle traffic flow. This evolution in the firewall design and capabilities has led to the development of a new firewall, the Integrated Firewall. We discuss details of this in the following sections.

Application-level firewalls work on the application layer of the *TCP/IP* stack (i.e., all browser traffic, or all *telnet* or ftp traffic), and may intercept all packets traveling to and from an application. *XML* firewall is an example of a recent application-layer firewall. These firewalls block other packets, usually dropping them without acknowledgment to the sender. In principle, application firewalls can prevent all unwanted outside traffic from reaching protected machines. By inspecting all packets for improper content, application firewalls can restrict or prevent the spread of networked computer worms and trojans. In practice, such inspection is difficult to achieve due to the variety of applications and the diversity of content in a given packet filter.

1.2.2 Firewall Products

There exist a wide variety of firewall products that includes three basic physical firewalls, namely, software based, application based, and integrated firewalls. *Software-based firewalls* typically run on top of a commercial operation system. *Appliance-based firewalls* are designed such that the filter and inspection software is tightly integrated into a custom-built or hardware operating system. Cisco PIX products, Juniper's NetScreen firewalls and the Symantec Enterprise firewalls are examples of *Appliance-based firewalls. Integrated firewalls* are a synthesis of other products with the traditional firewall. The synthesis has the benefit of reducing the number of hardware devices that require administration. This helps to lower administrative overhead necessary to deploy and manage these devices. Examples of *Integrated firewalls* include Cisco ASA and the TippingPoint X505 devices.

1.2.2.1 Software Firewalls

Software firewalls include the Sun SunScreen firewall, IPF, the Microsoft ISA Server, Check Point NG, Gauntlet, Linux's IPTables and FreeBSD, and OpenBSD's pf packet filter. The primary advantage of such firewalls is that the administrator can task them to be multipurpose in nature. For example, a firewall can also be a Domain Name System(DNS) server or it can act as a spam filter. The most important benefit of software firewalls is the ability to use commodity software for the device such that on failure, the replacement hardware is relatively straightforward. In smaller environments, software firewalls can be useful low-cost devices for advanced home users. However, for a typical home user, the low-end appliance-based firewalls (such as Linksys, D-Link, and NETGEAR) provides greater benefit due to the ease of setup and low maintenance overhead.

1.2.2.2 Appliance Firewalls

Appliance firewalls are those that are integrated tightly with custom-built hardware and provide firewall service to a network. They include the Cisco PIX, NetScreen firewalls, SonicWall appliances, WatchGuard Fireboxes, Nokia firewalls, *etc.* The underlying operating systems need not be a custom operating system. *Appliance firewalls* can also be highly customized version of the operating system such as the WatchGuard's use of Linux or Nokia's use of FreeBSD. Mostly, appliance firewalls offer better performance relative to software firewalls because of the nature of the customized underlying operating system and the use of specialized processors and application-specific integrated circuits (*ASICs*) for data processing and handling input and output (I/O) requests. Also, these firewalls have the benefit of fewer moving parts by eliminating the hard disk of the software firewalls.

1.2.2.3 Integrated Firewalls

Integrated firewalls are multipurpose devices that combine the traditional firewall with other features such as remote-access VPN, LAN-to-LAN VPN, intrusion detection or prevention, spam filtering, and antivirus filtering. These firewalls are designed for an "*all-in-one*" approach to network-edge security by collapsing the responsibilities of several devices into one device. The most important benefit of integrated firewalls is that they simplify the network design by reducing the number of devices on the network as well as provide a single system for administration, thereby reducing the admin-

istrative burden on the network staff. The major drawback of integrated firewalls is that the failure of such a device can lead to multiple exposures. Additionally, the complexity of the device makes it difficult to troubleshoot connectivity problems due to the interaction of different capabilities in the device and how they affect the underlying fundamental operation of a firewall.

1.2.3 Firewall Technologies

In this section we focus on the technologies used in various firewalls. The focus of this section is on wide range of firewall technologies, namely, *Personal firewalls*, *Packet filters*, *Network Address Translation (NAT) firewalls*, *Circuit-level firewalls*, *Proxy firewalls*, *Stateful firewalls*, *Transparent firewalls*, and *Virtual firewalls*. The firewall technologies presented in this section may be used by any of the three basic physical firewalls discussed earlier. Each of the subsection below focuses predominantly on the operation of the firewall in its function as a network device.

1.2.3.1 Personal firewalls *Personal firewalls* are designed to protect a single host machine. Typically, personal firewalls assume that outbound traffic from the system is to be permitted and inbound traffic requires inspection. By default, personal firewalls include various profiles that accommodate the typical network traffic. For example, ZoneAlarm has low, medium, and high settings that allow almost all traffic, selected traffic, or nearly no traffic, respectively, through to the protected system. Similarly, IPTables during the setup of the Linux system, enables installer to choose the level of protection for the system and the customization for ports that do not fall into a specific profile. The centralized management framework of personal firewalls makes it difficult to be adopted to large enterprise networks due to consistency of security policies amongst the various firewall units.

1.2.3.2 Packet filters *Packet filters* are network devices that filter traffic based on simple packet characteristics. The term packet filter originated in the context of *BSD* operating systems. These devices are typically stateless in that they do not keep a table of the con-

nection state of the various traffic flows through them. To allow traffic in both directions they must be configured to permit return traffic. Examples of simple packet filters are the Cisco IOS access lists, the Linux's ipfwadm facility, etc. Although these firewalls provide protection against a wide variety of threats, they are not dynamic enough to be considered as true firewalls. *Packet filters* operate at a relatively low level of the *TCP/IP* protocol stack, not allowing packets to pass through the firewall unless they match the established rule-set. The firewall administrator may define the rules; or default rules may apply.

1.2.3.3 NAT Firewalls A distinct firewall that exists for a short period is the Network Address Translation (NAT) firewall. NAT is a part of almost every firewall product, from the lowliest small-office/home-office firewall such as the Linksys BEFSX41 to the high-end enterprise PIX 535 firewall. NAT firewalls automatically provide protection to systems behind the firewall because they only allow connections that originate from the inside of the firewall. The basic purpose of NAT is to multiplex traffic from an internal network and present it to a wider network (such as the Internet) as though it was originating from a single IP address or a small range of IP addresses. The NAT firewall creates a table in memory that contains information about connections that the firewall has already seen before. This table maps the addresses of internal systems to an external address. The ability to place an entire network behind a single IP address is based on the mapping of port numbers on the NAT firewall.

1.2.3.4 Circuit-Level Firewalls *Circuit-level firewalls* work at the session layer of the OSI model and monitor "*handshaking*" between packets to decide whether the traffic is legitimate. Traffic to a remote computer is modified to make it appear as though it originated from the circuit-level firewall. This change makes a circuit-level firewall particularly useful in hiding information about a protected network but has the drawback that it does not filter individual packets in a given connection.

1.2.3.5 Proxy Firewalls A *Proxy* is a central machine on the network that allows other machines in that network to use a shared Internet connection. Proxy servers are intermediate

servers which accept requests from clients and forward them to other proxy servers, a source server, or service the request from their own cache. A *proxy firewall* provides Internet access to other computers on the network but is mostly deployed to provide safety or security. It controls the information going in and out the network. Firewall proxy servers filter, cache, log, and control requests coming from a client. A firewall proxy is one that is used for restricting connections from a proxy to the outside world or to the source server inside of the LAN. To support various services, the proxy firewall must have a specific service running for each protocol, a *Simple Mail Transport Protocol (SMTP)* proxy for email, a *File Transfer Protocol (FTP)* proxy for file transfers, and a *Hypertext Transfer Protocol (HTTP)* proxy for web services. Due to their inspection capabilities, proxy firewalls can look more deeply into the packets of a connection and apply additional rules to determine whether a packet should be forwarded to an internal host. Proxy firewalls suffer from the disadvantages of complexity and speed of operation due to the deep inspection functionality. Most modern firewalls include basic proxy server architecture in their operation by providing some form of proxy capabilities. For example, PIX OS 6 and earlier had the "*fixup*" command, and "*IPF*" provided an FTP proxy service to handle active FTP connections.

1.2.3.6 Stateful Firewalls *Stateful firewalls* combine features and capabilities of NAT firewalls, circuit-level firewalls, and proxy firewalls into one system. These firewalls filter traffic initially based on packet characteristics, but also maintain context about active sessions, and use that "*state information*" to speed up packet processing. Unlike proxy or circuit-level firewalls, stateful firewalls are typically designed to be more transparent (like the packet-filtering and NAT firewalls). In a packet filtering type of stateful firewall any existing network connection can be described by several properties, including source and destination IP address, *UDP* or *TCP* ports, and the current stage of the connection's lifetime, which includes session initiation, handshaking, data transfer, or connection completion information. If a packet does not match an existing connection, it will be evaluated according to the rule-set for new connections. If a packet matches an existing connection based on comparison with the firewall's state table, it will be allowed to pass without further processing. Stateful firewalls are more complex than their constituent component firewalls;

however, nearly all modern firewalls on the market today are stateful firewalls and represent the baseline for security in today's network systems.

Inverse to *Stateful firewalls* there exist *Stateless firewalls* that have packet-filtering capabilities, but cannot make more complex decisions on what stage communications between hosts have reached. Stateless firewalls therefore offer less security. Modern firewalls can filter traffic based on many packet attributes like source IP address, source port, destination IP address or port, destination service (*i.e.*WWW or FTP), protocols, TTL values, netblock of originator, domain name of the source, etc.. Commonly used packet filters on various versions of Unix are *ipf*, *ipfw* (FreeBSD/Mac OS X), *pf* (Upend, and all other BSDs), *iptables/ipchains* (Linux) [2].

1.2.3.7 Transparent Firewalls *Transparent firewalls* (also referred to as bridge firewalls) are a considered to be a subset of stateful firewalls. Whereas nearly all firewalls operate at the IP layer and above, transparent firewalls operate in Layer 2, the data link layer, and monitor Layer 3+ traffic. The transparent firewalls can apply packet-filtering rules like any other stateful firewall and still appear invisible to the end user, thus appearing as a filtering bridge between two network segments. This firewall provides an excellent way to apply security policies in the middle of the network segment without having to apply a NAT filter. The bridging firewall requires no changes to the underlying network, which is possible as the transparent bridge is plugged in-line with the network it is protecting. Since, transparent firewalls operate at the data link layer, no IP address changes are required and hence are simpler with lower processing overhead. The lower overhead enables them to provide better performance as well as deeper packet inspection. Finally, the stealth nature of these firewalls are due to the fact that they are Layer 2 devices. The firewall is invisible to an attacker and hence cannot be reached by an attacker.

1.2.3.8 Virtual Firewalls *Virtual Firewalls* are multiple logical firewalls running on a single physical device. This arrangement allows for multiple networks to be protected by a unique firewall running a unique security policy all in one physical appliance. A service

provider can provide firewall services for multiple customers, securing and separating their traffic while managing the entire system on one device. Service providers achieve the above by defining separate domains for each customer with each domain controlled by a separate logical virtual firewall. Typically, this capability is available in higher-end firewalls such as the Cisco PIX 525, Cisco PIX 535, and the Cisco ASA devices.

1.2.4 Open and Closed Source Firewalls

The *Open source firewalls* available today are Linux's IPTables, OpenBSD's pf, and the Solaris IPF firewalls. *Closed source firewalls* are the Cisco PIX and ASA firewall, Juniper's ScreenOS, and Check Point's firewall software. Some firewalls use an underlying open source operating system and firewall code with closed modifications. Most commercial firewalls provide tight integration of VPN capabilities for remote users as well as deep packet inspection within the firewall. In contract, Open source firewalls focus on the filtering capabilities in the firewall process rather than the integration of the firewall with other applications.

1.3 FIREWALL SECURITY POLICIES

The term "*security policy*" refers to the written policies that dictate how the organization manages the security of their resources and also the actual configuration of the network security device (firewall) with the help of access control lists (ACLs).

Security policies exist to provide a road-map of what needs to be done to ensure that the organization has a well-defined security strategy. Any organization's overall security policy typically consists of numerous individual security policies, which are written to address specific objective, devices, or issues. The objective of a security policy is to define what needs to be protected, who is responsible for protection, and in some cases how the protection will occur. The last function is typically separated out into a standalone procedure document such as the ingress-filtering, egress-filtering, or the management-access policy documents.

The security policy should simply and concisely outline the specific requirements, rules, and objectives that must be met, to provide a measurable method of validating the security of the organization.

The firewall can be represented as set of security layers as depicted in Figure 3, where each layer has a distinct operation. At the center is the firewall physical integrity layer, which is predominantly responsible for the physical access to the firewall. The security policy in this layer should address the issues related to gaining physical access to the device. The next layer is the firewall static configuration, which is concerned with access to the static configured software the firewall is running. In this layer, the security policy is responsible for defining the controls that will be required to restrict administrative access, including performing software updates and configuring the firewall. The third layer is the firewall dynamic configuration, which complements the static configuration by being responsible for the dynamic configuration of the firewall through the use of technologies, such as routing protocols, Address Resolution Protocol (ARP) commands, interface and device status, audit logs, and shun commands. The objective of the security policy at this point is to define the requirements around what kinds of dynamic configuration will be permitted. Finally, we have the network traffic through the firewall layer, which is responsible for protecting the resources with the help of ACLs and service proxy information. The security policy at this layer is responsible for defining the requirements as they relate to the traffic passing through the firewall.

1.3.1 Security Policy Format

In order to accomplish the security goals discussed earlier, security policies follow a particular format and share common elements. There are basically seven sections, namely, *Overview*, *Purpose*, *Scope*, *Policy*, *Enforcement*, *Definitions*, and *Revision history*. The *overview* section provides a brief explanation of what the policy addresses. The purpose of the security policy is discussed in the *purpose* section. The *scope* section defines what the policy applies to and

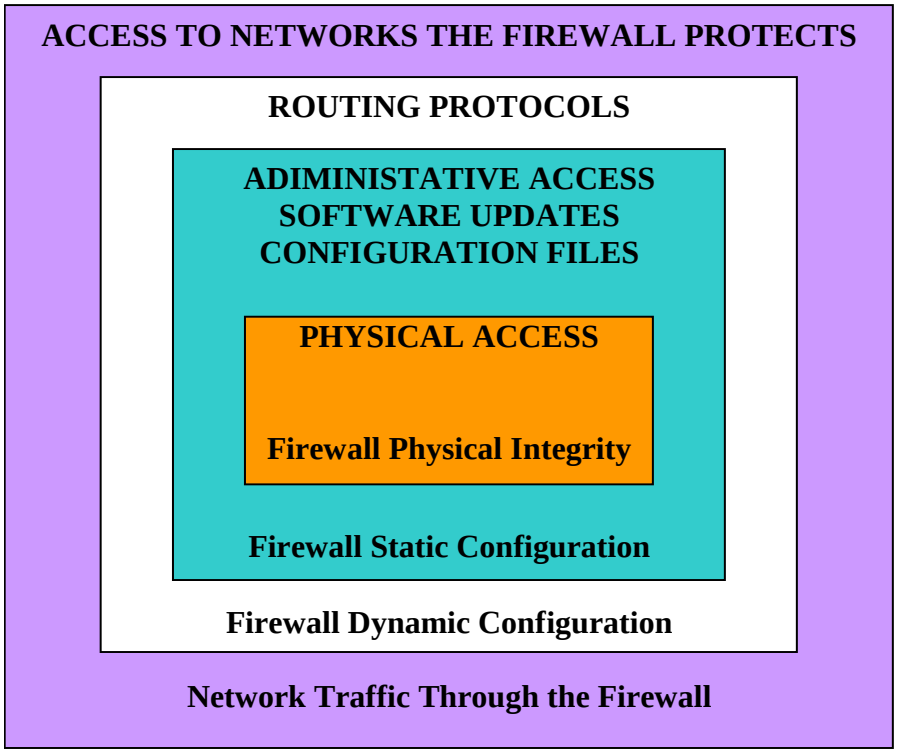

Figure 3: Firewall Security Layers

defines who is responsible for the policy. The *enforcement* section defines how the policy should be enforced and the repercussions of not following the policy. The *definition* section contains the definition of any terms or concepts used in the policy. Finally, the *revision history* section is where the changes to the policy are documented and tracked.

1.3.2 Common Security Policies

Each organization has unique security requirements and hence have unique security policies. However, there are a required number of common security policies, namely, *Management-access policy*, *Filtering policy*, *Routing policy*, *Remote-access/VPN policy*, *Monitoring/logging policy*, and *Demilitarized zone (DMZ) policy*. We will present each of them in detail in the following subsections.

1.3.2.1 Management-access policy The *Management-access policy* is used to define the permissible methods and manner of management access to the firewall. This policy tends to address the firewall physical integrity and static configuration security layers. This policy defines the permissible protocols for both remote and local management, as well as the allowable users that can connect to the firewall, and the permissions those users will have during the time they perform tasks. Additionally, the policy also defines the requirements for management protocols such as Network Time Protocol (NTP), syslog, TFTP, FTP, Simple Network Management Protocol (SNMP), etc.

1.3.2.2 Filtering policy A filtering policy is used instead of the actual firewall rule-set to define the kinds of filtering that must be used and where filtering is applicable. This policy addresses the firewall static configuration, specifically, the network traffic through the firewall layers. A good filtering policy requires both ingress and egress filtering for the given firewall. This policy also defines the general requirements in connecting different security level networks and resources.

1.3.2.3 Routing policy Complex perimeter design and increased use of firewalls within the internal network, causes firewalls to be part of the routing infrastructure. The routing

policy specifies the firewall in the routed infrastructure and defines the method in which the routing is conducted. This policy addresses the firewall static and dynamic configuration layers. Mostly, routing policy explicitly prohibits the firewall from sharing the internal routing table with any external resource. It also determines the scenarios under which static or dynamic routing protocols can be invoked.

1.3.2.4 Remote-access/VPN policy In today's network environment, all major firewalls can serve as the termination points for VPNs. Thus, the remote-access/VPN policy defines the requirements for the level of encryption and authentication that a VPN connection would require. This policy also defines the protocols that are used, namely, IP Security (IPSsec), Layer 2 Transport Protocol (L2TP), or Point-to-Point Transport Protocol (PPTP). Finally, the policy also determines what kind of access and resources will be made available to remote connections and the types of connections that will be allowed.

1.3.2.5 Monitoring/logging policy A firewall monitoring system is critical to ensure that a firewall is providing the expected level of security. The monitoring/logging policy defines the methods and degrees of monitoring performed by the firewall. It provides a mechanism for tracking the performance of the firewall as well as the occurrence of all security-related events and log entries. This policy addresses the firewall static configuration layer. This policy also defines the manner in which logging information is collected, maintained, and reported.

1.3.2.6 Demilitarized zone (DMZ) policy The objective of the Demilitarized zone (DMZ) policy is to define the standards and requirements of all devices, connectivity and traffic flow information. This policy tends to address the firewall static configuration and network traffic through the firewall layers. There are three broad standards defined for all DMZ-related devices, namely, *Ownership responsibilities*, *Secure configuration requirements*, and *Operational and change-control requirements*.

In addition to the above firewall-specific policies, there are many generally applicable

policies,namely, *Password policy*, *Encryption policy*, *Auditing policy*, and *Risk-assessment policy*. The password policy helps to define administrative access to the firewall, creation of preshared secrets, hashes, and community strings. All forms of encryption access, including Hypertext Transfer Protocol, Secure (HTTPS), Secure Sockets Layer (SSL), Secure Shell (SSH), and IPsec/VPN access are referred *via* the encryption policy. The audit requirements of the firewall employs the auditing policy. Enterprise risk-assessment policy helps to define the methodology that is used to identify the risks associated with all system updates(additions, deletions, and changes).

1.3.3 Firewall Policies/Rule-sets

The "*firewall policies*" or "*rule-sets*" is the actual policy build based on the requirements and specifications set by the organization. There are three common rule-sets for any given firewall, namely, *Ingress*, *Egress*, and *Management-access* filters. Ingress filter is used to restrict traffic coming into an interface or from a given network segment. This type of filtering is commonly applied to traffic coming from an untrusted source (Internet) to a trusted source (internal network). Unlike ingress filters, egress filters apply to traffic coming from a trusted network to an untrusted network. Thus, egress filtering is typically applied to firewall interfaces that connect to the internal network or to a DMZ segment.

The management access rule-set specifies the management details required by the enterprise network. It is important to restrict the management access to specific management workstations and never allow access of this information from an untrusted network. Due to the information criticality it is always advisable to use encrypted management method, if this is not a possibility, IPsec can be implemented to secure the traffic instead. There are various methods of performing remote management and logging of firewalls, namely, *Telnet and SSH*, *SNMP*, *Syslog*, *TFTP and FTP*, *HTTP and HTTPS*, etc.

The most important factor in ensuring that the firewall is an effective security measure as well as is secure in and of itself relies on a well-planned methodology of defining the security

requirements and objectives with a series of written security policies. The policies developed can thus be used to build effective and functional filters.

1.4 FIREWALL MANAGEMENT

From the perspective of the small office/home user, the firewall is a single device that protects the home network from malicious traffic. For the enterprise network, the firewall can be both an inbound filter as well as an outbound filter depending on how the security policy calls for enforcing the edge network. In both the above cases, it is important to manage firewalls to ensure secure operation. In this section we discuss the management of firewalls through the CLI and the GUI interfaces. We then discuss firewall management access and present the common firewall management tasks. Finally, we present the challenges towards firewall management and optimization for large scale enterprise networks, which is the focus of this research.

1.4.1 Firewall Management Interface

Modern firewalls can be operated *via* two administrative interfaces, namely, *Command Line Interface (CLI)* and *Graphical User Interface*. A CLI enables the use of a specific instruction set to configure the firewall. The initial configuration of the firewall is performed by the end user in the CLI interface. CLI usage demands the knowledge of the command set in the firewall product. Linux NetFilter, for the most part, is configured *via* command line interface. In contrast to CLI, a GUI provides a more user-friendly interface to configure the firewall. The GUI for Symantec Norton Internet Security is configured through a direct interface on the host system. GUI usage for Linksys, PIX 501 and 506E series systems comes with a preconfigured IP address and an administrative password to enable access to the firewall. An important advantage of CLI over GUI based interfaces is the availability of CLI *via* Telnet and SSH sessions as well as through direct connections to the serial port.

1.4.2 Firewall Management Access

Access control to the management interface of network systems is vital to maintain expected secure operation of the firewall. To provide this, network device access is usually restricted to the administrator of the network system. Firewall management access can be of two types, namely, *in-band* and *out-of-band* management. *In-band* management refers to the administrative access to systems and network devices over the same network as used by the traffic being filtered. In-band management should always be through encrypted communication, *via* SSH or HTTPS. In contrast, *Out-of-band* management results in access to the firewall through a secondary channel that is not carrying production traffic. This can either be a VLAN setup for administrative access to network devices and hosts, or, preferably, a completely separate physical network. Out-of-band management can also be used to provide access to the serial port of the network device for access, should the network fail. In comparison to in-bound management, out-of-bound management is a more time consuming and costly management access.

1.4.3 Firewall Management Tasks

The first step to firewall deployment, both for enterprise or small-office/home-office networks, is to configure it. Configuration involves changing the default administrative password, configuring the default gateway, configuring the IP addresses for the internal and external interfaces, and configuring the logging of messages from the firewall. In addition to the above, the firewall administrator also manages the configuration of the firewall over time. This task requires the use of change control system (Revision Control System (RCS)).

Initial configuration is the first task performed for firewall management and operation. This task requires information about both the internal and external interface IP addresses, the next-hop gateway, logging, and an administrative password. The firewall needs to be maintained and updated with the correct security policies of the organization it protects. *Modifying the configuration* task achieves this objective by enabling changing and modifying the firewall configuration. The modifications are with the help of *change control*, which

ensures that the changes made are tracked and logged in case of any unexpected behavior and/or anomalies. The final task while managing firewalls is to *Update the Firewall Software*. There are two primary reasons to perform this task. Firstly, this task helps to take advantage of new capabilities added to newer software versions. Secondly, this tasks aids the administrator to fix bugs and other vulnerabilities in the software.

Managing firewalls is very vital to ensure security in any given network. Updating firewall software and managing and modifying the firewall policy set will ensure that a vulnerability is detected and mitigated at its inception to maintain the security of such network environments.

1.4.4 Complexity of firewall management and optimization

Firewalls have proven to be useful in dealing with a large number of threats that originate from outside a network. They are becoming ubiquitous and indispensable to the operation of the network. The continuous growth of the Internet, coupled with the increasing sophistication of attacks, however, is placing further demands and complexity on firewalls design and management. Increased firewall complexity, undoubtedly, brings with it increased vulnerability and reduced availability of individual network services and applications. Analysis of real configuration data has shown that corporate firewalls are often enforcing rule-sets that violate established security guidelines.

Furthermore, the need to deal with large sets of diverse security policies and rules imposes additional burden on firewalls, thereby rendering the performance of the firewall highly critical to enforcing the network security policy. In this context, the protection that a firewall provides depends not only on the policies it is configured to implement, but equally on the speed at which it enforces these firewall policies. Under attack or heavy load, firewalls can easily become a bottleneck. As the network size, bandwidth, and processing power of networked hosts continue to increase, there is a high demand for "optimizing" firewall operations for improved performance.

Multi-dimensional firewall optimization is proven to be NP-hard [33, 45]. This has led the research community to focus on developing various "optimization" heuristics to make firewalls more efficient and dependable. Despite significant progress in the design of firewalls, the techniques for firewall optimization remain static, and as such, fail to adapt to the continuously varying dynamics of the network. This is mostly due to their inability to take into account the traffic characteristics logged by the firewall, such as source and destination, service requests and the resulting action taken by the firewall in response to these requests. Moreover, current firewall designs do not support adaptive anomaly detection and countermeasure mechanisms to deal with short and long term attacks. Consequently, they run the risk to become unstable under attack.

In this thesis we aim to address the above shortcomings and develop a sound and effective optimization framework to optimize firewall operations and "*adapt*" its performance to the dynamically changing network traffic characteristics. In the following section we present the thesis problem and challenges in achieving the desired goal.

1.5 THESIS PROBLEM AND CHALLENGES

The firewall technologies of present day Internet are "static" and "oblivious" to traffic dynamics in the network. With the advent of ultra-high speed ($Gbps$) networks, this "static" and "traffic oblivious" approach has become a major hindrance in providing security and performance of such network systems. Inspite of huge attention by the research community, unfortunately, to date there is no current solution which addresses the above shortcomings. Hence, as a whole security remains a major issue due to the:

- Lack of dynamics in the design of architectures and algorithms for intrusion detection and mitigation systems
- Lack of adaptation of such systems

Thus, the objective of this research is to address the above shortcomings and develop a sound and effective framework and algorithms to accelerate firewall operations and "adapt" its performance to the dynamically changing network traffic characteristics. The thesis aims at addressing the problem of firewall management and optimization taking the dynamics of network traffic into consideration. Further, the thesis aims to design a proactive security model which includes various optimizations to aid performance of such network systems.

Achieving this goal is a challenge, as the number of policies and security rules a firewall has to enforce for enterprise networks is large. In addition, there is a need for maintaining high policy integration. This is further compounded by the limited resources of firewalls relative to the increased ability of the network to process and forward traffic at extremely high speed.

Thus, the complexity of packet filtering along with providing content based services for such scalable network systems, under the constantly changing and evolving network dynamics makes the problem of firewall optimization a formidable task. Along with this, maintaining the availability of the filtering agents (core routers or firewall servers), while meeting real time requirements, is a daunting task. This thesis addresses these challenges by designing better ways to search, update and filter a large set of data based on specific policies and rules to optimize and better strategize to defend against attacks which go beyond the classic peripheral model.

To state formally, this thesis aims to answer the fundamental question as follows:

Whether dynamic de-centralized (as well as centralized) approaches to attack detection and mitigation feasible, and if so, would de-centralized, traffic aware "defense" mechanisms enhance the ability of a network system (servers and core routers) to protect against anomalies and malicious behavior.

Thus, the goal is the design of a ***Dynamic*** (gradual deployment through collaboration), ***Adaptive*** (traffic-aware) and ***Scalable*** global defense platform. Our approach involves a unique combination of algorithmic and architectural techniques that have a potential to outperform all current techniques in terms of *adaptiveness*, *speed* of operation (under attack/heavily loaded conditions) and overall operational *cost-effectiveness* of such network systems.

1.6 THESIS CONTRIBUTION

The objective of this thesis is the design of dynamic and traffic-aware defense architectures and algorithms to secure the present day network systems. The thesis contribution is enumerated as follows:

1. Design of techniques aimed at reorganizing firewalls and developing architectures and algorithms to optimize firewall rule-sets

   The first contribution is the design of *PITTWALL*, a centralized firewall optimization toolkit. We present the design of a traffic-aware *Firewall Optimization Framework* and optimization algorithms to improve the operational cost of firewalls. We validate our contribution by evaluating on rule-set and traffic log data available from a large *Tier-1 ISP* against the widely used commercial *Checkpoint NGX* firewall [4].

2. Expand the linear model of security in networked systems towards a de-centralized design to achieve "optimal" operation cost of firewalls and reduce their management overhead

   As the second contribution we propose the design of a firewall optimization framework and optimization approaches aimed at improving the performance and efficiency of de-centralized firewalls. We achieve the firewall transformation *via* "*rule-splitting*" and present an "*optimal*" and a "*heuristic*" approach to achieve the above transformation. We present the design and implementation of *OPTWALL*, a hierarchical traffic-aware optimization tool to validate the contribution. Our proposed approach is augmented

onto the open source firewall, *Linux IPCHAINS* [2].

3. Develop techniques to capture the nature and behavior of traffic and its characteristics in the design of efficient firewalls

   The third contribution of this thesis is the development of traffic-aware optimization techniques to capture traffic information in the design of firewalls. We validate the proposed techniques by evaluating on "*hit-count*" characteristics of the traffic. The proposed architectures and algorithms are fully flexible to include other traffic characteristics as per the decision of the network administrator. We validate the contribution on rule-set and traffic log data available from a large *Tier-1 ISP*.

4. Design and implementation of an online adaptation scheme to adapt to short term and long term traffic anomalies

   In the fourth contribution we present the design of a strong anomaly detection and mitigation approach to defend against short term and long term traffic behavior. Our proposed adaptation technique is based on the characteristics of "*variability* between the predicted and actual traffic data. We have incorporated the proposed contribution in both the centralized ($PITTWALL$) and hierarchical ($OPTWALL$) firewall optimization toolkit.

1.7 THESIS ORGANIZATION

The rest of the thesis is organized as follows: Chapter 2 provides literature review of current architectures and algorithms for firewall optimization. Chapter 3 presents the data set and the analysis that motivates the research work in this thesis. The centralized firewall optimization model and its architectures and algorithms are introduced in Chapter 4. In Chapter 5, we outline the details design of the De-centralized/Hierarchical Firewall optimization framework. The various approaches to improve Rule Splitting for Hierarchical

firewall design are discussed in detail in this Chapter. Finally we conclude with the thesis with the thesis summary in Chapter 6 and present the a very important future direction in Chapter 7.

2.0 BACKGROUND AND RELATED WORK

Firewalls are becoming ubiquitous and indispensable to the operation of the network. The continuous growth of the Internet, coupled with the increasing sophistication of attacks, however, is placing further demands and complexity on firewalls design and management. Increased firewall complexity, undoubtedly, brings with it increased vulnerability and reduced availability of individual network services and applications.

Analysis of real configuration data has shown that corporate firewalls are often enforcing rule sets that violate established security guidelines. Furthermore, the need to deal with large set of diverse security policies and rules imposes additional burden on firewalls, thereby rendering the performance of the firewall highly critical to enforcing the network security policy. In this context, the protection that a firewall provides becomes as good as, not only the policies it is configured to implement, but equally importantly the speed at which it enforces these policies. Under attack or heavy load, firewalls can easily become a bottleneck.

As the network size, bandwidth, and processing power of networked hosts continue to increase, there is a high demand for "*optimizing*" firewall operations for improved performance. The efficiency of firewalls in protecting the infrastructure, however, depends not only on the integrity and coherence of the security policies they are configured to implement, but equally importantly on the speed at which these policies are enforced. Optimizing firewalls, however, remains a challenge for network designers and administrators.

Due to the enormous impact of firewalls on network security, there has been a significant amount of research work on how to optimize firewalls. Recent researchers in both industry

and academia have focused extensively on the problem of packet classification and optimization. Though packet classification is extensively studied, it is an evolving problem. With the growth and changing needs of security threats and services, the security policies are larger and more and more complex. This increased complexity and size impose challenges to hardware based solutions and drive the design of software solutions. Current packet classification solutions aimed at improving the matching time of filters include hardware based, geometric based, specialized data structure based or other heuristics and statistical solutions. We discuss each of the approaches and their drawbacks in the following section.

2.1 PACKET CLASSIFICATION AND OPTIMIZATION

2.1.1 Hardware Based Solutions

Hardware based packet classification solutions using the concept of *Context Addressable Memory (CAM)* exploit the notion of parallelism in hardware to speed of the rate of packet matching. These solutions are limited to small policies due to cost, power and size limitations of the $CAMs$. There are other hardware based solutions described in [35], but all of the solutions are limited to number of rules. The policy structures the rules as a trie, with the classification time as $O(B)$, where B is the total number of bits in all dimensions. The value of B can be very large as the bits and dimensions in the tuples increase.

The *Aggregated Bit Vector (ABV)* [9] approach helps to solve the problem with d independent lookups on one dimension followed by a merging phase. Lookups are performed for each dimension and then the final rule list is computed by finding rules with highest priority. The memory consumed for storing the rules is extremely large and hence a compressed bit vector is used instead. [46] builds a table of all possible field value combinations and precomputes the earliest rule matching each cross-product. Search is conducted by separate lookups on each field. The results are then combined into a cross-product table followed by

indexing into the table. The limitation of this approach is that the cross-product table grows significantly with the number of rules. There have also been other similar hardware based solutions over the years but all of them lack the ability to handle (with real time guarantees) large policy sets ($\sim 1,000,000$) due to memory size, power and cost limitations.

2.1.2 Geometric based solutions

Another research direction to address this problem was to search for geometric based solutions. Feldmann *et. al.* in [19] proposed a geometric based solution and introduced a data structure called Fat Inverted Segment (FIS) Tree. FIS partitions the first dimension with the endpoints of the projection of the rules on that dimension. Each of the segments is then partitioned, according to the remaining dimensions of the rules covering each segment, into a number of d dimensional regions. To curb the storage requirement of such a structure, the d dimensional regions are linked in a FIS Tree of bounded depth, and the common partitions of the regions are pushed up in the tree. The proposed solution scales better than others, but still cannot meet the tuple sizes for large Tier-1 ISPs. Another geometrical based solution, the Decision-tree based algorithm was introduced by [48, 22, 25], builds a decision tree using local optimizations at each intermediate node to choose the next bit to test. Additionally, [48] uses multiple decision trees which helps to reduce storage with increase in search time. Similarly, [22] uses range checks instead of bit checks at each node of the decision tree to advance the search for a packet match.

2.1.3 Specialized data structures

Researchers have also proposed various specialized data structures to enable fast packet classification. [45] builds a table of all possible field value combinations and pre-compute the earliest rule matching each of them. Search is accelerated by performing separate lookups on each field and then combining the results into a combination table followed by indexing into the table. However, the approach does not scale to large number of rules. Furthermore,

numerous heuristic approaches have also been researched to aid fast packet classification. The Recursive Flow Classification (RFC) [23] approach pipelines various packet matching stages to achieve high throughput for hardware based implementations. Although all the above research contributes to fast packet filtering, they only focus on improving the worst-case (not the average case) matching time for packet filters. In addition, they exhibit high space complexity, which limits their practical deployment in large Tier-1 ISPs.

2.1.4 Statistical based solutions

In [24] the authors introduced a statistical data structures in optimizing packet filtering. They used depth-constrained alphabetic trees to reduce lookup time of destination IP addresses of packets against entries in the routing table. The authors show that using such statistical data structures it is possible to improve the average-case lookup time for packet filters. The work focuses on only single dimensional filters and does not consider any traffic dynamics in rule-set building or real time operation of packet filters. The work in [41] presents algorithms to optimize filtering policies by aggregating adjacent rules and eliminating redundant ones to reduce the size of the rule list. However, the work did not consider traffic information in its optimization approach.

2.2 FIREWALL OPTIMIZATION

2.2.1 Policy based optimization

Most of the current research in firewall optimization has been in the area of firewall policy modeling and optimization [20,21,42,22,25,44,17,28,47]. Very few attempts have been made to achieve multi-dimensional firewall optimization. In [8], a tool to model firewall policies and detect conflicts is described. In this work, the authors focus mainly on single attribute rules. Similarly, in [17] a constraint logic programming (CLP) framework to analyze rule-sets

is discussed. These research work offer a good insight in how to model and analyze rule-sets. Neither of these approaches, however, consider optimizing a multi-dimensional rule-set.

The approach proposed in [20] optimizes the firewall rule-set using Directed Acyclic Graphs (DAGs) to describe rule dependencies. However, it does not provide a methodology to build the DAG. Furthermore, for complex graphs this scheme is ineffective. The approach proposed in this thesis removes all the dependencies and hence it becomes possible to achieve optimum rule ordering. In [41], a framework to analyze and optimize rule-sets is described. However, the authors do not provide specific details on how optimization can be achieved within the proposed framework. Furthermore, this work does not consider the traffic characteristics in its optimization approach. In this thesis we design architectures and algorithms to optimize firewalls that differs from literature, in its unique approach to consider firewall traffic characteristics in optimizing the rule-sets.

2.2.2 Traffic based optimization

Recently there has been great attention to address traffic-aware firewall optimization. Some efforts in rule reordering using traffic specifications as in [16,26,27] have been proposed. But these approaches consider a very small firewall policy set (~ 200) in comparison to realistic Tier-1 ISP firewall data sets ($\sim 1,000,000$ policies). The current approaches also lack complete rule reordering due to existing dependencies in the policy sets. Furthermore, in these approaches all traffic characterizations are not considered for firewall optimization. [7] presents a tool geared towards adaptive optimization of list based firewalls, however, the work falls short of addressing non-linear policy optimization. In other words, it does not consider any traffic-aware design improvements in the firewall structure. All these previous efforts suggest the lack of and need for dynamic, global firewall optimization architectures and algorithms towards achieving next generation Internet security.

2.3 ANOMALY DETECTION AND MITIGATION

The design and deployment of the decentralized packet filtering framework requires the understanding of the various types of distributed attacks in the current networked systems. In the following we present a brief background of attacks [3,37,14]. The primary reason behind such attacks over the Internet is its design solely for functionality and not security. The Internet's distributed nature and the absence of a common policy makes it susceptible to various kinds of attacks. Today's Internet security is a highly interdependent concept; the resources are limited and the intelligence and resources are not collocated. A typical distributed attack starts when the attacker *recruits* multiple slave machines by infecting them with attack code. These slave machines typically launch attacks on behalf of the attacker with a spoofed source address. The reason behind this attack is to inflict damage on the victim either for personal reasons, material gain or for popularity.

2.3.1 Attack classification

Distributed attacks are classified either based on the means used to prepare and perform the attack, the characteristics of the attack itself, and the effect it has on the victim. In the classification based on *Degree of Automation* the attacker needs to locate prospective agent machines and infect them with the attack code. These attacks are classified as *manual*, *semi-automatic* or *automatic attacks*. Current distributed attacks are either *semi-automatic* or *automatic*. In the *semi-automatic* attacks the nodes are infected using automated scripts but then the attacker manually issues commands for the actual attack (with direct and indirect communication *e.g.* usage of IRC channels for agent/handler communication) and automatic attacks. Attack phase is also automated, thus avoiding the need for communication between the attacker and the agent machines. *Automatic* distributed attacks automate the attack phase. They avoid communication between the attacker and the agent machines. All operations of the attack are preprogrammed in the attack code.

The agent machines in the *semi-automatic* and *automatic* attacks deploy various scanning and propagation techniques. The scanning can be "*random scanning*" (*e.g.* Code Red), "*hitlist scanning*, "*topological scanning*", "*permutation scanning*", and "*local subnet scanning*". In *random scanning* the compromised host probes random addresses in the IP address space, using a different speed. The scanning in *hitlist scanning* is performed based on an externally provide list. This technique is very powerful as there are no collisions during the scanning phase. *Topological scanning* employs information on the compromised host to select new targets. Email worms follow topological scanning technique. In *Permutation scanning* all the compromised machines share a common pseudo- random permutation of the IP address space, where each IP address is mapped to an index in this permutation. *Local subnet scanning* preferentially scans for targets that reside on the same subnet as of the compromised host. *Code Red II* and *Nimda Worm* are examples of *Local subnet scanning* attacks.

Another classification of distributed attacks is based on the target of vulnerability during the attack. This attack classification of *Exploited Vulnerability* include those that are "*Protocol (TCP SYN attack, CGI request attack, authentication server attack, etc)*" and the "*Brute-force attacks*". *Protocol attacks* exploit a single or multiple feature of any protocol. *TCP SYN* attack, *CGI request attack* and *Authentication server attack* are examples of *Protocol attacks*. *Brute-force attacks* are achieved by initiating a vast amount of seemingly legitimate transactions. This exhausts the victims network resources. *Brute-force attacks* are classified as either *Filterable* or *Non-filterable* attacks. Attacks that employ packets for non-critical services of the victim's operation and can be filtered by a firewall belong to the category of *Filterable attacks*. UDP flood attack and ICMP request flood attack on a Web server are *Filterable attacks*. Attacks requesting legitimate services from the victim constitute *Non-filterable attacks*. Example of a *Non-filterable attacks* is a HTTP request flood targeting a Web server or a DNS request flood targeting a name server.

One other classification of distributed attack is based on the *Attack Rate Dynamics*. The attacks in this classification include those that are either "*Continuous rate attacks*" or "*Variable rate attacks*". Most attacks employ continuous rate mechanisms. The rate of

change of attacks can be either *increasing rate* or *fluctuating rate*. The final category of classification of distributed attacks is based on the impact of the attack. *Impact attacks* can be classified either as *Disruptive attacks* or *Degrading attacks*. In the *Disruptive attack* case the attacker completely denies the victim's service to its clients. Most distributed attacks are disruptive in nature. *Degradation attacks* as the name implies attack only a portion of the victim's resources. Due to its extent of disruption there attacks remain undetected for for a significant period of time.

2.3.2 Defense mechanisms

To counter these attacks there are various defense mechanism proposed in the current literature. The distributed defense mechanisms are classified either based on the *activity level* or on the *location of deployment*. Activity level defense mechanisms are either *Preventive* or *Reactive* in nature. *Preventive* mechanisms work by either mitigating the attack while maintaining desired availability to the legitimate clients or by completely eliminating the attack. These include resource accounting and resource multiplication mechanisms. *Preventive attacks* are classified as *attack prevention* and *denial of service prevention* mechanisms.

In *attack prevention* the system configuration is modified to eliminate the possibility of attack. In *denial of service prevention* the victim endures the attack while servicing the legitimate clients. Contrary to *Preventive* mechanisms, *Reactive* mechanisms include mechanisms where the goal is to detect as quickly as possible and have a low degree of false positives. *Reactive* mechanisms are classified based on their attack detection strategy to those of "*Pattern detection*", "*Anomaly detection*" and "*Hybrid detection*". In *Pattern detection* the known attack signatures are stored in a database. Periodic Comparison against a *normal behavior* model is the basis of *Anomaly detection*. Challenges in this detection strategy include threshold setting and model update. *Hybrid detection* uses a combination of the above two defense mechanisms.

Defense mechanisms of *location of deployment* are categorized based on the location as either *Victim-Network*, *Intermediate-Network* or *Source-Network* mechanisms. *Victim-Network* mechanisms protect the victim against distributed attacks. *Intermediate-Network* mechanisms provide support mechanisms and service numerous Internet hosts. In the *Source-Network* mechanism, the source network is monitored to prevent service abuse by clients using the network. All the above attack and defense classification helps to understand the gravity of distributed attacks and provide a platform to aid the design of mechanism for dynamic defense.

In this chapter we present the background and related research in the area of packet classification and optimization. The large size of the policy sets and the complexity due to the varied services imposes tough challenges to non-linear multi-dimensional firewall optimization. The current solutions lack in providing a realistic solution for large data sets, as that of Tier-1 ISPs. Our goal is the design of *dynamic* and *adaptive* optimization techniques that can detect in *real-time* that an attack has occurred or is underway to help mitigate such distributed attacks. To this effect this thesis proposes fundamental research in de-centralized firewall optimization and presents architectures and algorithms to mitigate anomalies over the Internet.

3.0 FIREWALL DATA AND ANALYSIS

To motivate the importance of considering traffic characteristics in firewall optimization, we will analyze the firewall data set and traffic log information from a real world scenario. The data set used in the study is obtained from list-based firewalls managed by a large Tier-1 ISP for its partner networks. The Tier-1 ISP provides secure access to and from about 300 business partners. The data set consists of two parts; the firewall rule-sets and the traffic logs. We have obtained firewall data from six firewalls of the Tier-1 ISP. The analysis is conducted on all six firewalls. In this chapter we present the analysis of a typical firewall data-set. All the other five firewalls follow the similar trends. Firstly, we discuss the various firewall policy representations, we then present the *Tier-1 ISP firewall data set* consisting of firewall rule-set and traffic log information, and finally present detailed analysis of the enterprise firewall data set.

3.1 FIREWALL POLICY REPRESENTATION

Firewall security policies for an organization are expressed in the form of a policy representation set. This representation governs the manner in which rules are invoked during firewall operation. There are various policy representations, namely, *trie-based*, *tree-based*, *direct-acyclic graph (DAG) based*, and *list-based*.

The *trie-based* policy representation is structured as an n-ary (where n is the number of rules in the security policy set) retrieval tree with k levels, also referred to as the *trie*. Each level corresponds to a network attribute and the nodes in the *trie* store the actual value of

the security policy. In the *tree-based* policy representation the rules are structured as a single rooted tree. Each node of the tree represents a network field, and each branch at this node represents a possible value of the associated field. A rule is defined by a tree path starting at the root and ending at the leaf node. Rules that have similar network field value at any given node share the same branch representing that value. As the name suggests in the *DAG based* policy representation the rule-set is modeled as a *DAG*, in which the vertices are firewall rules and edges indicate precedence relationships. A rule is modeled as an ordered tuple of sets, the order is necessary to maintain the predefined semantic order of the original policy set. In the *list-based* policy representation the security policies are in the form of a list and the network packets for filtering traverse sequentially through the list. The focus of this thesis is the *list-based* firewall representation.[1] We will discuss *list-based* firewalls details in Section 4.1.

3.2 FIREWALL DATA

The Tier-1 ISP firewall data consists of *firewall rule-set* and *firewall traffic log* information. The ISP provides service to *300* customer or business partner networks. Partner network services are disjoint from one another. The security policies detailing a partner network security requirements are represented in the form of a *block*. The firewall rule-set consists of a number of blocks, each block corresponding to a customer network as depicted in Figure 4. The security policies within blocks are disjoint from one another. Each block consists of a set of multi-dimensional security policies called the *rules* as illustrated in Figure 5.

A typical block consists of several thousand rules. Each rule is a multi-dimensional structure of tuples. A typical rule on average consists of more than a million tuples. The number of tuples indicate the strength of the data set under consideration. Figure 6 shows an instance of a rule structure. The multidimensional structure of the rule include the source address: *src*, the destination address: *dst*, the service type: *srv*, and the *action* field. The

[1]Most of the ISP firewalls at its core are list-based representations.

action can take values as *accept*, *drop* or $forward$. It is important to note that each dimension contains multiple values. Each such instance of the firewall rule is a *tuple*. We define *tuple* as the fundamental unit of packet filtering of the firewall. A typical tuple instance of a rule is represented as the following:

$< \ src : 10.10.10.2; \ dst : 10.20.10.1; \ srv : ospf; action : accept >.$

The Tier-1 ISP firewall data also consists of the *firewall traffic log* information. Figure 7 depicts an entry of the firewall traffic log. The firewall traffic log information is reported on every action of the firewall. Rule traffic is logged once per entry for every action per session of firewall operation. *Firewall traffic log* includes various packet log information such as the date and time the packet was filtered by the firewall rule, the action taken on the packet (either accept, drop or forward), the source and destination ip address, protocol type, service type, port number, etc. We conduct detailed analysis on the firewall rule-set and traffic log to understand the manageability and performance of the Tier-1 ISP firewall.

The goal of all data analysis and optimizations throughout the proposed research is aimed at improving the operational cost (average processing time) of the tuples for the given firewall rule-set. In the next section we discuss the various factors that can impact the performance of a firewall and conduct analysis on each of them for the Tier-1 ISP firewall data set.

3.3 DATA ANALYSIS

In this section we will detail the various analysis performed on the data set obtained from the large Tier-1 ISP. The aim of this analysis is to understand the management and performance of firewall rule-set and traffic log information. In the first part we conduct detailed

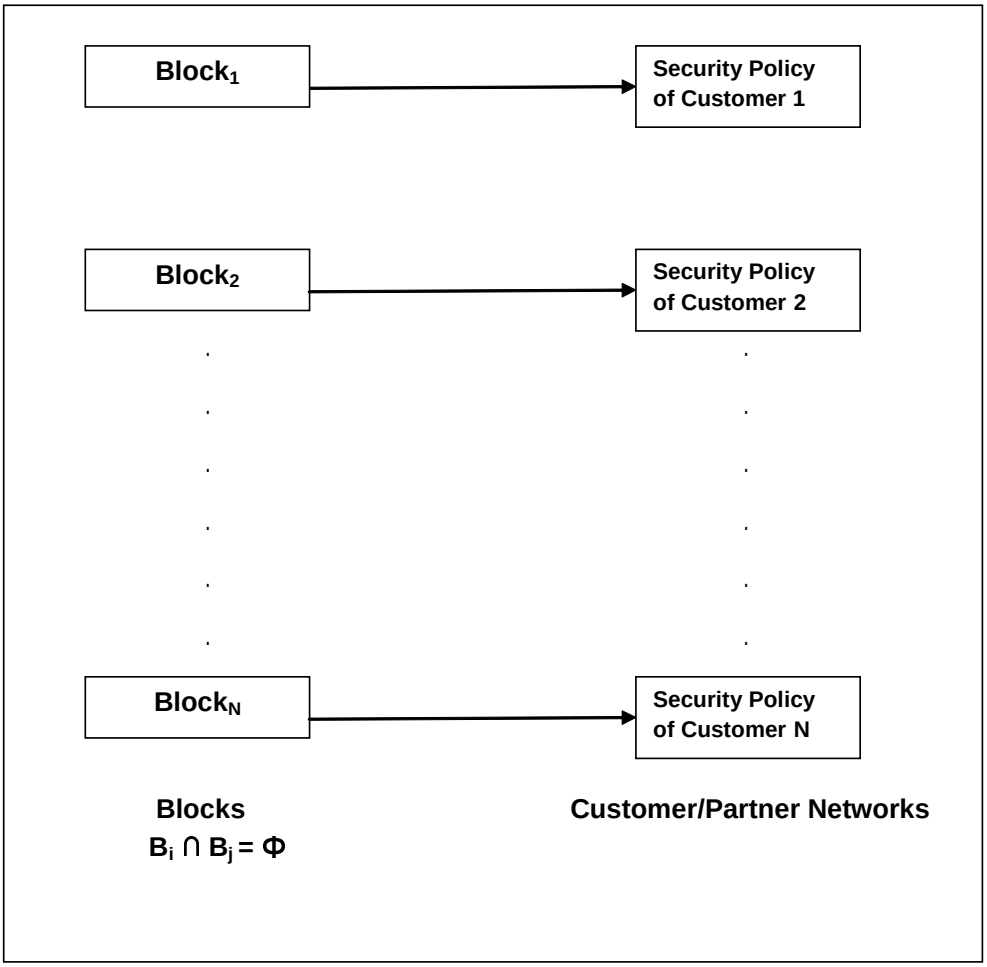

Figure 4: Firewall Structure

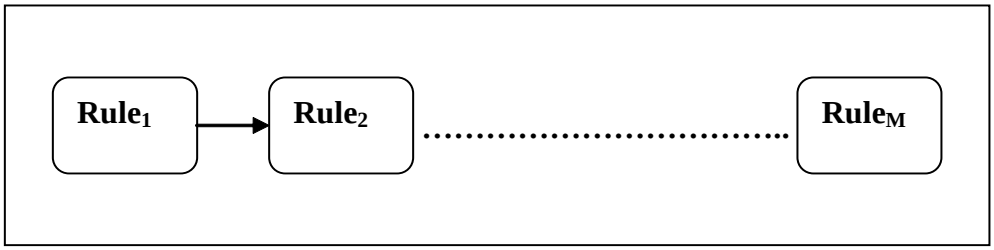

Figure 5: Block Structure

```
: rule (
        : src (                         : srv (
              : 10.10.10.2                    : ospf
              : 10.10.10.3                    : traceroute
              : 10.10.10.4                    : echo-requests
              : 10.10.10.5                    : ping-replies
              )                               )
        : dst (                         : action(
              : 10.20.10.1                     : accept
              : 10.20.10.4                     )
              : 10.20.10.5      )
              )
```

Figure 6: Rule Structure

```
num;date;time;orig;type;action;alert;i/f_name;i/f_dir;
product;src;dst;s_port;service;proto;……..

1;27Jul2005;23:59:04;10.10.10.1;log;accept;;qfe1;
inbound;X;10.30.10.1;10.20.10.1;53480;161;udp;;;
```

Figure 7: Traffic Log Instance

analysis on the firewall *rule-set* data. Later, we perform the *traffic based* analysis on the obtained data set from the large Tier-1 ISP. We present detailed analysis of the results in the following subsections.

3.3.1 Rule-set analysis

3.3.1.1 Block size distribution Each rule-set in the firewall is organized into a set of blocks. Each block of a rule-set corresponds to a given corporate network served by the *Tier-1 ISP*. The number of rules in a block are dependent upon the requirement of the corporate network served by the Tier-1 ISP. Hence, the blocks can consists of non-uniform set of rules. This study is performed to analyze the distribution of rules within the different blocks. We determine ranks based on the occurrence of blocks in the rule-set. Figure 8 shows that the number of rules per block distribution *w.r.t.* the rank of the block.

The results depict statistics of average values for one week of firewall data. It is surprising that blocks do not follow any definite pattern of occurrence. The blocks are appended to the list of firewall rules without any optimization considerations. We would imagine for uniform rule hit distribution the blocks containing higher number of rules should be placed earlier in the rule-set to decrease the operational cost of firewalls. For the realistic case of non-uniform rule hit distribution, the hit distribution should govern the rank of the blocks in the rule-set. The analysis results show the lack of any such optimization in the operation of the Tier-1 ISP firewall.

3.3.1.2 Duplicates amongst blocks The basic functionality of any firewall management system should ensure that each corporate network (client) have disjoint *w.r.t* to other corporate networks or blocks. The disjoincy in the rules ensure that the rules in the blocks do not perform any redundant operation. Figure 9 shows the existence of duplicates amongst blocks in the firewall data of the large Tier-1 ISP.

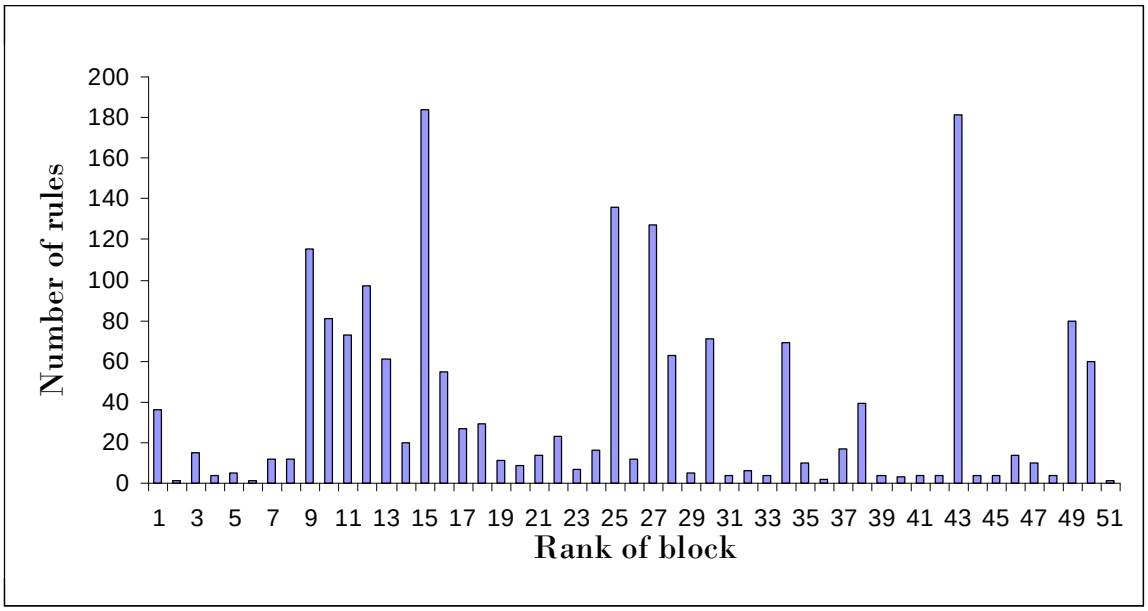

Figure 8: Block size distribution

The redundancies cause the increase in the average operational cost of firewalls, hence affecting the performance of the concerned network system. We have designed our analyzer to be able to identify these rule tuples (src, dst, srv, action pairs) and report them to the firewall administrator for proper action.

Days	Duplicate Count	Duplicate Block
06/06/05-06/13/05	2	16, 46
06/14/05-06/19/05	2	14, 44

Figure 9: Duplicates amongst blocks

3.3.1.3 Rule set variation Another important *rule-set* based analysis is to determine the variation of rules in the rule-set over period of time. We have determined a typical set of data for our analysis. Each rule-set consists of about 2000 rules. The rules are categorized into three categories: "*added*" rules, "*deleted*" rules and "*changed*" rules. A rule is said to be *changed* if there is a change in its rank from one day to another without any change in its contents or if the rule's attributes (such as *src*, *dst*, *srv*) are modified. The modification of rules in the blocks occur as the new rule-set is determined and loaded during routine (daily) operation of the firewall. Results in Figure 10 depict the rule-set variations of typical data set of the Tier-1 ISP.

From the analysis we observe very few rule rank change (movement of rules) based on daily traffic. This observation concludes that most rules follow long term stable patterns. We also observe that there are very few rules that are *added*, *deleted* or *changed* from one day to another. This analysis concludes that the rule changes are stable for most part of the Tier-1 firewall operation.

3.3.1.4 Dependency amongst rules Another important criteria to analyze the firewall rule-set data is to understand the dependency relationships that exists amongst the rules. Comprehending dependency relationships amongst rules will aids to perform traffic based optimization. It is important to note that rule dependencies amongst rules limit the position of rules in the rule sets.

From our analysis we conclude that very few rules are dependent on one another in a given typical firewall rule-set (100 of 1900 rules). Our aim is to detect these dependency relationships amongst rules and eliminate them in order to provide the firewall optimization tool with *full flexibility of rule reordering* at runtime. In the next chapters we will propose techniques to exploit the above rule reordering flexibility to aid traffic aware optimizations.

Change from 06/06/05 to 06/07/05 New Addition: 1311-1322 Deletions: 1311-1320 Changes: 389-390	**Change from 06/12/05 to 06/13/05** No Changes
Change from 06/07/05 to 06/08/05 No Changes	**Change from 06/13/05 to 06/14/05** New Addition: 167-186, 1393-1406, 1625-1770 Deletions: 167-298, 1505-1506, 1725-1869 Changes: 62
Change from 06/08/05 to 06/09/05 No Changes	**Change from 06/14/05 to 06/15/05** New Addition: 168-188, 1279-1283 Deletions: 168-186, 1277-1280
Change from 06/09/05 to 06/10/05 Changes: 67-78	**Change from 06/15/05 to 06/16/05** No Changes
Change from 06/10/05 to 06/11/05 No Changes	**Change from 06/16/05 to 06/17/05** No Changes
Change from 06/11/05 to 06/12/05 No Changes	**Change from 06/17/05 to 06/18/05** New Additions: 170-190 Deletions: 170-188

Figure 10: Rule set variation over days

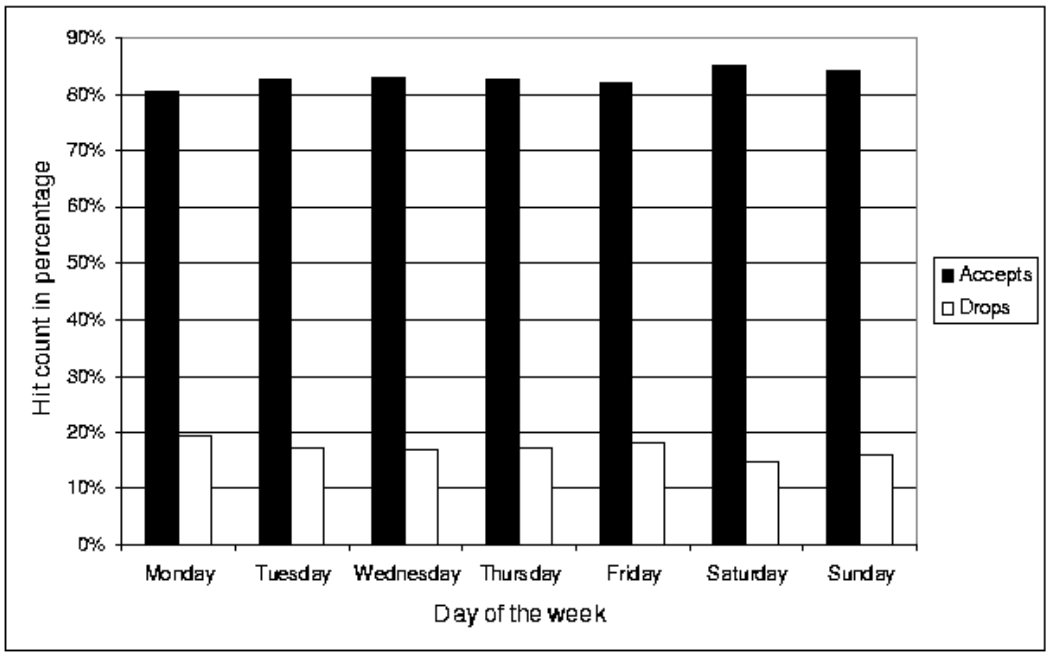

Figure 11: Accepts vs. drop statistics

3.3.2 Traffic log analysis

3.3.2.1 Distribution of Accept vs. Drop rules The next set of analysis are based on the traffic log data obtained from the large Tier-1 ISP. The first study in this category measures the accept and drop hit rates of the firewall traffic over a period of time. It is widely believed that drop rules, especially the default deny reject rule, are the ones which contribute towards the operational cost of the firewall.

Contrary to this belief, results in Figure 11 depict that there is a considerable number of accept and drop rules. The ratio of accept hit rules is on average *3.5* times higher than the drop hit rate. From our analysis we conclude that both accept and drop rules should be considered while performing firewall optimization.

We further analyze the firewall log information to determine the hit distribution of the

accept and drop rules. Our aim in this study is to determine the highest hitting accept and drop rules in each category. Figures 12 and 13 depict the distribution of the accept and drop rules *w.r.t.* to their rank or priority level in the firewall rule-set. The results presented are for a typical week data of the analyzed firewall.

We observe from the results that the rule-set is not optimized *w.r.t.* the traffic it experiences over the network. This adversely affect the operational cost of firewalls. To counter this problem we will present various traffic aware optimizations in the following chapters.

3.3.2.2 Rule hit distribution Another important traffic log analysis is the study of rule hit distribution based on traffic characteristics. Our observation is for typical daily and weekly hit distribution of the Tier-1 ISP firewall. Figures 14 and 15 depict the distribution of top ten hit rules over various interval periods. For optimal firewall operation the rule rank or priority level of rules in the rule-set should be proportional hit distribution of the rules.

From the analysis results, we conclude that heavy hit rules are not appropriately ranked in the rule-set. In the proposed research we suggest to reorder the rules based on their hit-counts to improve the performance of the firewall. From the study of the results we propose that heavy hit rules, which have lower ranks, should be assigned higher ranks. For example, in Figure 15, upon reordering, rule ranked 115 should be ranked 2, to improve the average operational cost of the Tier-1 ISP firewall.

3.3.2.3 Default deny rule hits Default deny rules are those that do not match any pre-defined rules in the operational rule-set of a given firewall. The terminal rule of a firewall is usually a default deny action to eliminate all non-match rules after they have traversed the entire list in a list-based firewall. All attack traffic cause the firewall to invoke the default deny, as they cause the firewall to over work on searching for a match in its entire rule-set. The attacker plans to send high volume of traffic that do not match the firewall

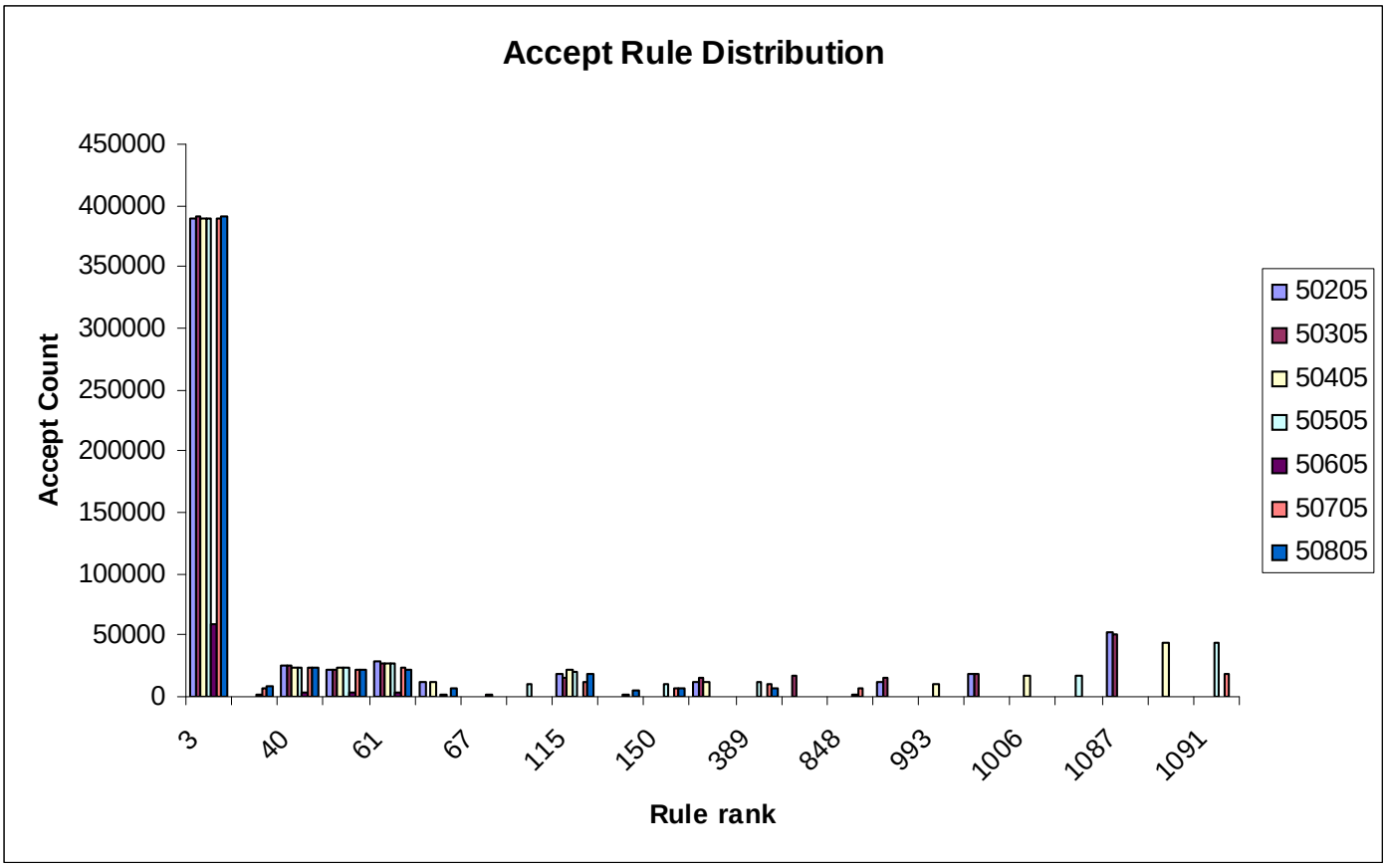

Figure 12: Accept rule distribution

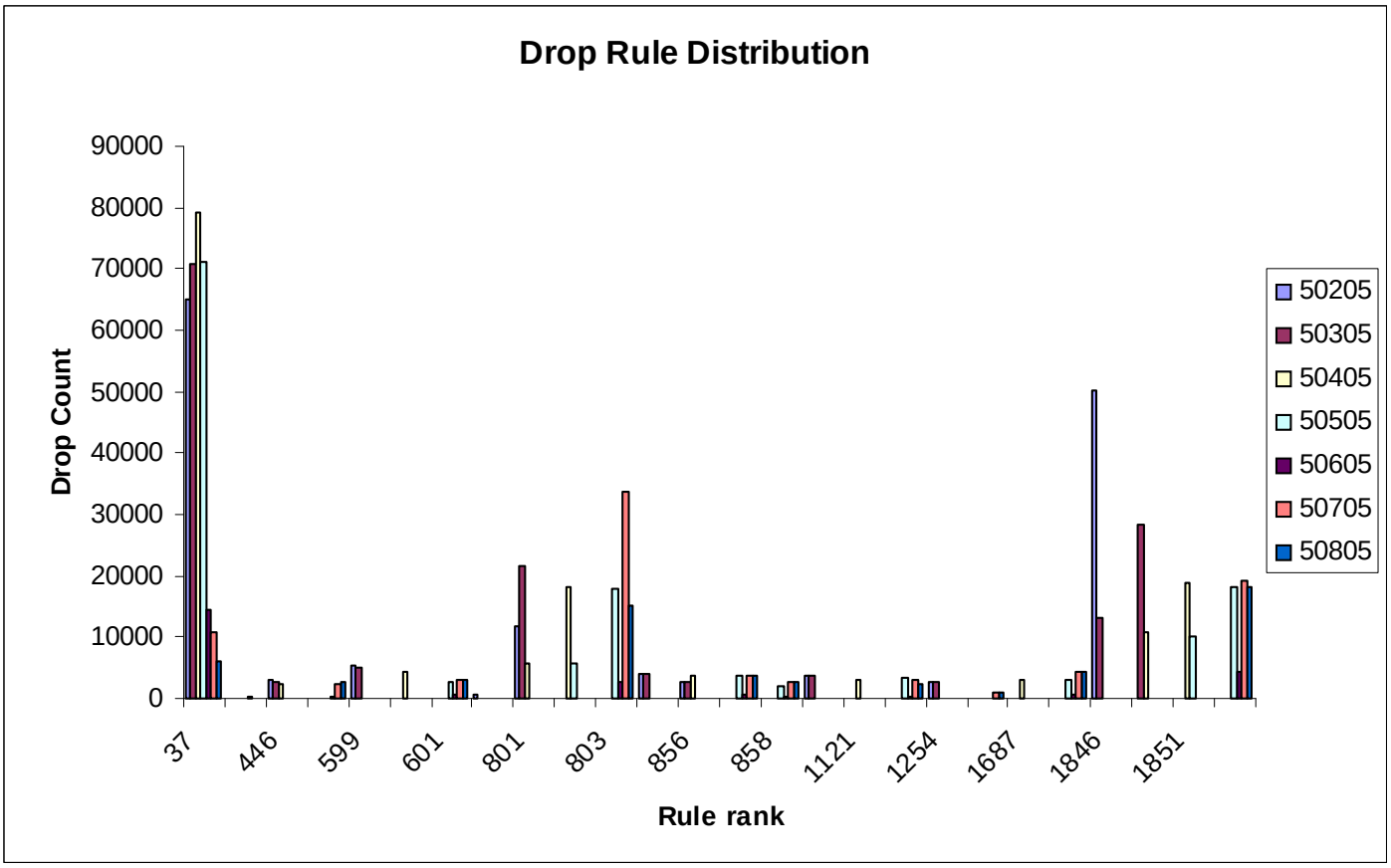

Figure 13: Drop rule distribution

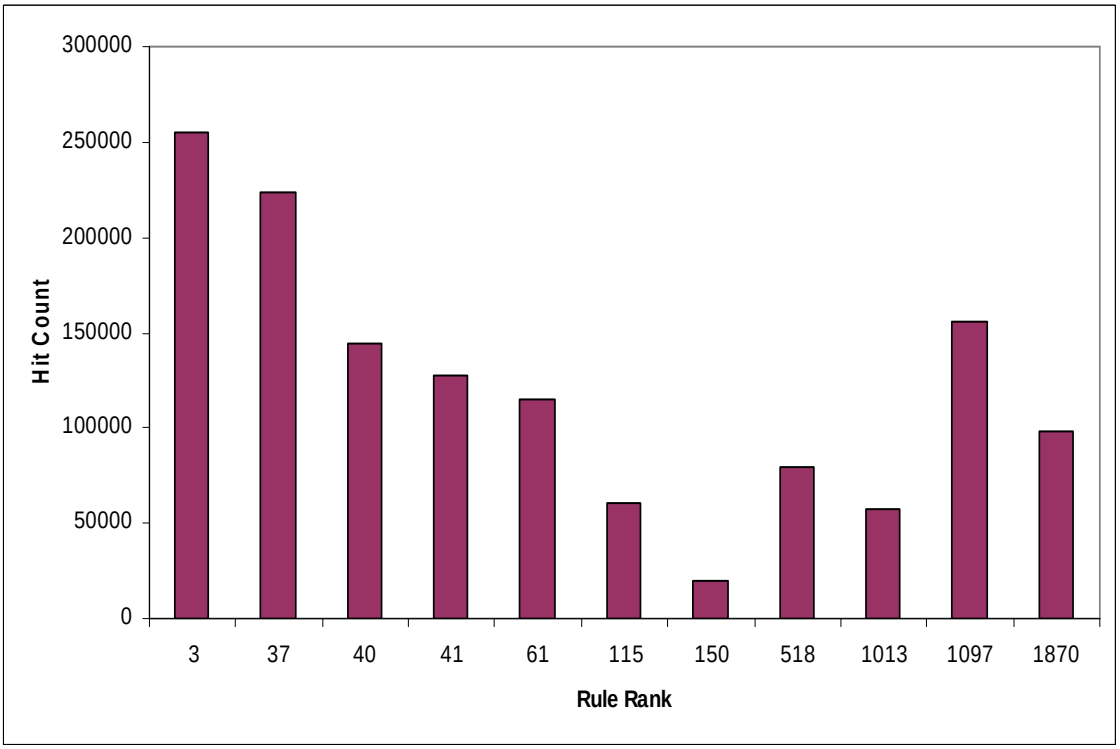

Figure 14: Rule hit distribution: Over weeks

definitions, but cause the unnecessary work in the firewall. These attacks cause the increase in the average operational cost of firewalls, in turn making the firewall a bottleneck system. To understand the severity of the default deny action,we analyze the hit distribution of the rules in a typical day for given firewall rule-set. As these attacks are usually short term and transient, we observe the hit distribution over hours for a typical day. Results in Figure 16 depict that hit-count information of the default deny rules in a given firewall rule-set.

We conclude from our analysis that the current firewall operation is heavily bottle-necked due to these anomalous conditions. To understand the problem better we take a closer look at firewall data for *3* months. From our observation we conclude that on average about *65%* of the tuples are consistently repeated in the default deny hit tuples set. This shows that there is a large number of consistent default deny hit tuples. We will propose various techniques in the following chapters to eliminate these anomalous traffic early in the firewall rule-set operation.

3.3.2.4 Field count distribution In this study we aim to analyze the field count distribution in the traffic log information of the Tier-1 ISP firewall. As we have mentioned earlier, traffic is logged on every action of the firewall. The log information helps to capture the traffic characteristics during the operation of the firewall. Any optimization based on traffic log will be limited by the size of the above logged information. Results in Figure 17 depict the use of only *"16"* out of *"121"* traffic log fields.

From the above results we conclude the presence of unnecessary huge overhead of maintaining counters for unused fields. We propose to remove the unused traffic log fields in the definition of log information in order to aid traffic-aware firewall optimization.

3.3.2.5 Protocol distribution The final traffic log analysis is to analyze the distribution of protocols in the Tier-1 ISP firewall. Figure 18 depicts the protocol distribution for a

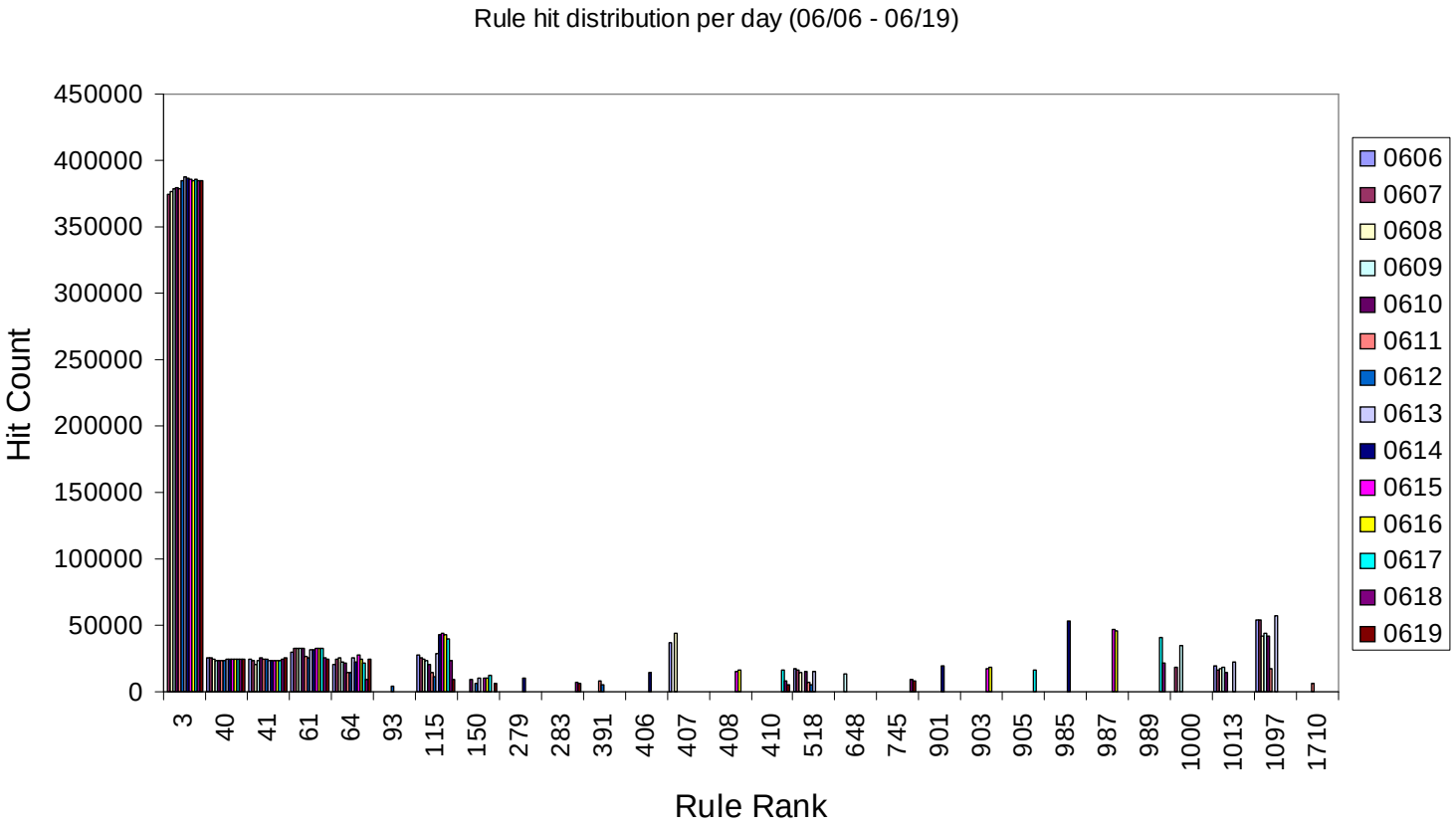

Figure 15: Rule hit distribution: Over days

typical week of the firewall. From our the results we conclude that *ICMP*, *TCP* and *UDP* form the major protocols defining most of the traffic. From the analysis results, we propose that rules using these protocols should be placed earlier in the rule-set to improve the average rule processing time in the firewall. We believe that these statistics must be taken into consideration in rule priority setting to improve the operational cost of firewalls.

The outcome of the above studies clearly illustrates the fact that considering traffic characteristics in the optimization of firewalls is crucial in achieving significant improvement in the performance of such network systems.

3.4 SUMMARY

In this chapter we present extensive data analysis of the firewall data set and traffic log information of the Tier-1 ISP. The analysis and observations helps to establish the fact that consideration of traffic information is vital to the design of efficient firewall operation. Based on the above analysis, the proposed research aims at designing architectures and algorithms towards efficient management and optimization of firewalls *via* traffic-aware approaches. We strongly believe that such traffic-aware techniques would impact the performance, availability and security of today's networked systems. To summarize, the goal of this research is to address the problem of firewall optimization by designing tools to enhance the capability of firewalls to manage traffic volume and dynamics efficiently along with providing high system availability. Furthermore, the research intends to design novel strategies to efficiently deal with various network anomalies *(DDoS, etc)* in a *proactive* manner.

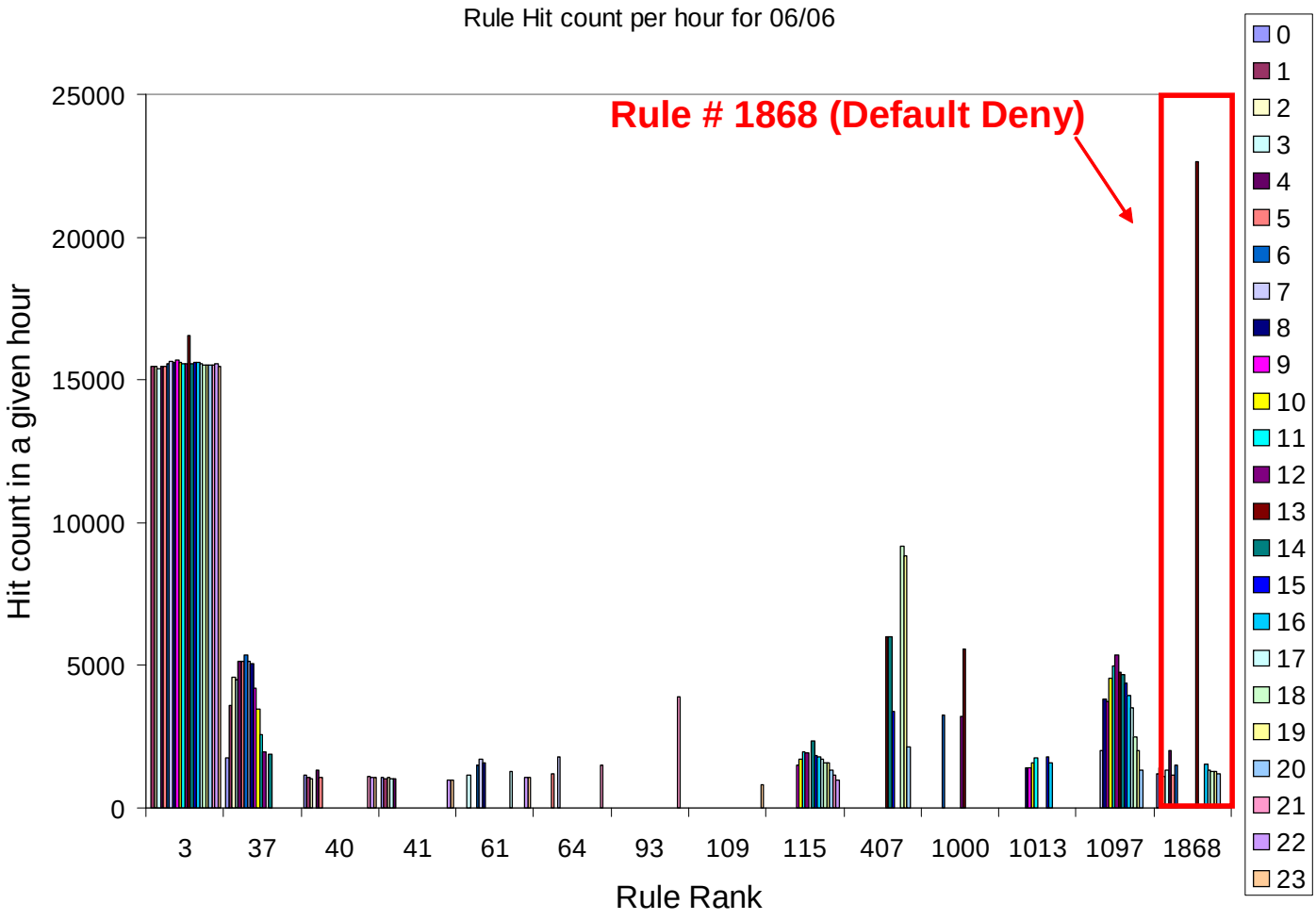

Figure 16: Default deny rule hits

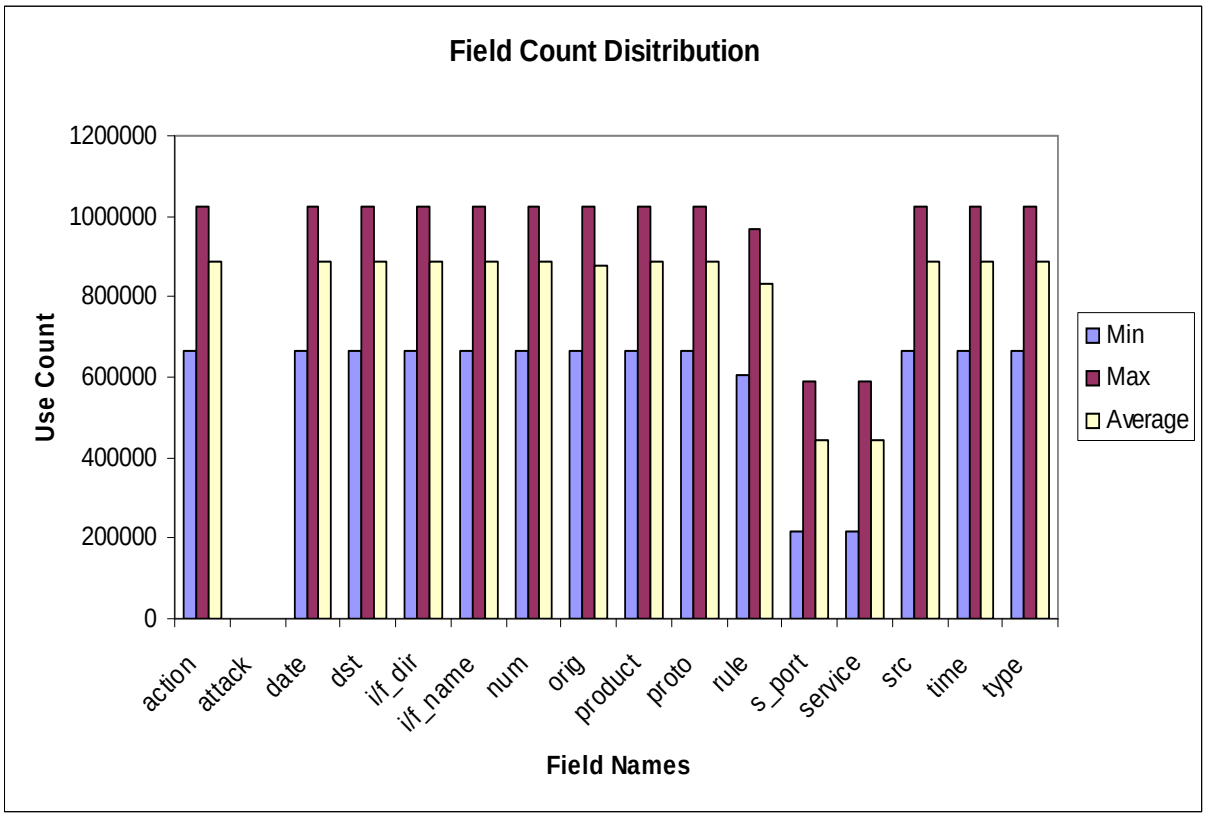

Figure 17: Field count distribution

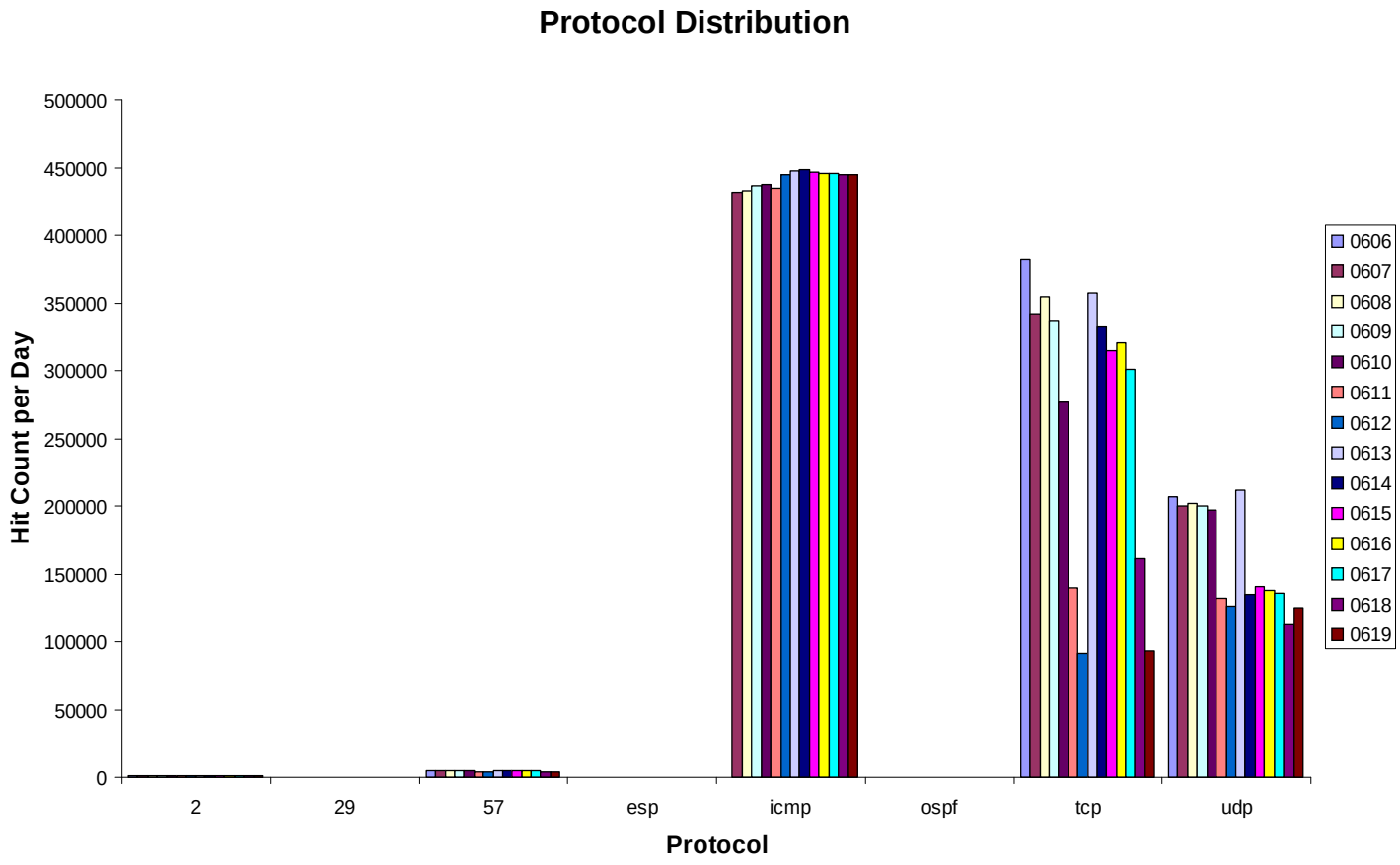

Figure 18: Protocol distribution

4.0 PITTWALL: A CENTRALIZED FIREWALL OPTIMIZATION APPROACH

In this chapter we presents the architectures and algorithms for a centralized firewall optimization toolset: *PITTWALL* [7]. Our research aims at understanding the problem and challenges towards firewall optimization via an in-depth study of firewall rules (packet filters) and daily firewall logs (traffic statistics) obtained from a large *Tier-1 ISP*. The analysis helps to understand the problem of firewall optimization. Further, it lays the foundation towards the design of efficient architectures and algorithms for managing and optimizing such network defense systems (firewalls). Moreover, the firewall data analysis in the previous chapter also supports the claim towards the design of a proactive security model to detect and defend against attacks. We will first present details of *list-based* firewall representation in Section 4.1, which is the focus of this thesis. We introduce the *firewall optimization model* in Section 4.2. The details of the metrices are discussed in Section 4.3. Finally we conclude the chapter with the evaluation study in Section 4.4.

4.1 LIST BASED FIREWALLS

In this thesis, we concentrate on "*List based*" firewalls, one of the widely used firewalls for most large *Tier-1 ISPs*. List-based firewall structures rules and policies in a "priority" list. The list contains the security policies that govern the network packet filtering process in the operation of the firewall. The set of rules in the firewall set are referred to as the firewall "rule-set". The priority of a rule or a policy in the firewall "rule-set" is based on its position in the list. Earlier occurring rules have higher priority than later ones and are enforced first.

Subsequently, the packet is filtered by the earliest rule or policy that matches the filter definition. This is referred to as the *first hit principle*[1]. In the following subsection we present the *list-based* policy representation which is the focus of this thesis work.

A firewall security is typically defined by a set of rules or filters. In this section we will detail the most widely used firewall representation, *List Based Firewalls* which is the focus of this research[2].

A firewall rule is a multi-dimensional structure, where each dimension is either a set of network fields or an action field. A network field can be *source address*, a *destination address*, a *service type*, a *protocol number* or a *port number*. An action field can be either *accept* or *deny*, or *others* (e.g. redirect to a server that perform further processes etc). Formally, a rule R can be represented as: $R = [\Phi^1, \Phi^2, \cdots, \Phi^k; \Sigma]$, where Φ^j, represents network fields and Σ is an action field. In an Internet environment, a typical rule can be represented as follows:

$$< src = \{s_1, s_2, \cdots, s_n\};\ dst = \{d_1, d_2, \cdots, d_m\};$$
$$srv = \{\sigma_1, \sigma_2, \cdots, \sigma_l\};\ action = \{drop\} >,$$

where s_i represents a source IP address, d_i a destination IP address, and σ_i a service type. In list-based firewalls, rules describing the network security policies form a "priority" list. The priority of a rule, also referred to as its *rank*, is based on its position within the list. Earlier occurring rules have higher rank than later ones.

List-based firewalls work by logically examining the rules in sequential order. For each packet, the first matching rule determines the action taken by the firewall. This is referred to as the *first hit principle*. It is to be noted that not all firewalls work with the first hit principle, also there are list-based firewalls which do not use first hit principle. In the next section we will discuss the challenges towards managing and optimizing firewalls for current N/W systems.

[1]It is to be noted that all firewalls do not work with the first hit principle. Majority of *Tier-1 ISP* firewalls, which is the focus of this thesis, follow the *first hit principle*.

[2]Most of the ISP firewall representations at its core are list-based

4.2 FIREWALL OPTIMIZATION MODEL

Firewall policies of an actively managed enterprise network may often change in response to new services, new threats or when the underlying network changes. The intrinsic complexity of the firewall policies makes it difficult to track down these changes. As a consequence, inefficiency, such as redundancies between rules, and suboptimal representations of rule-sets and fields within a rule, arise. Furthermore, the *availability* and *goodput* of the networked system is greatly hindered due to the inefficiency in rule representation and filtering. To this effect, the proposed research attempts to design a novel optimization model that involved traffic characteristics to aid efficient firewall optimization. Furthermore, the research presents a dynamic proactive security model to detect and defend against attacks by anomaly detection and countermeasure, thus improving the overall performance of such systems. In the next subsections we will detail the methodology and the steps of the proposed optimization framework. The overall architecture of this optimization framework is depicted in Figure 19.

The core component of the optimization process uses a "*rule-set based*" optimizer and a "*traffic based*" optimizer. Both optimizers cooperate to adaptively optimize the rule-set in response to dynamically changing traffic characteristics. This cooperation is achieved through a dynamic feedback mechanism. The rule-set based optimizer takes as input the pre-optimized rule-set and produces a rule-set based optimized set of rules. This set is then fed to the traffic based optimizer. Using the current traffic log, the traffic based optimizer produces an optimum rule-set which reflects the current characteristics of the traffic without violating the semantic integrity of the initial rule-set. The traffic-aware optimized rule-set is used by the firewall to enforce the security policy. This continues until changes in the traffic characteristics take place. In response to these changes, the adaptive optimization process is re-invoked using the current rule-set and a new traffic-aware optimized rule-set is produced. This process continues iteratively, until the enterprise network security administrator changes the rule-set. When this occurs, the new rule-set is pre-optimized before the rule-based and traffic-based optimizers are invoked. In the following subsections, the basic optimization steps, along with their collaborative interactions, are discussed in detail.

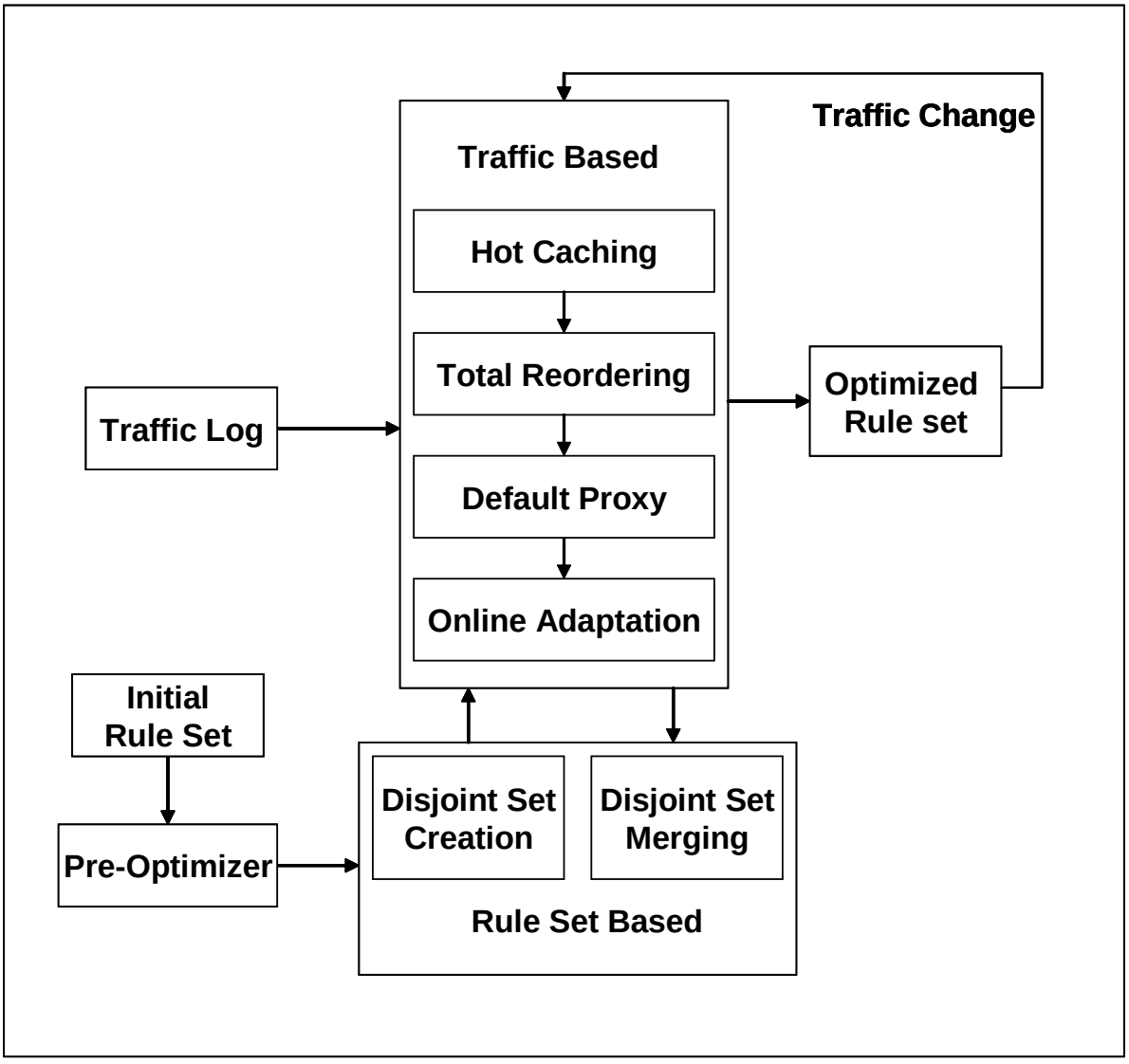

Figure 19: Firewall Optimization Framework

4.2.1 Stage I: Pre-optimization

The process starts with the *Pre-optimization* phase. The main objective of this phase is to remove all redundancies in the rule-set. At the end of this phase, all internal and external redundancies in the rule-set are removed. The details of rule redundancies with examples from firewall rule filters are discussed in [6].

4.2.2 Stage II: Rule-set based optimization

The rule-set based optimizer operates exclusively on the rule-set, with no additional consideration of other factors impacting network or traffic behavior. The optimizer continuously seeks to create new definitions in order to make rules in the current rule-set disjoint. This, in turn, provides the traffic based optimizer with full flexibility to reorder rules based on traffic characteristics.

The rule-set based optimizer is composed of two basic components, namely the *Disjoint Set Creator (DSC)* and the *Disjoint Set Merger (DSM)*. These two components are typically executed sequentially. Initially *DSC* detects and removes dependencies from the current rule-set. Then it creates new rule definitions in order to make the entire rule-set disjoint. It is to be noted that this phase may lead to an increase in the rule-set size. This is due to the fact that more rules may be needed to define each set of dependent rules. It is typical that there is only a small portion of rules that are dependent on other rules. In the analyzed firewall data set, this ratio is around $(1/15)^{th}$ of the total number of rules.

The main task of *DSM* is to merge the rules of the disjoint rule-set produced by *DSC* in order to optimize the rule-set representation. The merging process iteratively selects one rule and tries to merge it with other rules. Merging occurs between rules with same action field, to preserve semantic integrity. Merging between two rules, with respect to a specific different field, occurs when the other corresponding field values are same in the field space. Upon completion of this optimization step, the rule-set size is reduced to its most concise representation.

Notice that it is possible to reduce the rule-set based optimization strategy to rule merging only, without the creation of disjoint rules. Such an approach still results in improved rule-set representation, while minimizing the processing overhead. Combining disjoint set creation and merging, however, enables the optimizer to effectively capture the dynamics of the traffic characteristics, thereby resulting in an optimized rule-set representation. It is important to note that *DSC* enables full flexibility of rule reordering during online adaptation based on the dynamics nature of the incoming traffic to the firewall. As we will discuss in Chapter 5 making the rules disjoint from one another also aids de-centralized firewall optimization.

To illustrate the process of creation of disjoint rules out of an initial pre-optimized set of rules, and merge the resulting disjoint rules into a concise rule-set representation, consider the example of a pre-optimized rule-set, $\mathcal{S}_{\mathcal{I}}$, as shown in Table 1.

Notice that R_2 is dependent on R_1, since the source and destination fields of R_2 intersect with the corresponding fields of R_1, while the action fields of the two rules are different. These rules can be made disjoint, without violating semantic integrity. This is achieved by keeping R_1 unchanged and forking R_2 into two new rules, R_2^1 and R_2^2, resulting in the disjoint rule-set, $\mathcal{S}_{\mathcal{D}}$, as shown in Table 2.

As observed from the above example, creating a new disjoint rule-set increases the size of the original rule-set. The new set size can be further optimized by merging rule R_2^2 and R_3 into R_4, to produce the final rule-set, $\mathcal{S}_{\mathcal{F}}$, as shown in Table 3.

4.2.3 Stage III: Traffic based optimization

The traffic based optimizer operates on the rule-set produced by the rule-set based optimizer. The optimizer uses current traffic characteristics to determine the order in which rules in the rule-set are to be invoked to optimize the operational cost of the firewall. To achieve this

Rule	Src	Dst	Srv	Action
R_1	s_1, s_2, s_3	d_1, d_2, d_3	σ_1	drop
R_2	s_2, s_3, s_4	d_2, d_3, d_4	σ_1	accept
R_3	s_5	d_4	σ_1	accept

Table 1: Pre-optimized rule-set: $\mathcal{S}_{\mathcal{I}}$

goal, we have designed four schemes, namely *hot caching*, *total reordering*, *default proxy*, and *online adaptation*.

4.2.3.1 Hot caching *Hot caching* revolves around the concept of a *hot rule-set*. A rule is said to be *hot* if it experiences a large number of traffic hits. The basic idea of this approach is to identify a small set of very heavily used or *hot* rules, relative to the original rule-set, and cache these rules at the top of the rule-set. Such a strategy results in dealing with a large amount of traffic hits, very early in the inspection process, thereby reducing the overall firewall operational cost. It is to be noted that we are able to perform *hot caching* due to the removal of dependencies amonsgt the rules during *Disjoint Set Creator*. This scheme is supported by the fact that in such network systems 80% of the traffic is filtered by 20% of the rules.

4.2.3.2 Total reordering Contrary to the first scheme which focuses only on a small set of rules, the *total reordering* scheme takes a more aggressive approach and performs a total reordering of the rule-set based on the current traffic characteristics. This *reordering* is achieved based on a priority assignment which takes into consideration, not only the frequency at which the rule is invoked, but equally importantly the rule size. More specifically, the priority of rule, R_i, can be expressed as: $Pr(R_i) = \frac{hit\_count(R_i)}{size(R_i)}$. Notice that ordering firewall rules based on the above priority assignment achieves the lowest expected cost [6].

Rule	**Src**	**Dst**	**Srv**	**Action**
R_1	s_1, s_2, s_3	d_1, d_2, d_3	σ_1	drop
R_2^1	s_4	d_2, d_3, d_4	σ_1	accept
R_2^2	s_2, s_3	d_4	σ_1	accept
R_3	s_5	d_4	σ_1	accept

Table 2: Disjoint rule-set: $\mathcal{S}_{\mathcal{D}}$

Rule	**Src**	**Dst**	**Srv**	**Action**
R_1	s_1, s_2, s_3	d_1, d_2, d_3	σ_1	drop
R_2^1	s_4	d_2, d_3, d_4	σ_1	accept
R_4	s_2, s_3, s_5	d_4	σ_1	accept

Table 3: Final rule-set: $\mathcal{S}_{\mathcal{F}}$

4.2.3.3 Default proxy The *default proxy* is the third scheme and is based on the observation that, during traffic inspection, the default deny action is heavily invoked, in comparison to actions resulting from other rules. In a list-based firewall, the default deny action is "enforced" when a packet fails to match any of the rules within a rule-set. A relatively high hit ratio of the default deny action is, therefore, bound to increase considerably the overall operational cost of the firewall. The main reason for this increase is that, before a default deny action is enforced and the packet is dropped, all rules in a rule-set have to be examined. This is mainly caused by the absence of any representation of the default deny action in the rule-set. This, in turn, suggests that the addition of drop rules may alleviate the problem. Adding drop rules, however, brings about several issues, including how may rules must be created, what values should be associated with these new reject rules and what should be their priorities.

The *default proxy* scheme addresses these issues by creating a set of reject rules. The field values of these rules are derived from the corresponding fields of the packets dropped by the default deny action. Initially, the fields of a reject rule are set to *any*, except for the action field which is set to *drop*. The reject rule can be represented as:

$$<\Phi^1 : any;\ \Phi^2 : any;\ \cdots, \Phi_n : any;\ action = drop>$$

As packets are dropped by default deny rule, the values of the reject rule are set to the values of corresponding fields of the dropped packets. This corresponds to the hit rate of the reject rule. The priority of each newly created reject rule is computed based on its hit rate and size similar to the process as in total reordering.

4.2.3.4 Online Adaptation This optimization is a proactive security measure to improve the availability of firewalls under dynamically changing network environment and The *online adaptation* scheme encompasses two basic mechanisms: *profile based reordering* and *anomaly detection and countermeasure*.

Profile based reordering uses traffic characteristics to build a long-term rule hit profile during offline operation. The approach used to build this profile exploits traffic variability as discussed in [43]. The resulting rule hit profile is then used to detect long and short term anomalies and adapt the rule-set accordingly during online operation of the firewall.

The basic idea of *Anomaly detection and countermeasure* is to compare the short term traffic pattern with a long term established traffic profile. The later is used to optimize the firewall rules. If a significant discrepancy exists between the short term traffic pattern and long term profile, and this discrepancy can result in bad predicted performance, the rules are adjusted as a countermeasure against anomalies. Adjusting the rules entails rule reordering and adding explicit reject rules.

Note that anomalies can be either *transient* or *long-lived*. If the anomaly analysis reveals a potential performance hazard, a temporary reordering of rules is performed. If a given anomaly occurs consistently then it is absorbed into the long term offline profile. The same anomaly detection and countermeasure procedure is also applied to the default deny rule. Depending on any potential performance hazard created by a default deny rule, a temporary default deny rule is added to the short term profile. If the pattern is repetitive then the new default deny rule is added to the rule-set based on its priority and hence absorbed into the long term profile.

4.3 THEORY: RULE SIZE AND COST METRIC

The main factor that affects the performance of a firewall is the processing overhead due to packet inspection. The metric calculation is performed for a rule and can easily be applied to tuples within a rule.

We define two metrices to capture the overhead cost incurred by a firewall to process a rule and enforce the security policy. The first metric, denoted as *rule_size()*, measures the size of a given rule in terms of the number of bits necessary to determine unambiguously a match between the rule definition and the corresponding fields of a packet under inspection. The assumption underlying the *rule_size()* metric stems from the fact that the complexity of a matching operation is proportional to the size of the rule. Formally, given a rule r, rule_size(r) can be defined as:

$$rule\_size(r) = \begin{cases} \sum_{\mathcal{S}_p,\mathcal{D}_p}\{\alpha_1 \times \|s_p\| + \alpha_2 \times \|d_p\|\} \\ +\beta \times N_s \times (\|Pr_r\| + \|Po_r\|), \end{cases}$$

where, α_1, α_2, and β are weight parameters, $\mathcal{S}_p$ and $\mathcal{D}_p$ are respectively the set of source and destination prefixes which occur within the definition of the rule, s_p and d_p are the bit representation of the source and destination prefixes, respectively, N_s is the number of

services defined within the rule, and Pr_r and Po_r are the bit representation of the protocol and port identifiers, respectively.

The second metric used in our experimentation is the cost of operating a given rule-set. This cost depends on the rule's rank and size, and on how often the rule is invoked by the firewall. Formally, given a set of rules $r_1, r_2, \ldots, r_k$, the cost of a given rule, r_i, $cost(r_i)$, is defined as follows:

$$cost(r_i) = \ hit\_count(r_i) \ \times \ \sum_{\forall r_k \in \mathcal{P}r_i} \|r_k\|$$

where, $\mathcal{P}r_i$ is the set of r_i's predecessors in the list-based set of rules.

Using the above metrics, the aim of optimization is to reduce the rule-set size and consequently the processing time of the rule-set. This in turn reduces the overall firewall operational cost. The resources that affect are the CPU utilization and the memory usage of the firewall machine.[3] In the following subsection we present the proof of the optimality of the metric *via* contradiction.

Theorem 4.1. *Firewall rules in a list-based ordering based on priority achieves the lowest expected cost.*

Proof. Assuming a rule-set in the order of $r_1, r_2, \ldots, r_n$
achieves the lowest expected cost and priority $(r_i) < priority(r_{i+1})$, the total cost associated with this rule order is

$$Cost = \sum_{t=1}^{n} cost(r_t) = \sum_{t=1}^{n} f(r_t) \sum_{k=1}^{t} ||r_k||$$

We now show that by swapping r_i and r_{i+1}, the new rule order has a lower cost. The function $f()$ defines the rule hit frequency. This is proportional to the hit count of the rule. The rule hit frequency is denoted by the function $f()$. The total cost associated with the new rule order is

[3]The metric used in this research follow the same guideline as in [7].

$$
\begin{aligned}
Cost' &= \sum_{t=1}^{i-1} cost(r_t) + cost(r_{i+1}) \\
&\quad + cost(r_i) + \sum_{t=i+2}^{n} cost(r_t) \\
&= \sum_{t=1}^{i-1} cost(r_t) + f(r_{i+1})(\sum_{k=1}^{i-1} ||r_k|| + ||r_{i+1}||) \\
&\quad + f(r_i)(\sum_{k=1}^{i} ||r_k|| + ||r_{i+1}||) + \sum_{t=i+2}^{n} cost(r_t) \\
&= Cost + f(r_i)||r_{i+1}|| - f(r_{i+1})||r_i||
\end{aligned}
$$

Since $priority(r_i) < priority(r_{i+1})$, we have

$$
\begin{aligned}
\frac{f(r_i)}{||r_i||} &< \frac{f(r_{i+1})}{||r_{i+1}||} \\
f(r_i)||r_{i+1}|| &< f(r_{i+1})||r_i|| \\
f(r_i)||r_{i+1}|| - f(r_{i+1})||r_i|| &< 0
\end{aligned}
$$

We have $Cost' < Cost$. Thus the new rule order has a lower cost than the earlier rule order. This contradicts our assumption that the first order is optimal. We arrive at a contradiction and thus prove the theorem. □

4.4 EVALUATION

4.4.1 Firewall Optimization

In this section we discuss the performance of the proposed Centralized Firewall Optimizer and the discuss the results of the evaluation study. The details of the metrices designed for the evaluation are presented in Section 4.3.

In order to evaluate the impact of the various optimization strategies on the firewall performance, an experimental simulation-based study is conducted. The simulation is run on *SunOS 5.8* over a *Sun-Fire-15000*. Results show that the optimization strategies lead to considerable firewall performance improvement.

4.4.1.1 Rule-set based optimization The results depicted in Figure 20 show that the final rule-set size after optimization is similar to the size of the initial rule-set, but most importantly, the rules in the resulting rule-set are all *disjoint* from each other. Rule-set disjoincy helps to provide the network system administrator full flexibility to reorder the rules based on traffic characteristics, as is necessary to respond to the traffic dynamics in such network systems.

4.4.1.2 Traffic based optimization In this experiment, traffic based optimization are applied to the firewall rule-set. Results in Figure 21 demonstrate a significant decrease in the number of rules. More specifically, the results show that nearly *20%* of initial operational rules are eliminated.

The final experiment is aimed at evaluating the impact of the various optimization strategies on the operational cost of firewalls. The results depicted in Figure 22 indicate that the optimization strategies, applied to the pre-processed dataset, result in reducing the initial operational cost to around *6.3%*.

The evaluation study clearly indicate that the proposed traffic-aware optimization strategies have great potential to significantly improve the performance of firewalls and reduce their operational cost. A more extensive analysis of the schemes and experimental results can be found in [6].

Furthermore, the study also confirms the importance of integrating traffic characteristics into the optimization process. Finally, it should be noted that adaptive anomaly detection is crucial in preventing and eliminating attacks, and avoiding bottle-necked firewalls. The

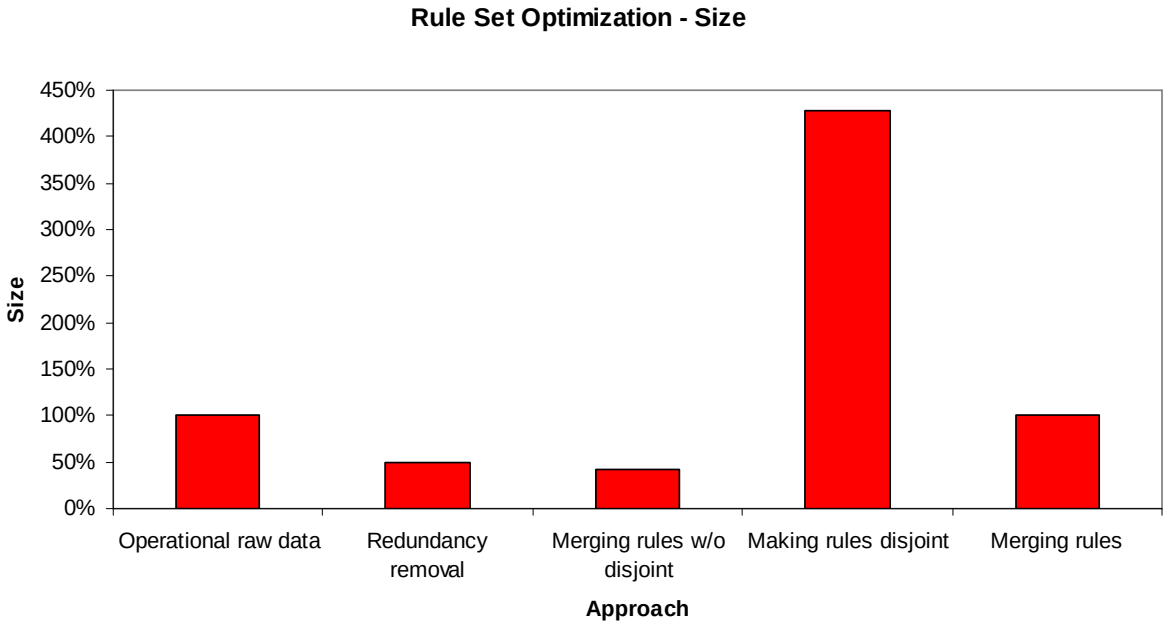

Figure 20: Rule Set Based Optimization: Size-based

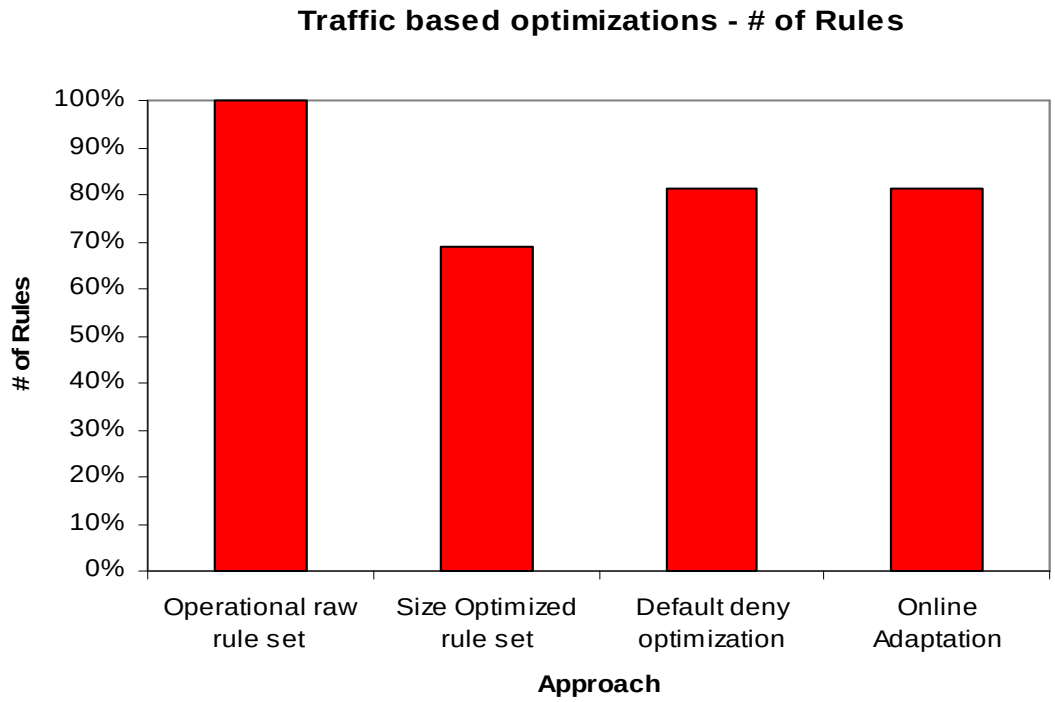

Figure 21: Traffic Based Optimization: Size-based

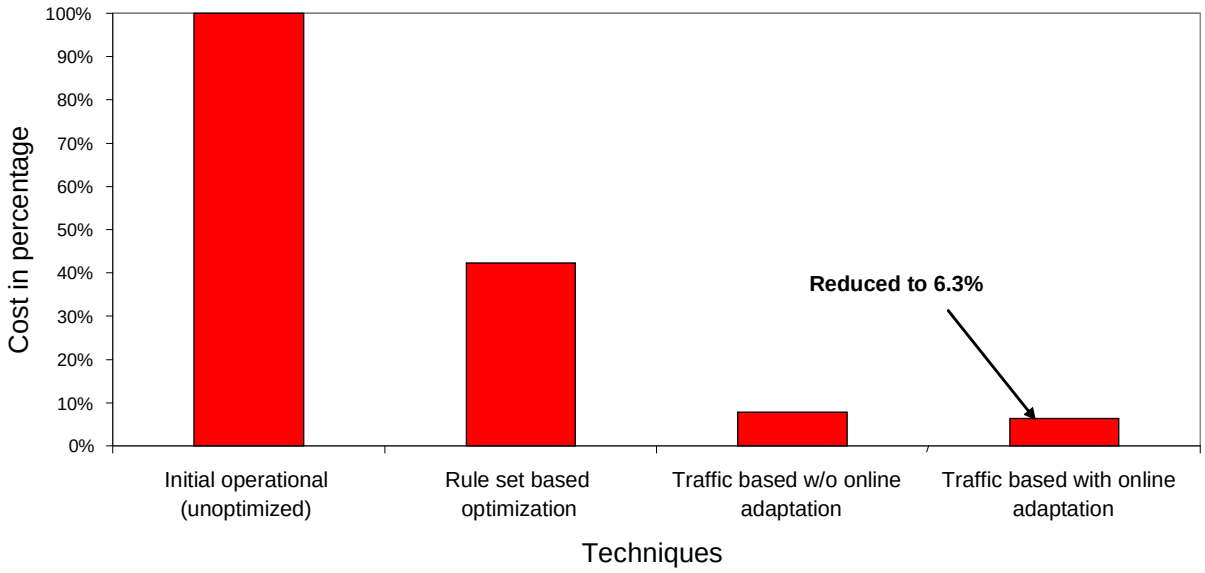

Figure 22: Traffic Based Optimization: Cost-based

above proposed approaches helps to maintain availability of network defense systems during critical conditions *(overload/attack)* to ensure the design of secure networks.

4.4.2 Online Adaptation

In this section we detail the evaluation of the online adaptation module for the proposed Centralized Firewall Optimization approach.

4.4.2.1 *Benefit/Cost* evaluation It is important to estimate the *Benefit/Cost* ratio for invoking the online optimization module in *PITTWALL*. In our experiments we evaluate the ratio for the *emulated*(*average*) case scenario. The firewall used for the experiment is open source *Linux IP Chains*. The rule-set and traffic load for the evaluation is emulated rule-set and traffic log information from a large *Tier-1 ISP*. It is to be noted that this evaluation is workload dependent. The traffic load is at the rate of *7000* packets/sec. The results are validated over *20* runs of the experiment.

The results depicted in Figure 23 demonstrates that an adaptation interval of *75* minutes is best (taking into account the benefit-cost ratio) for a typical packet filter set with the typical traffic load. Results will vary with the variation of filter sets and traffic characteristics. The evaluation acts as feedback to the network designer to prevent over engineering the adaptation interval beyond the point of diminishing returns.

4.4.2.2 Determining best Adaptation Interval This evaluation is for the *emulated (average)* case scenario. The firewall used for the experiment is open source *Linux IP Chains*. The rule-set and traffic load for the evaluation is emulated rule-set and traffic log information from a large *Tier-1 ISP*. The traffic load is at the rate of *7000* packets/sec. The results are validated over *20* runs of the experiment. The result for the best adaptation interval depends on the workload under evaluation.

The results depicted in Figure 24 show that an adaptation interval of *3* hours is the best

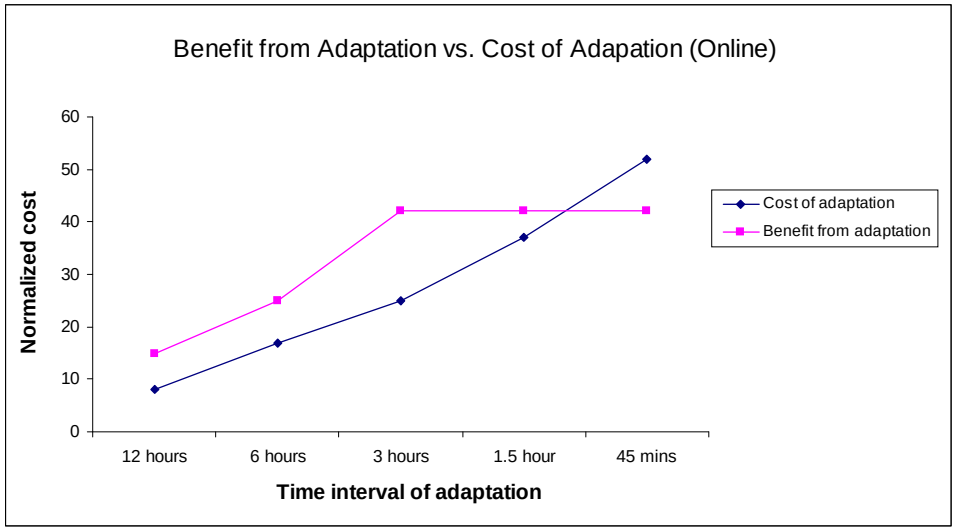

Figure 23: Online Adaptation Benefit/Cost Curve

choice for the above test case scenario. The average rule processing time is the least in this case. The result is an indication for setting the adaptation interval for executing the online optimization. The most suited interval length will change as the packet filter set changes and/or with dramatic changes in the in-coming traffic into the intrusion detection system.

4.4.2.3 Benefit of adaptation with attack traffic This evaluation is for the worst case scenario of firewall operation. The firewall used for the experiment is the open source *Linux IP Chains*. The rule-set considered for evaluation is the worst case filter set. The traffic load is at the rate of *2000* packets/sec. The results are validated over *20* runs of the experiment. It is to be noted that the result depends on the workload under consideration.

For this evaluation we designed an attack scenario on a single filter by increasing its hit count step by step following an exponential path. The rule-set is also designed to fit the worst case evaluations. The attack duration is *2* hours. The attack increased in intensity from *0-100-0* percent during the observed time period. The attack; in terms of hit count on the filter which attack is aimed at; followed a bell shaped curve. The best interval for

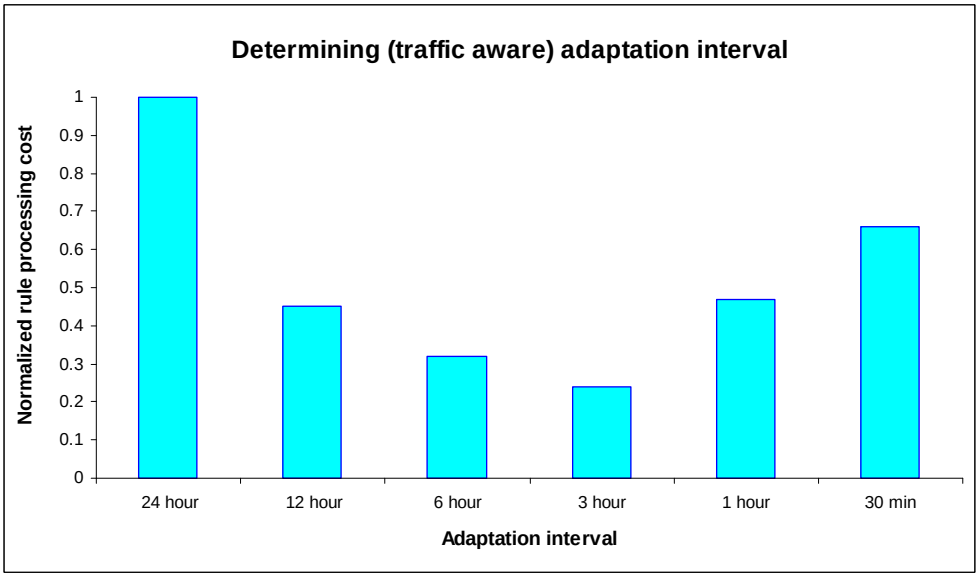

Figure 24: Determining Best Adaptation Interval

adaptation for this case is a period of every *30* minutes.

For attack traffic the benefit is very high, but the cost is also considerable. The main aim in an attack scenario is to make sure there is *"availability"* of the networked defense system. In this case the network administrator is ready to pay a higher average rule processing price by having frequent adaptations and finer adaptation intervals. This enables fine monitoring and rule switching in the rule-set to discard attack traffic at the earliest possible time.

The evaluation study demonstrates for a typical attack traffic the gain in rule processing time by performing the adaptations (including the cost of performing such adaptations) is more than *4* folds in comparison to the optimizer with no online adaptation. Along with this the firewall remains available and fully functional during the attack (in contrast to the firewall getting unavailable due to the attack).

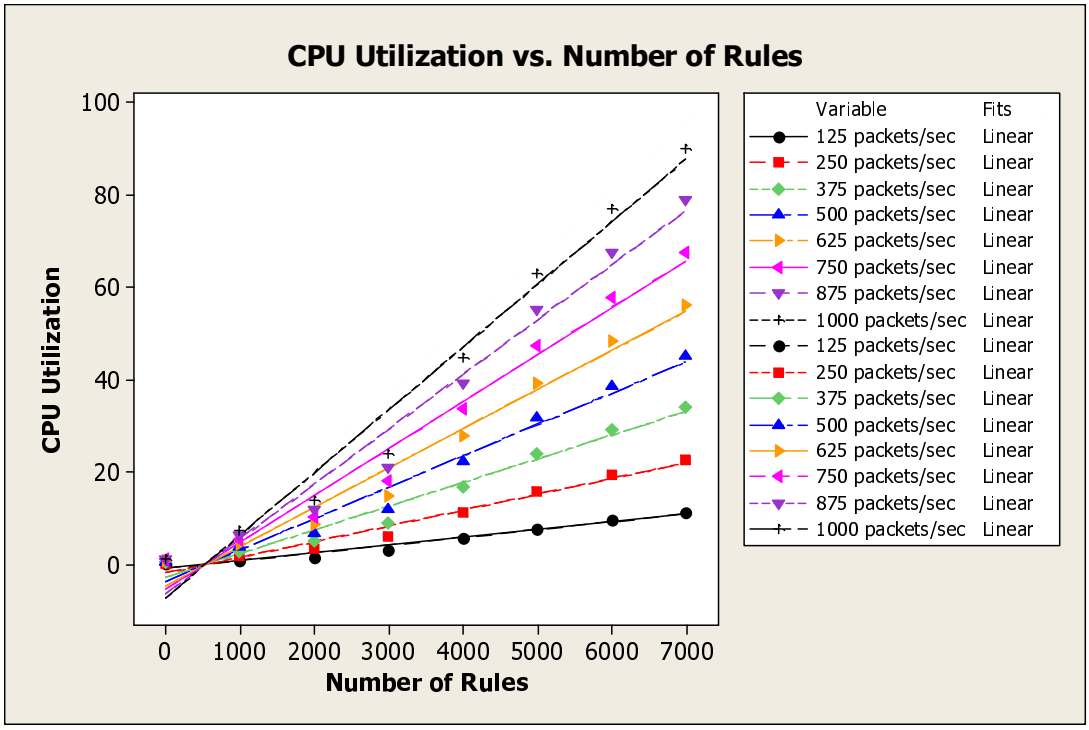

Figure 25: CPU Utilization vs. Number of rules

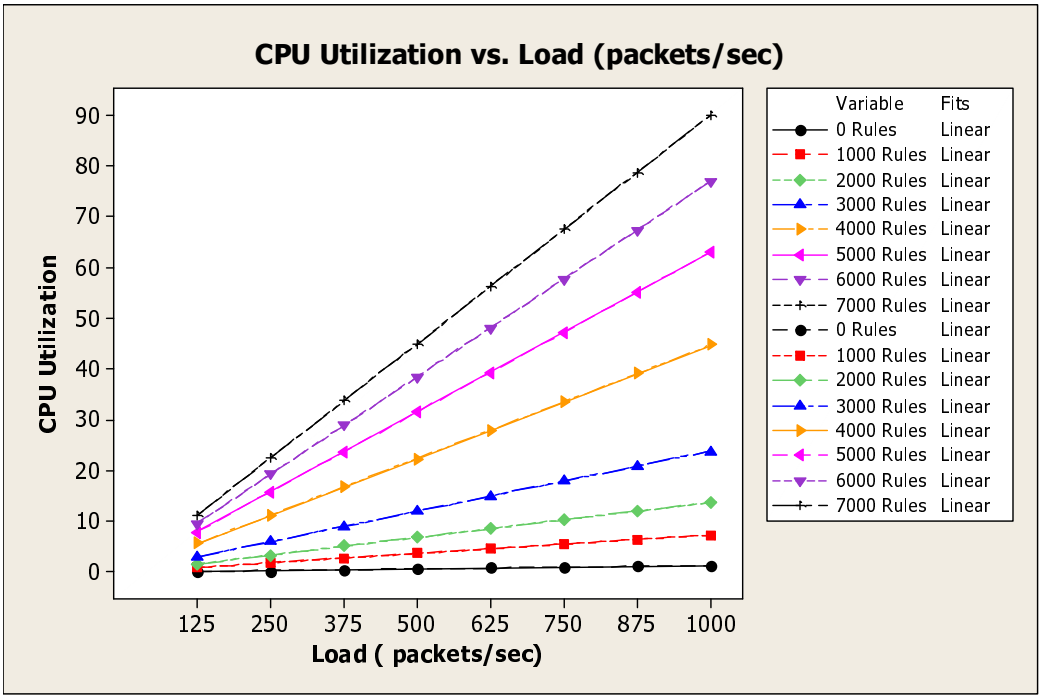

Figure 26: CPU Utilization vs. Load

4.4.3 Proportionality of rule processing cost

The final evaluation is to study the variation of CPU utilization with the change in the network traffic load or changes in the number of rules in the firewall rule-set. This study helps to establish the relationship between the above stated entities. The firewall rule-set is from a large *Tier-1 ISP* and the traffic is emulated from traffic traces obtained from the *Tier-1 ISP*. The firewall used for our experimental study is the open source *Linux IPCHAINS* firewall. Results in Figure 25 and 26 depict the proportionality relationship between CPU utilization verses number of rules or variation in traffic load. We conclude from this evaluation that the cost of rule processing is linearly proportional to the number of rules in the firewall rule-set.

4.5 SUMMARY

In this chapter we present the architectures and algorithms for Centralized Firewall Optimization. It is to be noted that this research is the first effort in incorporating the concept of firewall traffic log information to design and optimize firewall rules sets. Both rule-set based and traffic based optimizations are integrated in the proposed firewall optimizing tool. We also introduce a novel adaptive anomaly detection/countermeasure mechanism to deal with short term and long term anomalies. We have also performed a detailed validation of the size and cost metrices and the optimization results as discussed in [6] and in Section 4.3. Our proposed model and tool is flexible to be used by other *ISP* datasets and firewalls. We believe this research is the first step in the design of a complete optimizing toolkit for firewall optimization.

5.0 OPTWALL: A HIERARCHICAL FIREWALL OPTIMIZATION APPROACH

The overall efficiency, reliability, and availability of a firewall is crucial in enforcing and administering security, especially when the network is under attack. These challenges require new designs, architecture and algorithms to optimize firewalls. Contrary to a list-based structure, a de-centralized (hierarchical) design leads to efficient organization of rule-sets, thereby significantly increasing the performance of the firewall. The objective is to transform the original *list-based* rule-set into more efficient and manageable structures, in order to improve the performance of firewalls. The main features of the proposed approach are the hierarchical design, rule-set transformation approaches, online traffic adaptation mechanisms, and a strong reactive scheme to counter malicious attacks (e.g. *Denial-of-Service (DoS) attacks*).

5.1 INTRODUCTION

With the dynamic change in the network load, topology, and bandwidth demand, firewalls are becoming a bottleneck. All these factors create a demand for more efficient, highly available, and reliable firewalls. Optimizing firewalls, however, remains a challenge for network designers and administrators. A typical present day firewall enforces its security policies *via* a set of multi-dimensional packet filters (rules). Optimization of this multi-dimensional structure has been proven to be a NP hard [33,45] problem. This has motivated the research community to focus on various approaches to provide reliable and dependable firewall optimization methods. In-spite of a strong focus towards an efficient design, the techniques used

in current literature are static, and fail to adapt to the dynamic traffic changes of the network. The large number of security policies in enterprise networks pose the most important challenge to traffic-aware firewall optimization. Furthermore, with the increased ability of current networks to process and forward traffic at extremely high speed, firewalls are becoming highly resource constrained. Thus, the main objective of this chapter is to address the shortcomings of the current firewalls and increase their ability to deal with dynamic changes in network load and topology, particularly under attack conditions. In this chapter, the focus is on optimizing the most widely used *'list-based'* firewalls. To achieve this goal we propose a firewall transformation framework aimed at creating hierarchical firewall optimization rule subsets, to improve the operation and manageability of firewalls. The main challenge in the construction of these rule subsets is the need to maintain semantic integrity of the policy set at each level of the hierarchy. The overall goal it to improve the performance and manageability of such network systems.

The rest of the chapter is organized as follows: Section 5.2 introduces the Firewall Transformation Framework. We introduce the theory of the proposed transformation approach in Section 5.3. We present details of the splitting approaches in Section 5.4. In Section 5.5 presents the architecture and implementation methodology details of the proposed hierarchical firewall. We present the evaluation and results in Section 5.6. Finally, we conclude the chapter in Section 5.7.

5.2 FIREWALL TRANSFORMATION FRAMEWORK

In this section we introduce the proposed *Firewall Transformation Framework* aimed at improving the performance and manageability of firewalls. A software firewall defines its security policies *via* a set of security policies or rules. These security policies govern the filtering of network packets to and from the autonomous system. In this chapter our aim is to improve the *availability* and *good-put* of firewalls by proposing transformative algorithms to the *List-based* firewall representation into more manageable and performance efficient

representations. Any proposed transformation should preserve the semantic integrity of the original firewall rule-set, in order that the *Tier-1 ISP* network administrator accepts and registers to replace the original firewall rule-set with the transformation. Additionally, the transformed firewall rule-set should reduce the operational cost of packet filtering, in turn improving the efficiency and manageability of the firewall. The proposed *Firewall Transformation Framework* is defined formally as follows:

Let F represent the original "list-based" firewall rule-set. Let $T(F)$ represent the transformed firewall that preserves the properties and rules of the original firewall rule-set F. We define the cost function, $C(f)$, that represent the average operational cost of operation of firewall f.

$T(F)$ is a acceptable transformation of F iff:

- $T(F)$ preserves the properties and rules of F (Semantic Integrity property)
- $C(\mathrm{T(F)}) \leq C(\mathrm{F})$ (Cost property)

We discuss details of these properties and prove the property for our proposed *Firewall Transformation Approach* in Section 5.3.

5.3 FIREWALL TRANSFORMATION APPROACH

In this section we detail the theory behind the proposed firewall transformation approach. As stated in the previous section, the objective of this transformation is to obtain an "*acceptable*", $S(F)$, such that the transformation preserves the "Semantic Integrity" and "Reduced Cost" properties as discussed in the following.

This thesis proposes to achieve the firewall transformation by the process of "splitting" or dividing the original "list-based" firewall rule-set into groups of rule subsets. The input firewall rule-set F is first sorted based on traffic characteristics (*hit-count*) before the trans-

formation is initiated. The resulting rule subsets preserve the "semantic integrity" properties and rules of the original "*list-based* rule-set. To state formally:

Let $S(F)$ represent the proposed *Firewall Transformation*" approach of the original *list-based* firewall rule-set F.

Theorem 5.1. *$S(F)$ is a acceptable transformation of F iff:*

- *$S(F)$ preserves the properties and rules of F (Semantic Integrity property)*
- *Cost ($S(F)$) $\leq$ Cost (F) (Cost property)*

Proof. We prove the above theorem *via* "*proof of contradiction*".

- Semantic integrity property

  Let there be atleast one rule r in $S(F)$ such that the action of r on a network packet p is different than that of F on p. This implies that the action of the firewall rule-set F on a given network traffic is different from the action taken by the new transformed representation $S(F)$. Thus, the semantics of the original firewall rule-set is not equivalent to that of the transformed rule-set.

  The rules in the *list-based* rule-set are scanned in a sequential manner and divided into rule subsets based on traffic characteristics. Hence, the rules in the rule subsets in $S(F)$ originate from the rule-set F. The manner in which the rules are sub-divided into rule sub-sets, does not add any new rule or cause any rule deletions. This implies that no new rules are created or deleted due to the transformation process. This proves that the rule r in $S(F)$ must belong to some priority level in the original rule-set F, which implies we arrive at a contradiction.

  Hence, there are no rules in the transformed firewall rule-sets $S(F)$, that have an action different than one that is in the original rule-set F. This proves that the semantic properties and the rules of the original rule-set are preserved after transformation.

- Cost property

Let the operational cost of the original firewall rule-set and the transformed rule-set be $C(F)$ and $C(S(F))$, respectively. The focus of this research is on the most widely used *list-based* firewalls. For our evaluation we have assumed that rule matching is the most expensive operation. The operational cost of rule matching is proportional to the number of rules in the rule-set[1].

Let us assume for contradiction that $C(F) > C(S(F))$. For this assumption to be true, there exists atleast one packet p such that it matches a rule r in the firewall rule-set, where the cost of matching in the transformed firewall set is higher than the cost of matching in the original rule-set. Let the operational cost of processing the network packet p which matches rule r in the firewall rule-set F be represented as x and that matches rule r in transformed firewall $S(F)$ be y, where $y > x$. Due to the *list-based* firewall operation, the only way y is greater than x, is if the rank of r in $S(F)$ is lower than the rank of r in F. Since, both the firewall rule-sets are sorted by traffic characteristics (*hit-count* of incoming traffic), there exists rule r' in the firewall rule-subset $S(F)$ that has higher hit-count than r and is lower rank in the original rule-set F. Since, all the rules in F are sorted according to hit-count information and there are no new rules created or deleted due to the transformation process, we arrive at a contradiction.

Hence, we prove that the operational cost of firewall rule-sets $S(F)$ is <= the cost of the original rule-set F. In the worst case, the length of $S(F)$ will be equal to the length of F, which implies $C(F) = S(F)$.

□

In the next section 5.4 we discuss the details of two *Firewall transformation* approaches, namely, the *Optimal approach* and the *Heuristic approach*.

[1]This result has been proved for *Linux IPCHAINS* as presented in Chapter 4

5.4 FIREWALL SPLITTING APPROACHES

5.4.1 Optimal Approach

The *Optimal splitting* approach is based on an $A*$ search strategy. Achieving an optimal partition is possible since the cost can be calculated cumulatively for any partition as it is fixed and does not vary with the tuple priority. The basic steps of the *Optimal Approach* are depicted in Algorithm 1.

```
1  g(i,n) = cost of list_a and list_b after adding tuple n to list i;
2  h(n) = current cost of optimally placing the remaining tuples;
3  cost(i,n) = g(i,n) + h(n);
4  filter_a, list_a filter and tuples for list A;
5  filter_b, list_b filter and tuples for list B;
6  stack = stack ordered with least cost on top;
   input  : tuples[] = list of tuples sorted by cost
7  counter = 0;
8  list_a = ∅;
9  list_b = ∅;
10 currentTuple = tuples[counter];
11 Compute filter_a, filter_b, list_a, list_b;
12 while counter < tuples.size() do
13     if cost(A, currentTuple) < cost(B, currentTuple) then
14         if filter_a ∩ filter_b.widen(currentTuple) ≠ < any, any, any, any > then
15             stack.add(< list_a, list_b ∪ currentTuple, filter_a, filter_b.widen(currentTuple),
16             counter >);
17         filter_a.widen(currentTuple), list_a.add(currentTuple);
18     if filter_a ∩ filter_b = < any, any, any, any > then
19         < list_a, list_b, filter_a, filter_b, counter > = stack.pop();
20     else
21         if filter_a.widen(currentTuple) ∩ filter_b ≠
22         < any, any, any, any > then
23             stack.add(< list_a ∪ currentTuple, list_b, filter_a.widen(currentTuple), filter_b,
24             counter >);
25         filter_b.widen(currentTuple), list_b.add(currentTuple);
26     if filter_a ∩ filter_b = < any, any, any, any > then
27         < list_a, list_b, filter_a, filter_b, counter > = stack.pop();
28 counter + +;
29 currentTuple = tuples[counter];
   output : < filter_a, list_a, filter_b, list_b >
```

Algorithm 1: Optimal Approach for Rule-set Splitting

The function $g(n)$ determines the cost of the configuration in the current state. The function $h(n)$, on the other hand, computes the optimal cost of the remaining unassigned

tuples if placed in either of the subsets. The function $h_{max}(n)$ calculates the maximum cost of the remaining tuples. This can be used as a guideline to terminate the computation of the filters if the cost benefit resulting from this new filters does not improve on the gains of the previous configuration.

Another mechanism, which is used to reduce the overhead incurred by the search of the feasible optimal solution, is to prune the search space. This is triggered when the difference between $h_{max}(n)$ and $h_{min}(n)$ is lower than a specified error percentage. This enables the search to converge to filters of a nearly optimal solution at a much faster rate.

Even though a feasible optimal solution can be obtained, the worst case time complexity is of the order of 2^N, where N is the number of tuples. As the number of tuples becomes large searching for such a solution leads to a firewall bottleneck. Another shortcoming of the optimal solution is that the memory requirement can also become prohibitive as the number of tuples becomes very large. To address these drawbacks a set of heuristics are proposed. These heuristics converge to a nearly optimal solution, while maintaining a time complexity linear in the number of tuples.

5.4.2 Heuristic Approach

The heuristic solutions proposed are *local greedy search* solutions aimed at determining a set of filters and splitting the list-based tuple set into two tuple subsets. Each tuple of the list based set is disjoint from the other. This aids the performance and effectiveness of the approach to split the tuples into smaller tuple subsets. As mentioned in [26] application of greedy scheme works best when the tuples are all disjoint from one another. In other words, making tuples disjoint from each other enables full flexibility for tuple splitting and reordering based on traffic characteristics.

Depending on the choice of the initial filters, five different variations of the *Greedy Heuristic*

are proposed. The first variation is to deterministically assign the highest priority tuples as the initial filters. This heuristic is referred to as **Hit count-Hit count Heuristic**. The idea behind choosing the highest ranked tuples as the initial filters is to assign the highest costing tuples into different tuple subsets in order to arrive at a cost balanced solution. The main steps of the algorithm is described in Algorithm 2.

```
   input  : tuples[] = list of tuples sorted by cost
 1 filter_a = tuples[0];
 2 filter_b = tuples[1];
 3 Compute filter_a, filter_b, list_a, list_b;  for i = 2 to tuples.length() do
 4     if filter_a.matches(tuples[i]) then
 5         add tuple[i] to list_a;
 6         else
 7             if filter_b.matches(tuples[i]) then
 8                 add tuple[i] to list_b;
 9                 else
10                     distance_a = filter_a.distance(tuples[i]);
11                     distance_b = filter_b.distance(tuples[i]);
12     if distance_a < distance_b then
13         filter_a.widen(tuples[i]);
14         add tuple[i] to list_a;
15         else
16             filter_b.widen(tuples[i]);
17             add tuple[i] to list_b;
   output : < filter_a, list_a, filter_b, list_b >
```

Algorithm 2: Hit count-Hit count Heuristic

The next variation of the *Greedy Heuristic* is to assign one initial filter as the highest costing tuple and the next initial filter as one amongst the rest of the tuples which is at a maximum distance from the highest cost tuple. The distance is calculated using the *DISTANCE()* function as stated previously. This variation of the *Greedy Heuristic* is referred to as the **Hit count-Max distance Heuristic**.

The third variant of the *Greedy Heuristic* uses a randomly selected initial filter assignment. This heuristic is referred to as **Random-Random Heuristic**. A randomized algorithm is used to determine initial filters from a set of possible filters. The selected set is then used to build the hierarchical structure.

The fourth variant of the *Greedy Heuristic* is to consider the distance between all possible pairs of filters. The pair which contains the filters with maximum distance from each other is selected. This strategy has potential to split the tuples into well balanced sets. This heuristic is referred to the **Max distance-Max distance Heuristic**. The complexity for all the above approaches is proportional to the number of tuples in the initial tuple set.

The fifth variant of the *Greedy Heuristic* is the **All Pair Heuristic**. This variant considers all possible pairs of tuples as initial filters. Using the method depicted in Algorithm 2 we determine a split for each possible pair and then pick the split with the least cost.

The results for *All Pair Heuristic* are not included as the heuristic never converged to a solution due to the excessive overhead required to obtain the most cost efficient configuration among all possible pairs of tuples. The time complexity of this heuristic is of the order of N^3, where N is the number of tuples. For large values of N, the computational cost of the heuristic becomes prohibitive.

5.4.3 Improvements to rule-set splitting approaches

As we discussed in the previous section, establishing near optimal rule splits directly affects the performance of the hierarchical filter in turn improving the operational cost of the firewall. The reason being the filter split governs the "*re-splits*", "*re-promotions*" and "*re-orders*" of the rules in the rule subsets, which helps to incorporate traffic dynamics in the filter operation for a given firewall. The better the split (based on the traffic characteristics) the more stable and balanced the hierarchical firewall.

In this section our goal is to exploit the traffic characteristics further in order to determine the patterns in data which will aid the definition of better splits to enable hierarchical firewall optimization. In our analysis we conclude that the choice of initial filters should be accurate and is a defining factor in the design of stable rule split subsets. As discussed in

Algorithm 2, consideration of a single criteria (*e.g.* traffic volume/hit-count) information is insufficient to determine accurate initial filters for the hierarchal rule split. In this chapter we propose to consider the patterns in the traffic in order to aid the choice of the initial filters. This will necessarily affect the computation of better or closer to optimal rule-set splits (*aiding better traffic filtering*). In the following subsection we present the *clustering rule split* approach to aid de-centralized firewall optimization.

5.4.3.1 Clustering rule split In the proposed approach we determine the patterns in the traffic data *via* an exhaustive search algorithm to output a group of clusters. The criteria for the cluster organization is specified by the firewall administrator. The criteria could be a single attribute (*e.g. protocol:TCP/UDP*) or a combination of attributes. Such a search results in sets of clusters based on the specified criteria. Since our problem deals with the design of two initial filters for the hierarchical split operation, we arrive at two cluster sets (groups) based on the self-similar criteria as defined earlier. These self-similar clusters are the precursors to the rule splitting approaches as discussed above.

We then pick the initial filters one from each cluster group. We then follow the rule splitting steps as discussed in Algorithm 2. The cluster groups act as a feedback to the rule splitting process in this step. Each rule during the splitting process tries be retained in its self-similar cluster group during the process of hierarchical filter set determination. Since the clustering of the rules are run offline; we have the liberty to run the exhaustive search algorithm. The *disjoint* nature of the rules in the rule-sets enables the clustering and rule splitting process in determining traffic balanced rule split sets.

5.4.3.2 Parallel *A approach** In the next improvement, we propose a change in the *Optimal splitting* approach presented above to aid faster splits and to enable the split of larger data sets. Previously we have proposed an $A*$ search strategy to search for the defining filters for each split set. The runtime for such an approach in worst case is 2^N, where N

is the number of initial rules in the linear rule-set. For rules greater than *1000* Algorithm 1 did not terminate and hence did not produce the split sets we required. To overcome this problem we incorporated parallelism in the $A*$ search to enhance the splitting performance on larger data sets. We extended the optimal search approach developed earlier to include parallelism. The challenge is to keep track of the various intermediate sub solutions. We used multiple hashing mechanisms to store the locally optimal solutions to be used later to arrive at the final optimal split subsets. The proposed parallel $A*$ approach perform much better and is able to arrive at rule splits for large data sets (*nearly 10,000 rules*).

5.4.3.3 Weighted distance function The distance function as presented in Algorithm 2 is calculated by assigning equal weights to all the dimensions of a packet filter. Our analysis concludes that this assumption is not accurate. We propose that the distribution of the tuples amongst the dimensions should determine the weights that the dimensions takes and the normalization should occur on them. We changed the present splitting approach of finding the defining filter of a tuple by including a weighted distance function to determine the weighted distance of a tuple from the defining filters. This approach leads to more balanced splits. Results show that the split is about the improve from a $30 : 70$ split to a almost $45 : 55$ split. This optimization aids to reduces the worst case packet matching time for the hierarchical firewall we designed.

5.5 DESIGN ARCHITECTURE AND METHODOLOGY

In this section we presents the architectures and algorithms for a de-centralized firewall optimization, *OPTWALL* [5]. As we have discussed earlier, contrary to a list-based structure, a hierarchical design leads to efficient organization of rule-sets, thereby increasing significantly the performance of the firewall. *OPTWALL* uses a hierarchical approach to partition the original rule-set into mutually exclusive subsets of rules to reduce the overhead of packet

filtering.

In *OPTWALL*, the processing of a packet at a firewall starts at the root of the hierarchical structure. The packet is subsequently forwarded to the remaining levels of the hierarchy for further processing. Packet processing completes if a match between the attributes of the packet, as defined by the firewall security policy, occurs. In this case, the *action*, defined by the corresponding firewall rule, is enforced. Alternatively, on a non-match, a default action is invoked. The default action can either be *accept*, in which case the packet is forwarded to destination, or *reject*, in which case the packet is dropped. In the following, a formal specification of the objective and basic operation of *OPTWALL* are discussed.

5.5.1 OPTWALL Design Goals

Given a large rule-set of size *'N'*, the objective of *OPTWALL* is to partition this set into *'K'* mutually exclusive subsets. Each subset is associated with a unique filter which represents a superset of the associated policy subset. The hierarchical approach of the *OPTWALL* architecture is driven by three main design goals:

- Reduce the cost of processing the firewall rule-set, defined as the average processing time a packet incurs before an action is enforced by the firewall,
- Preserve the semantics of the original rule-set, and
- Maintain the optimality of the rule-set as traffic patterns and rule-sets change.

It is to be noted that in its general form the *'K-partition'* problem is NP hard, as it can be reduced to the *'Clustering'* [11] or the *'K-median'* problem [12]. Figure 27 depicts the process of partitioning *'N'* rules into *'K'* subsets.

To address the complexity of the partitioning problem, *OPTWALL* uses an *iterative* approach to partition the original set of rules and produce a *multi-level hierarchy* of *mutually exclusive*, *cost-balanced* rule subsets. Initially, the rule-set is divided into two subsets and

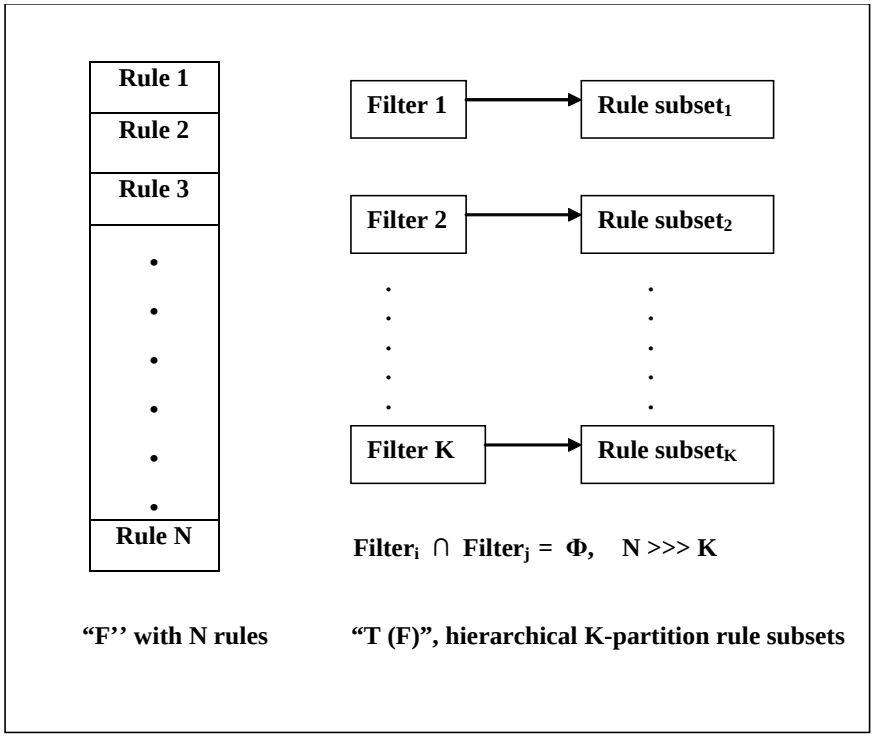

Figure 27: N rules into K partition problem

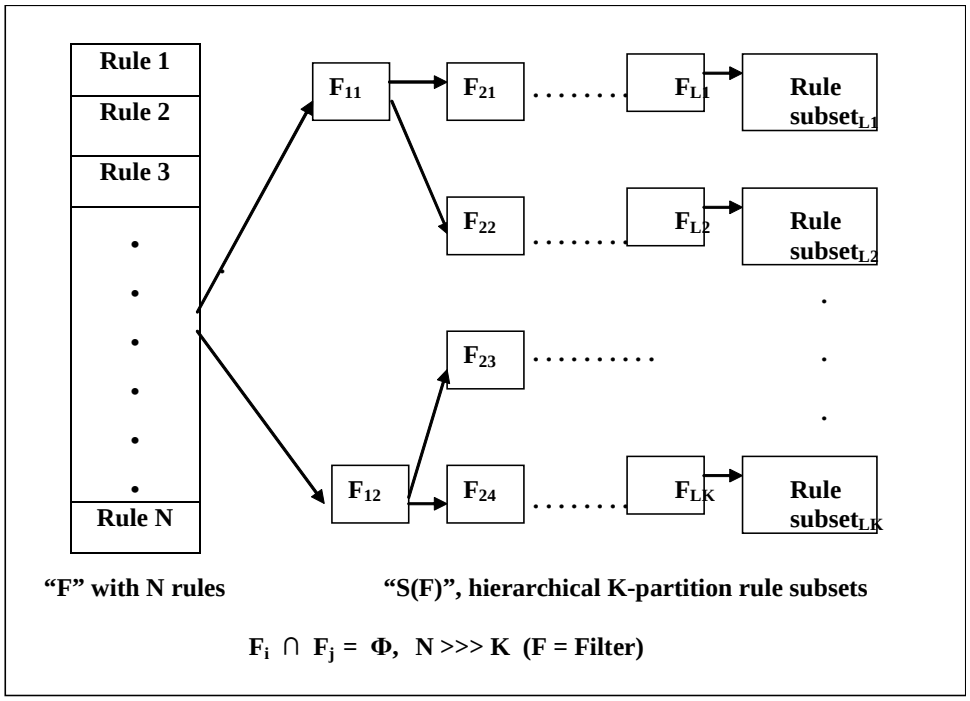

Figure 28: Basic operation of OPTWALL

filters, which covers the rules contained in each subset. The resulting subsets, along with their corresponding filters, form the first level of the hierarchy. This iterative process continues until further division of the subsets at the current level of the hierarchy is no longer cost effective. It is to be noted that this cost also includes the cost of determining the filters. The *OPTWALL* partitioning process is described in Figure 28.

5.5.2 Hierarchical Firewall Optimization Model

In this section we will present the processes used to achieve each of *OPTWALL* design goals. We first describe the multi-level data structure composed of rule subsets and their corresponding filters. We then discuss the procedure used to build the *OPTWALL* hierarchical structure and the actions required to maintain this structure.

5.5.2.1 Data Structure In order to process the rules, *OPTWALL* uses a hierarchical data structure in which the deepest level of the hierarchy contains the rule subsets and the intermediate levels contain filters which cover the rules included in those subsets.

The design of the data structure must ensure that the operational cost is reduced. The design must also ensure that the semantic integrity of the original rule-set is preserved. It is to be noted that the operational cost is determined by the deepest rule subset. Balancing the hierarchical structure in order to reduce the length of the deepest rule subset is, therefore, vital if the desire is to achieve the maximum reduction in processing cost. Furthermore, the data structure must be designed in such a way that the re-balancing process, in response to traffic changes, can be achieved with minimal overhead.

Semantic integrity of the original rule-set can be achieved, during the rule-set partitioning process, by computing filters that represent accurately and completely the rule subsets. Furthermore, packet processing must follow the same semantics specified by the filters result-

ing from the partitioning process. If the rules are split and reordered, in order to optimize operational cost, the process of re-enforcing the original rule semantics must be achieved with reduced overheard.

5.5.2.2 Hierarchical Structure Building The process of building the hierarchical structure described previously is accomplished using three basic stages:

- *pre-processing*,
- *ordering*, and
- *splitting*.

In the following, we discuss the basic operations carried out at each of these design stages.

The *pre-processing* stage takes the original list-based rule-set as its input and produces an optimized rule-set. This optimized rule-set consists of fully disjoint and concise rules, where all rule redundancies and dependencies are removed [7]. The fact that the rules in the rule set are mutually disjoint provides *OPTWALL* with full flexibility to reorder the rules and divide them into rule subsets, without violating the semantics of the original rule-set.

In the *reordering* stage, rules are reordered such that the highest cost rules are moved to the top of the rule-set. As stated previously, the cost of a rule is based upon the size of the rule and the amount of traffic processed by that rule, as indicated by its hit-count. By reordering rules the overall cost of processing traffic is reduced.

The goal of the *splitting* stage is to produce a partition of the original rule-set into a set of *mutually disjoint* rule subsets. This process involves taking the pre-processed rule-set and dividing it into rule subsets. Each rule subset is defined by a tuple which covers all rules in the subset. All such covering tuples are disjoint from one another. To partition the original rule-set, *OPTWALL* uses a multi-step process, whereby it initially splits the original rule-set into two subsets. It then recursively runs this splitting process on the subsets produced by

the previous stage to generate the next level of the hierarchical structure. This splitting process continues until the overall processing cost overshadows the benefit gained by further splitting the current subsets. When this occurs, the splitting process terminates and the previous level is selected as the feasible optimal depth of the hierarchical structure.

The efficiency of the partitioning process strongly depends on the way the rule subsets are produced at different levels of the hierarchy. Several strategies to produce feasible rule-set splitting can be used. These strategies are discussed in later sections of this chapter.

The produced hierarchical structure is then converted to a series of *IP-table* rule subsets. It is to be noted that most list based firewalls, such as *Linux IPCHAINS*, support the ability to forward packets from one list to another for further processing. Consequently, the *OPTWALL* hierarchical structure can be used to augment the filtering capabilities of list-based firewalls.

5.5.2.3 Hierarchical Structure Maintenance The hierarchical structure is built to reflect the current traffic pattern and rule-sets. As the traffic pattern and rule sets change, the hierarchical structure must be updated to maintain its balance. To detect changes, *OPTWALL* monitors the traffic logs in real-time and adjusts the hit-counts. *OPTWALL* asserts that changes have occurred if the difference between the old and updated hit-counts of any rule exceeds a predetermined threshold. This threshold, a tunable parameter, is determined based on the traffic characteristics and the policy set under consideration.

If the need to balance the hierarchical structure rises, *OPTWALL* uses the existing traffic logs to update the cost of rules in the rule subsets, including rules which have been added to reflect a new security policy. *OPTWALL* then uses *reordering*, *re-splitting*, and *promoting* to re-establish the balance of a hierarchical structure.

Re-ordering consists of re-prioritizing the rule subsets at the deepest level of the hierar-

chical structure. This process is necessary to take into consideration the impact of traffic variations on the hit-count of rules in a given rule-set. *Re-ordering* is triggered when the difference between the current and previous hit-counts of a given rule exceeds a preset threshold.

Re-splitting is invoked when a sub-hierarchical structure becomes out of balance, due to traffic variations. A sub-hierarchical structure is considered to be out-of-balance if the average packet processing cost exceeds a predefined threshold. This process can occur at any level, including the root of the hierarchical structure. When sub-hierarchical structure is out of balance, splitting is applied to the original rule subset that generated this sub-hierarchical structure. In some cases, it is not possible to produce a more balanced hierarchical structure, in which case the level is marked as currently optimal and the threshold for the intermediate levels are increased.

Promoting aims at reducing the overhead of packet processing at different levels of the hierarchy. The need for rule promotion occurs when a single rule hit-count increases dramatically and exceeds its predefined threshold. This scenario is likely to occur during anomalous traffic behavior, typically observed during *Denial-of-Service (DoS)* attacks. To mitigate the impact of *DoS* attacks and drastically reduce the cost of processing traffic generated by these attacks, the rule is promoted to a level above the filters. Depending on the rule's priority, promotion may continue recursively until it reaches its appropriate priority level. In the extreme case, the rule may be moved all the way up to the root of the hierarchical structure. This promotion is temporary and the rule is not removed from the rule subsets. The reason behind the temporary promotion stems from the transitory nature of *DoS* attacks. Once the traffic has returned to its normal levels, the promoted rule can be removed from the higher levels.

The automatic interaction between the levels (parent-child modules) of *OPTWALL* is illustrated in Figure 29. Each level, starting from the root, acts as a central authority to a lower level in the hierarchy.

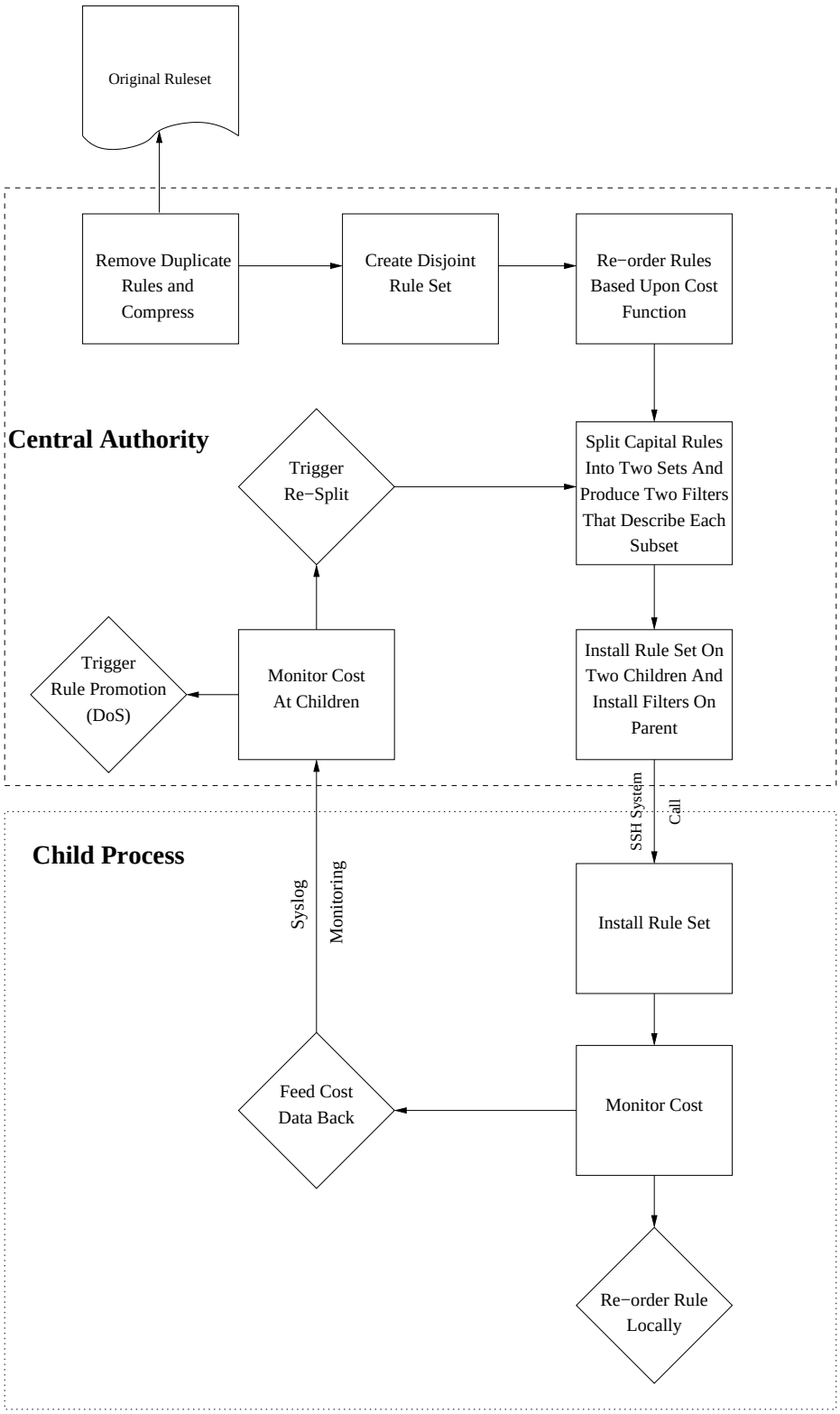

Figure 29: OPTWALL: Architecture

The efficiency of the splitting process, in terms of *packet processing overhead*, strongly impacts the performance of the firewall. We first describe the splitting process and discuss various solutions proposed for splitting the rule-set. We define a tuple, as a rule with single attribute value. We will use the tuple set as the input to our splitting process.

The output of the splitting operation are two filters and their corresponding tuple subsets. The filters and tuple subsets are semantically similar to that of a single list-based tuple set. The process of splitting relies upon three basic functions:

- *MATCH()*,
- *DISTANCE()*, and
- *WIDEN()*.

All three function are available on the filter object and all accept a single argument of a tuple.

The *MATCH()* function checks to see if a tuple is covered by the filter. The source and destination *IP* addresses are compared to the range specified in the filter. Similarly the port number is compared to the port range specified in the filter. The protocol type is matched to a list of protocol types the filter evaluates upon. This function returns true if the tuple matches the tuple and false otherwise.

The *DISTANCE()* function calculates the distance between a given tuple and the filter. If the filter matches the tuple then the value returned by this function is *0*. Otherwise, this function returns a positive number between *0* and *1*. The distance is based on the entire tuple.

The numerical value between two *IP* addresses represents the distance between them. If the *IP* addresses represent ranges, the distance function based on the distance between the two farthest points within the ranges is calculated. A similar procedure is used to calculate the distance between ports or port ranges. The protocol distance is set to *0* if the protocol already exists in the protocol list for the filter. Otherwise the distance is set to *1*. All the

distances are then normalized to the maximum values of their respective fields. The summation of this normalized values are then weighted and re-normalized to produce a value between *0* or *1*.

The *WIDEN()* function is used to expand a filter such that it matches the given tuple. This is achieved by expanding the *IP* range, port range, and protocols. A function calculates the cost of the tuple based on traffic characteristics and other tuple properties.

The driver of the splitting process is the search for a set of filters, which covers the hierarchical structure without violating the semantic integrity of the original rule-set, aiming at improving the operational cost of the firewall. Ideally, optimal splitting ensures that, at the end of the partitioning process, all subsets has equal cost. Consequently, when an optimal split is achieved, the average processing cost of each packet is reduced by half of its original cost. An optimal strategy for performing a cost-balanced split of the original set of rules is to use two sub-lists and alternatively place the rules in each list, starting with the highest cost rule, until the set of rules is exhausted. While this strategy is optimal, it is not always feasible. This due to the fact that each rule subset produced at each stage of the splitting process must have a mutually disjoint set of filters. Computing such filters may not be always achievable.

In the next section we present the detailed evaluation study of the proposed *OPTWALL* implementation.

5.6 EVALUATION

In this section we describe the experiments and evaluations to validate the proposed *Hierarchical Firewall Optimization* approach. We perform our validations by improving on the widely used open source firewall, *Linux IPCHAINS*. The data used for our experiments is emulated data from a large *Tier-1 ISP*. Our choice of firewall is representative of *list-based*

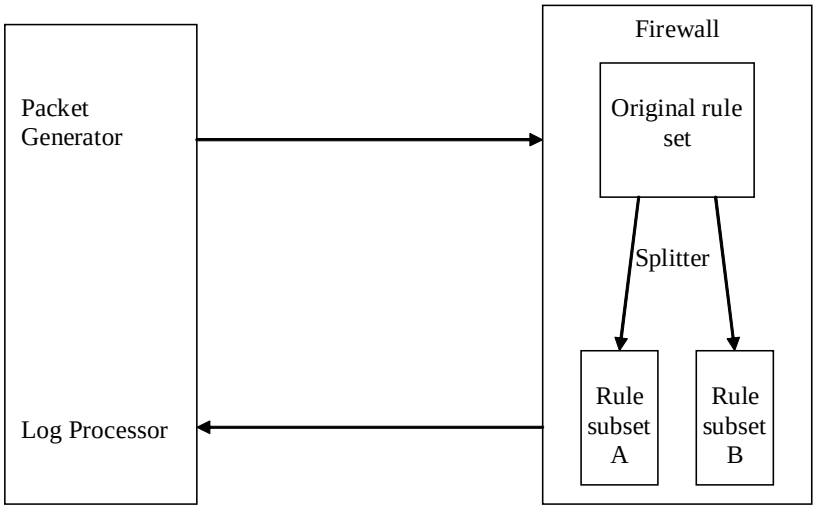

Figure 30: Experimental Setup

firewall, which is the focus of this research.

The experimental set up for the evaluation of the proposed approach consists of a machine acting as a firewall and another generating traffic and collecting logs. The machines used for our evaluation are $AMDAthlon^{tm}$ *64* bit Processors *3000+* running *Ubuntu Linux* operating system. The machines are isolated for testing to ensure that there are no other variants. Figure 30 shows the block diagram of the experimental setup.

There are two types of traffic characterizations used to evaluate *OPTWALL*, namely, the *worst case* and the *emulated case* behavior. In the worst case scenario, traffic is composed of a single packet type that does not match any of the tuples. This assures that the packet will be filtered only by the default action tuple. The emulated traffic is generated by creating packets that match each tuple and proportionally instantiating them to a traffic trace similar to that of a large *Tier-1 ISP's* firewall operation. The worst case traces are used to study the worst case performance of *OPTWALL* in comparison to the baseline case, a

list-based firewall. Performance at worst case is determined by using constant traffic rates and measuring the overall CPU utilization. Traffic rates are determined by loading the firewall from 25% to 100% utilization with the installed list-based rule-set. A similar approach is used to determine the load for the emulated traffic evaluations.

5.6.1 Evaluation results

The following subsection discusses the various results highlighting the potential of the proposed *OPTWALL* approach.

5.6.1.1 Hierarchical model evaluation This study is performed to evaluate the potential of the hierarchical design and its effect on efficient firewall optimization *w.r.t.* a list-based design. The extent of the hierarchy depends on the *tuple set size*, the *traffic characteristics* and the *variability in traffic*. For our evaluation we fixed the tuple size, load applied and the splitting approach used to determine the benefit from the proposed hierarchical design. The experiments are conducted on a heavily loaded system and using the best performing heuristic amongst all the solutions proposed earlier in the chapter. We use a tuple set of nearly *5,000* tuples, load of *1,440* packets/sec and the *Max Distance-Max Distance* Heuristic for our evaluations. Results as in Figure 31 shows the potential of the proposed OPTWALL framework. It is to be noted that after a point, re-splits cause more harm than good. The results depict a way to arrive at a sweet spot between the number of re-splits and the gain to due the hierarchical design.

5.6.1.2 Worst case performance evaluation The next study performed is to determine the worst case packet processing cost of the firewall. A worst case packet processing occurs when every packet entering the system requires processing of the entire tuple subset. This means that it will match the last tuple, which is default deny. We used various tuple sizes for our evaluations. The results are for a typical large tuple set, consisting of *60,000*

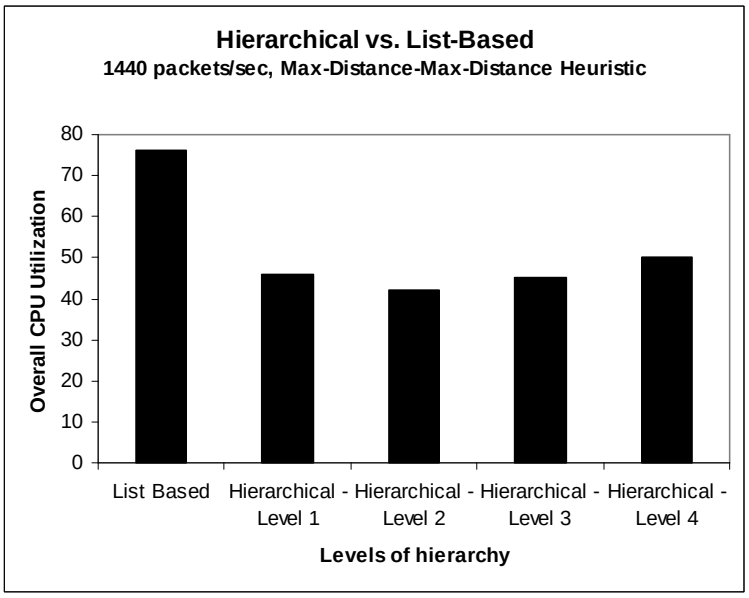

Figure 31: Hierarchical vs. List-Based

tuples. Due to the memory limitation of using the *Optimal Approach*, we use a pruned approach, $\sim A*$ *approach* for our evaluation. Results in Figure 32 demonstrate that the $\sim A*$ *Approach* and *Max distance-Max distance Heuristic* perform best in comparison to the baseline *list-based* approach. It is to be noted that filters output by the $\sim A*$ *Approach* perform better traffic filtering than the heuristics approaches.

5.6.1.3 Emulated traffic performance evaluation The next study is to determine the *CPU* consumption of the firewall when the traffic applied follows the normal traffic trace. Results as in Figure 33 show the benefit of the proposed scheme. The *CPU* improvement in the worst case is about 35% and in the emulated case is about 14%. Since, the CPU consumption is additive, any gain on the emulated case can be translated as a capacity for dealing with more anomalous traffic that can be handled by the firewall. In other words, *OPTWALL* can deal with a larger predicted traffic volume and also a much larger anomalous traffic.

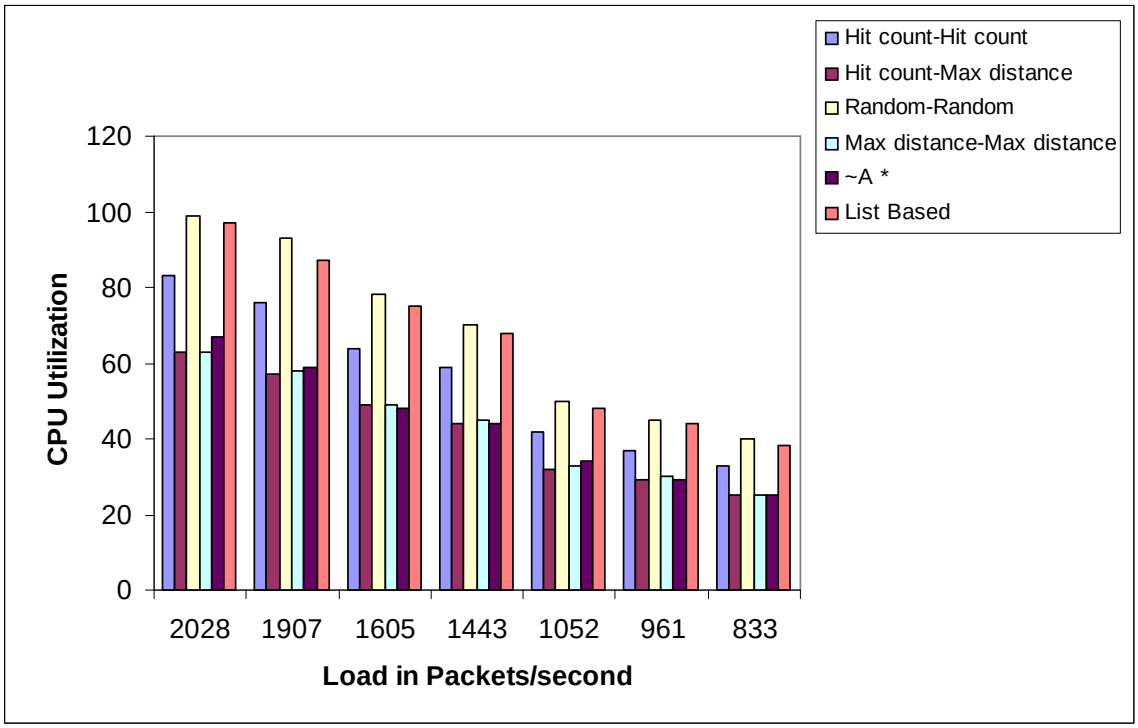

Figure 32: Performance Evaluation (Worst-Case - 60,000 tuples)

5.6.1.4 Handling attacks evaluation The aim of this study is to test the strength of *OPTWALL* in handling attacks and traffic fluctuations. Since the hit-counts for default action tuples are large and unpredictable, it can cause a huge bottleneck to the entire firewall operation. Figure 16 illustrates an instance of a large hit-count for a default action tuple. To test the performance of *OPTWALL* in handling such attacks we emulated the attack and increasing the hit-count of a certain default action tuple from $0 \sim 100,000$. Figure 34 shows the competence of OPTWALL in countering dynamic traffic changes and hence aiding the steady maintenance of firewall operation.

5.6.1.5 Sensitivity analysis evaluation The final study is aimed at *sensitivity analysis* of the proposed approaches. The analysis is performed for tuple sizes varying from *0 - 1000* tuples. Figure 35 details a comparative study between the *baseline list-based*, the *best performing heuristic solution* and the $\sim A*$ *approach*. The evaluation is conducted for a heavily loaded firewall operation. From the results we conclude that the proposed *heuristic solutions* are best suited for hierarchical firewall optimization.

5.6.1.6 Improved rule splitting In this study we evaluate the benefit of the improved traffic-aware splitting approach presented in Section 5.4.3. The evaluation is for the worst case and the emulated case operation of the firewall. The traffic load is *2000* packets/second and *7000* packets/second for worst case and emulated case, respectively. We invoke the best performing *Max-distance-Max-distance* heuristic to determine encompassing filters from the resulting cluster groups. The result is averaged over *20* runs of the experiment. The *Y-axis* represents the *CPU-Utilization* and the various approaches are represented in the *X-axis*. Results in Figure 36 and Figure 37 demonstrate the benefit of the proposed approach in improving the operational cost of firewall.

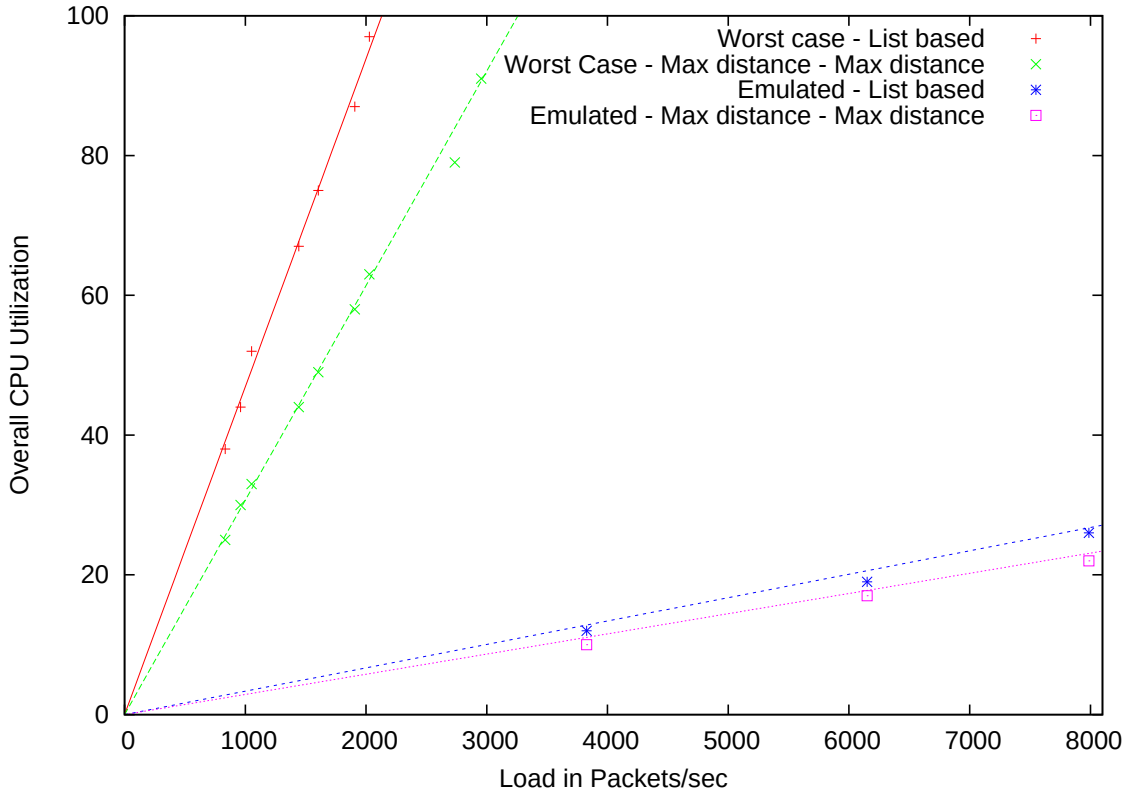

Figure 33: Emulated Traffic Performance Evaluation

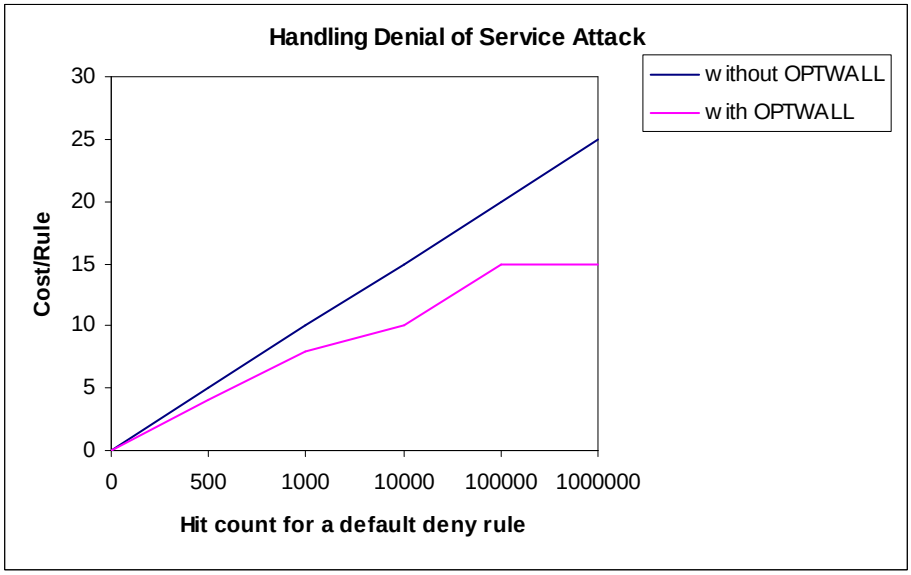

Figure 34: Countering DoS Attacks

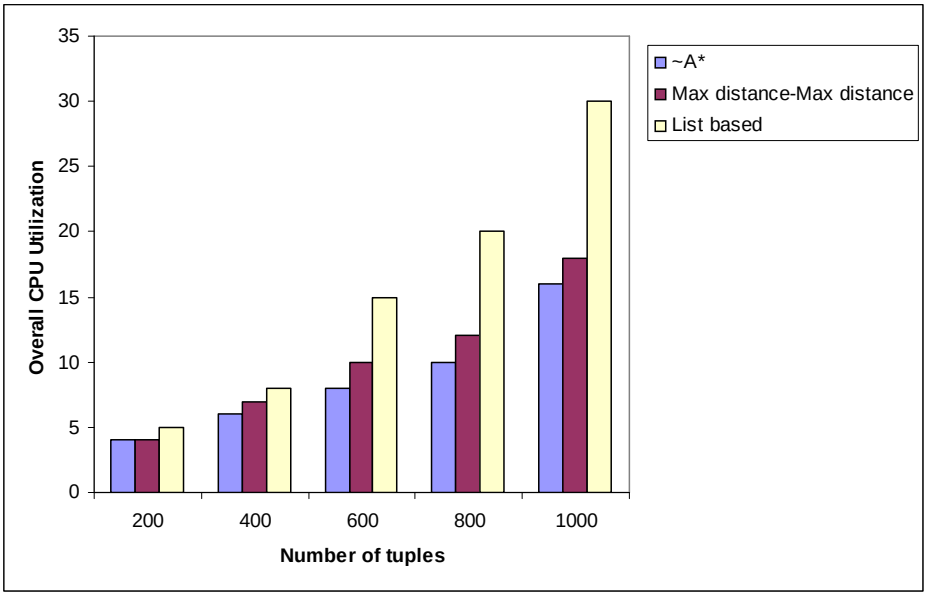

Figure 35: Sensitivity Analysis

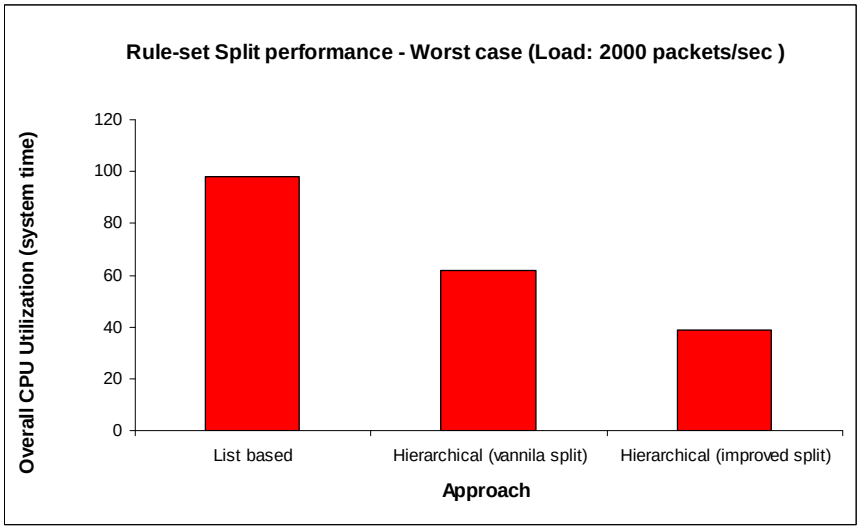

Figure 36: Weighted split performance - Worst case

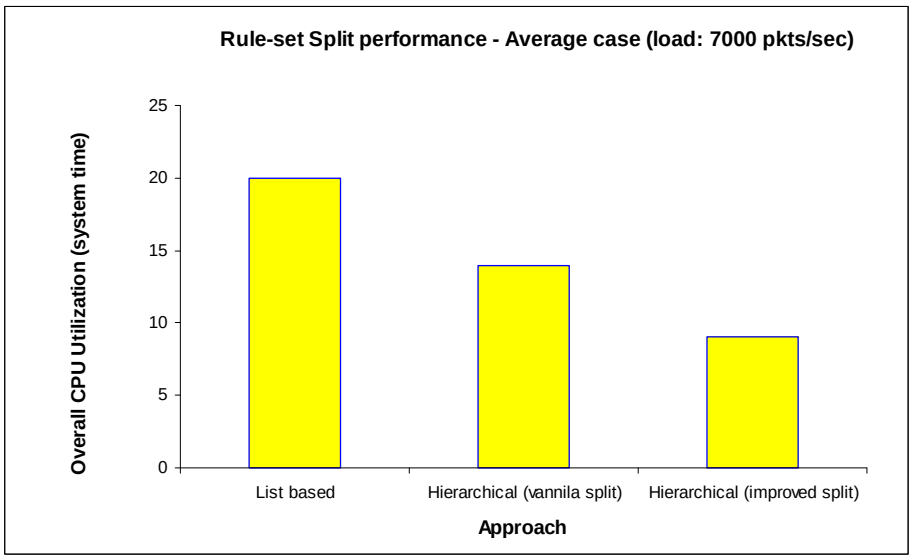

Figure 37: Weighted split performance - Emulated case

5.7 SUMMARY

This chapter introduces a novel *firewall transformation framework* and *transformation approaches* aimed at improving the performance and manageability of de-centralized firewall operation. We introduce a *hierarchical splitting* approach *via* the proposed *OPTWALL* toolset. *OPTWALL* helps to achieve the maximum benefit *via* various splitting processes to arrive at feasible optimal and near optimal solutions. We study the performance of *OPTWALL* both for worst case and normal firewall operation. We also introduce a novel adaptive anomaly detection/countermeasure mechanism to deal with short term and long term anomalies. Our proposed model and tool is flexible to be used in different firewall environments and data sets. We believe this research presents the design of a complete optimizing toolkit for firewall optimization. Results demonstrate that the proposed approach aids in improving the performance of de-centralized firewall operation.

6.0 CONCLUSION

The explosive growth of the Internet in the last few years has given rise to a vast number of services being deployed on a global scale. The increase in the Internet size, volume of data traffic, and the daunting task of processing this huge Internet traffic, has created new challenges in building high performance network devices. Thus, Internet has undoubtedly become the largest public data network, enabling and facilitating both personal and business communication over the whole world.

Data communication networks have become an infrastructure resource for businesses, corporations, government agencies, and academic institutions. The volume of traffic over the Internet is expanding at an exponential rate everyday. More and more communication is taking place over the Internet via Emails, the World Wide Web, Remote access, Collaboration, File sharing, Streaming media, Voice Telephony (VoIP), etc. In very simple terms, Internet is the *life-line* of today's world. As such social dependence on networked information systems continues to grow exponentially, unfortunately, a similar growth in threats against the security of the networked infrastructure and services is taking place.

Attacks span the spectrum from computer worms and individual and localized intrusions aimed at gaining access to information and system resources, to coordinated and distributed attacks aimed at disrupting services and disabling critical infrastructure. As these attacks proliferate and grow in scope and sophistication, different institutions find themselves under growing pressure to place signification restrictions on open Internet accessing in the form of firewalls, selective application deployment and mandatory proxies. The goal of our research is to solve the challenging problem of providing confidentiality and integrity along with fast

response and availability in the present day network systems.

This thesis presents dynamic architectures and algorithms to secure networks. The proposed architectures are geared towards today's a global data-driven environment. The main characteristics of the techniques and approaches presented is the ability to involve traffic characteristics to aid and optimize the design of such secure network systems.

The main contribution of this thesis is the development of theory and tools for *Firewall Optimization* for large scale enterprise networks. This is achieved by involving the key mechanism of traffic information from the Internet as a feedback to design better and smarter packet filtering mechanisms. The main challenge which we faced in such a design is the number of security rules and policies that the firewall has to enforce. As the network speed increases there is an increased burden on the firewall for fast packet filtering. Along with all this demands there is also an invariable need to maintain high policy integration. All these make the problem of *Firewall Optimization* a very challenging one.

We present architectures and algorithms for firewall optimization and show the tremendous improvement in operational cost of firewalls. We also present transformation framework and approaches geared towards optimizing de-centralized firewalls. We present detailed analysis and our results are validated over a real world open source firewall, *Linux IPCHAINS*. Our proposed *PITTWALL* and *OPTWALL* framework is highly portable to other data set and firewall environments. We believe, this research is one of the first efforts in using firewall traffic log information to design and optimize firewall rules sets. Both rule set based and traffic based optimizations are integrated in our firewall accelerating tool. We have also introduced a novel adaptive anomaly detection and countermeasure mechanism to deal with short term and long term anomalies. We have performed detailed validation on the rule size and cost metrics to demonstrate the strength of the proposed research.

Our research conclusions are based on investigations over a large Tier-1 Internet Service Provider's (ISP's) firewall traffic information. Investigation resulted in a tremendous scope

for adaptive, traffic-aware Firewall Optimization. Results demonstrated nearly 10 fold improvement in the operational cost of firewalls for the Tier -1 ISP. These outstanding results propelled us to study in depth the issues related to the design of a sound and robust infrastructure for Firewall Optimization.

At the core of the proposed firewall optimization research are two driving goals; better ways to search, update and filter a large set of data based on specific policies and rules to optimize the operational cost of firewalls; and better strategies to defend against attacks which go beyond the classic peripheral model. The proposed architectures and algorithms are evaluated using traffic information from the large Tier-1 ISP. Evaluation results demonstrated the powerful potential of the traffic-aware optimization strategy. To enlist, results on the de-centralized firewalls achieved nearly 35% improvement in the operational cost of firewalls. As future work we intend to use other ISP datasets and firewalls to study, optimize and validate our approaches. We would also be extending our optimization ideas on other types of firewalls (not only list based ones). We believe this research is the first step in the design of a complete accelerating toolkit for firewall optimization.

Fortunately, the firewall optimization techniques developed in this thesis are incorporated in the secure network system of a large *Tier-1 ISP*. We hope that our proposed solutions would be included in numerous ISPs and would become a core part in the design of the next generation Internets. Our long term goal is to include the proposed research in the day-to-day operation of the Internet. In summary, we strongly believe that this area of research is very vital and of profound importance to every aspect of human-technology interaction. Our long term goal is to deploy and test our theory and tools over various networks to fully demonstrate their potential. As future directions we propose to enhance and complement the area of cyber-security with adaptive and on-time filtering and by protecting the privacy of information exchange amongst applications. Another important research direction of great potential is to develop a formal trust management model applicable to such a global distributed environment. To conclude, we present the design of *traffic-aware de-centralized* defense infrastructure for next generation network systems.

7.0 FUTURE RESEARCH DIRECTION

With the tremendous growth in the dependence of services on the Internet, service disruption has become less and less tolerable. The greatest threat to service availability is the rapid growth in the complexity and frequency of large-scale distributed attacks. These attacks cause economic losses due to unavailability of services and potentially serious security problems by the incapacitation of critical infrastructures. Despite the tremendous attention by the research community to find distributed attack countermeasures, a practical and comprehensive solution is yet to see the light of the day. In this chapter we present a future research direction aimed at finding a solution to the above problem that builds on the contributions of this thesis.

It is well understood that it is difficult to eliminate all distributed attacks, as it would require securing all machines on the Internet against misuse, which is not a feasible solution. A possible practical approach is to design defense mechanism that will detect the attack and respond to it by dropping the excessive malicious traffic. Generally, it is very easy to detect the distributed attack near the *destination*, however the approach is too late in attack detection. In the ideal case, the attack should be stopped as close to the source as possible, but with the distributed nature of the attack it is not possible to decipher such attack with the little information available near the source of the attack. Additionally, the attack source is distributed. Thus, a realistic solution should move away from single-point and local solutions towards a global and collaborative approach of attack detection and subsequent attack mitigation. In this chapter we propose a future research direction to dynamic, de-centralized firewall optimization approaches proposed in the thesis.

7.1 INTRODUCTION

Distributed attacks, *i.e. Distributed denial-of-service (DDoS)* attacks are a serious threat for the Internet's stability and reliability. A *DDoS* attack is an explicit attempt to interrupt an online service by generating a high volume of malicious traffic. These attacks consume all available network resources, thus rendering legitimate users to face service disruptions. The impact of the attack can vary from minor inconvenience to the users of a web-site, to serious financial loss to companies that reply on their on-line availability to do business [36,40].

Recent massive Internet worm outbreaks such as Slammer [38], Blaster [29] or Sasser [30] have shown that a large number of hosts [34] are patched lazily or are operated by security-unaware users. Such hosts can be compromised within a short time to run arbitrary and potentially malicious attack code transported in a worm or virus or injected through installed back-doors. *Distributed denial-of-service* attacks *(DDoS)* use such poorly secured hosts as attack platform and cause degradation and interruption of Internet services, which result in major financial loss, especially if commercial servers are affected [15].

Keeping a commercial server available round the clock is a tough task; while attackers are able to exploit the processing and bandwidth resources and the flexibility of a huge number of compromised hosts to construct new attack tools and variants; operators of Internet servers are left without appropriate means to counteract attacks. Widespread availability of attack tools makes it trivial for naive users to carry out large-scale attacks. As a consequence, new attacks appear frequently, while defense strategies lag far behind. We believe that current security technologies and concepts that focus on end system and access networks soon cannot cope anymore with the growing number and the increasing intensity of Internet attacks. We are convinced that large-scale attacks can only be efficiently handled by providing increased security within the network.

This motivates the need for robust, scalable and effective architectures for detecting and mitigating such attacks. Any effective and realistic solution to see the light of the day

should not add significant complexity in the core routers and should exhibit characteristics that allow it to scale to large network sizes and to handle large-scale distributed attacks. The attack defense infrastructure should possess mechanisms to aid collaborative operation and be able to easily adapt to the traffic dynamics of the network. Most importantly such a solution should enable ease of security policy enforcement while maintaining and preserving the semantic integrity of the policy set. Finally, the proposed solution should not require considerable changes in the existing infrastructure in order to ease deployment in such large scale networks.

In this chapter to present a research direction that focuses on a clean-slate approach and argues for a separate security plane to detect and mitigate distributed attacks. The decision logic for the security mechanisms is distributed in a small set of active security guards (*Active sentinels*), each responsible for a set of core routers. The *Active sentinels* collaborate both by proactive and reactive mechanisms to mitigate attacks. In the following section we present a brief description of the collaborative defense model aimed at limiting distributed attacks.

7.2 COLLABORATIVE DEFENSE MODEL

Defending against large-scale, *DDoS* attacks is challenging, with large changes to the network core or end-hosts often suggested. Any practical solution must provide incentives for deployment. Furthermore, as link speeds increases, there is mounting pressure on routers to perform forwarding at link speed. With the increased complexity of the control, management and data planes, it is important that a solution does not add further complexity to the core routers. The salient features of the proposed collaborative defense infrastructure are the following:

- No changes required to the core routers or in the end-hosts.
  This implies that the proposed architecture can be easily deployed.
- No change in the computational overhead on core routers.

- The proposed architecture adds no complexity to management, control or data plane.
- Better detection capability against a variety of attacks.

  The strength of the proposed architecture is its ability to handle distributed *DoS* attacks by the coordination of *Active sentinels* which cannot be achieved by simple edge to edge filtering.
- Attack detection is possible both from *in-network* and *out-of-network* attacks.

The task of the *Active sentinels* is to actively probe packets sent probabilistically by routers in the network. Upon the detection of an attack the *Active sentinels* send messages to the *Edge firewall nodes* to filter the attack traffic. The *Active sentinels* can also direct the core routers to increase p, the sampling rate. The *Collaborative Defense Architecture(CDA)* is presented in Figure 38. The collaborative action is achieved *via* two mechanisms, namely, *Intrusion Detection and Response* and *Packet Filtering and Traffic Monitoring.*

7.2.1 Mechanisms for Collaborative Defense

7.2.1.1 Intrusion Detection and Response consists of an infrastructure comprising of a network of peers, that dynamically and collaboratively defend against intrusions and denial of service attacks. The thrust of this task is the design of resource efficient algorithm for probabilistic sampling and inspection of packets to detect attacks, and the development of algorithms and methods for a collaborative Sentinel deployment which guarantees "optimal" network coverage and a scalable response to attacks.

7.2.1.2 Packet Filtering and Traffic Monitoring are critical components of the *Intrusion Detection System* for the proposed collaborative defense architecture. Upon detection of an attack, a security policy, consisting of a set of rules and their corresponding actions, must be in place to drop attack packets before they infiltrate the systems which they target. With the dramatic advances in link speed, packet filtering must be constantly optimized to cope with traffic demands and attacks. This problem is even more critical when application level filtering is used. In this task, we aim to investigate and develop efficient, yet easy to implement, collaborative packet filtering techniques. The focus will be on dynamic,

collaborative solutions, which use the network traffic statistics to reduce the overhead. We propose to import architectures and algorithms designed in the area of firewall optimization as discussed in Chapter 4 and 5 to achieve the above goal.

7.2.2 Type of Nodes in *CDA*

We propose three basic node types in the *CDA* infrastructure as discussed in the following:

- "*Active sentinels/Dominating nodes/Smart nodes*"
  Active sentinels are smart nodes which are responsible for monitoring the core nodes in their coverage. These nodes are responsible for taking decisions about traffic anomalies and inform the *Firewall-enabled routers* for changes in the rule-set to mitigate the upcoming anomaly. *Active Sentinels* also receive information from other peers in the *Autonomous System(AS)* informing them of local anomaly information at distant parts of the network. On receiving this information they take a decision to inform their *Firewall-enabled routers* to take appropriate action. Alternatively, on detection of an anomaly in their covered network of core nodes, the *Active Sentinels* inform their peers of the same. They also pass information about flows which do not raise individual alarms probabilistically amongst each other. Depending on a specified criteria the *Active sentinels* then aggregate the values periodically (during online operation) to check if the defined threshold is exceeded. If the aggregate behavior of a flow is unexpected, the *Active sentinels* inform the filtering agents to take necessary action to prevent distributed attacks. This is a collaborative process and helps to mitigate distributed attacks.

- "*Passive sentinels/Core routers/Covered nodes*"
  The *Passive sentinels* are the core router nodes which are covered by the *Active sentinels*. The "r" criteria or *redundancy* criteria determines the number of *Active Sentinels* that are responsible for these nodes. Their function is to perform normal router operation and also forward the packet to the enquiring *Active sentinel* at the desired rate set by "p" (assigned by the *Active sentinel*). It is to be noted that the core router node is not devi-

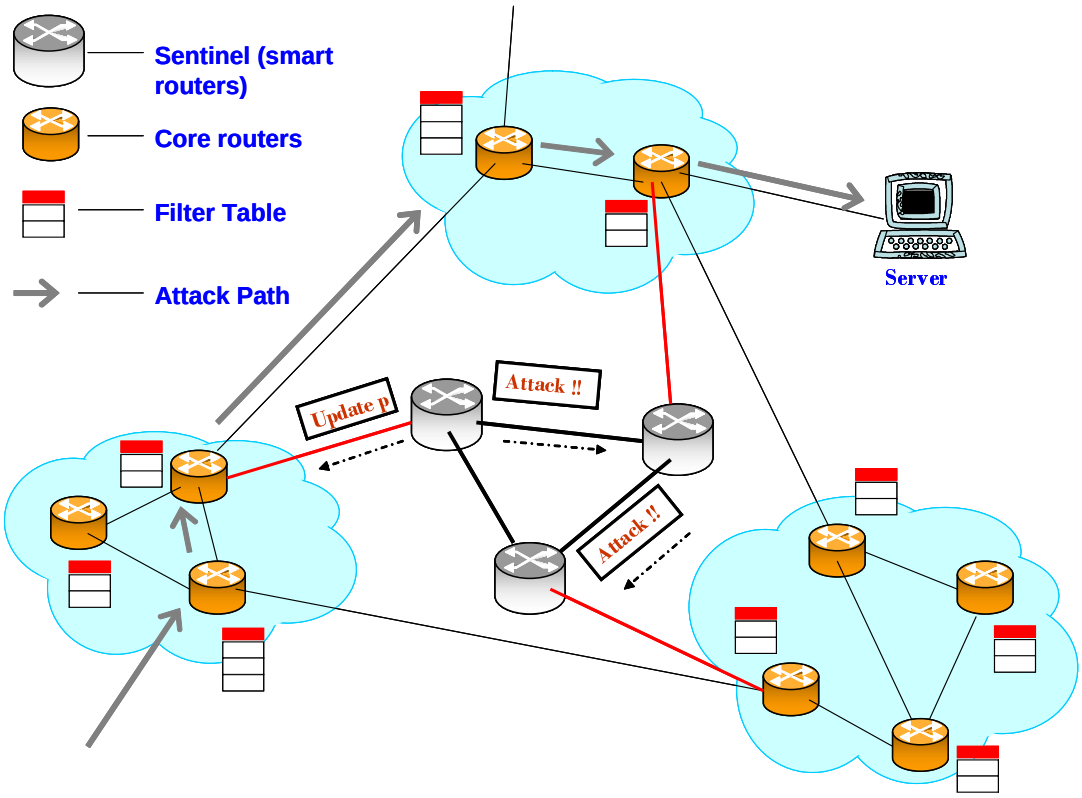

Figure 38: Collaborative Defense Architecture (CDA)

ated from it normal operation and hence would not inhibit its normal routing capabilities.

- "*Firewall-enabled routers*"

  These nodes are the filtering agents pre-assigned by the *Autonomous System (AS)* administrator. We propose to model them to include the hierarchical firewall optimization as discussed in Chapter 5. These routers receive messages from the *Active sentinels* and incorporate the anomalies by making online as well as offline changes to their rule-sets. The change can in the form of a "*re-splits*", "*re-promotions*" or "*reorders*".

7.2.3 Types of Communication in CDA

The proposed communication amongst the various nodes in the collaborative defense infrastructure are detailed below. These interactions help to achieve the mitigation of anomalies in the network. Due to the scale of the network system we propose the asynchronous method of message passing. The messages are passed between the *Active sentinel* and the *Agents/Routers* in a given *Autonomous System* or are passed between groups of *Active sentinels*. There are five basic types of communication as described in the following:

- *Active sentinel - Passive sentinel*

  This communication is from the Active sentinel to the Passive sentinel informing it to increase/decrease "p", the sampling rate to forward packets to the *Active sentinel*.

- *Passive sentinel - Active sentinel*

  The Passive sentinel samples packets at the assigned rate p and then sends the sampled information to the *Active sentinel*. It also receives messages from the *Active sentinels* to update p periodically.

- *Active sentinel - Firewall enabled router*

  In this communication the *Active sentinel* sends a list of rules whose priority is to be updated to the *Firewall enabled router*.

- *Firewall enabled router - Active sentinel*

  The *Firewall enabled router* receives the update message from *Active sentinels* and ac-

cordingly modifies its rule-set. The rules are either reordered or new rules are added in the form of default deny rules.

- *Active sentinel - Active sentinel*

  The communication between the *Active sentinels* helps to propagate local traffic information to the peering nodes. This is the basis for performing collaborative defense operation.

The basic message types for *CDA* are illustrated in Figure 39. There are three basic message types, *Activate*, *Update* and *Create*. The *Activate* message is sent by the *Active Sentinels* to their respective core nodes specifying them to forward the packet at a probability "p". The *Update* message is invoked when the traffic characteristics of a packet filter(s) exceeds the previous assigned threshold. The *Create* message is a variant of the *Update* message that helps to set a different threshold.

7.3 COLLABORATIVE DEFENSE OPERATION

7.3.1 Optimal Sentinel Placement

In its most general form, the *Optimal Sentinel Placement*(OSP) problem seeks to determine a minimum number of sentinels, S, such that any core node, i, is *covered* by at least one node in S. This is necessary to ensure that any traffic entering the network is ultimately inspected by the active sentinels. To illustrate this, consider the case where core routers A and B use adjacent router C to forward traffic to its destination. It is clear that activating router C to forward probabilistically selected traffic to the covering active sentinel is sufficient to cover router A and C. The minimum cardinality of the dominating set S is denoted by C', and is called the domination number. The OSP problem is closely related to the *Dominating Set* (DS) problem, where coverage represents adjacency to a sentinel in S. The DS problem can be formally described as follows:

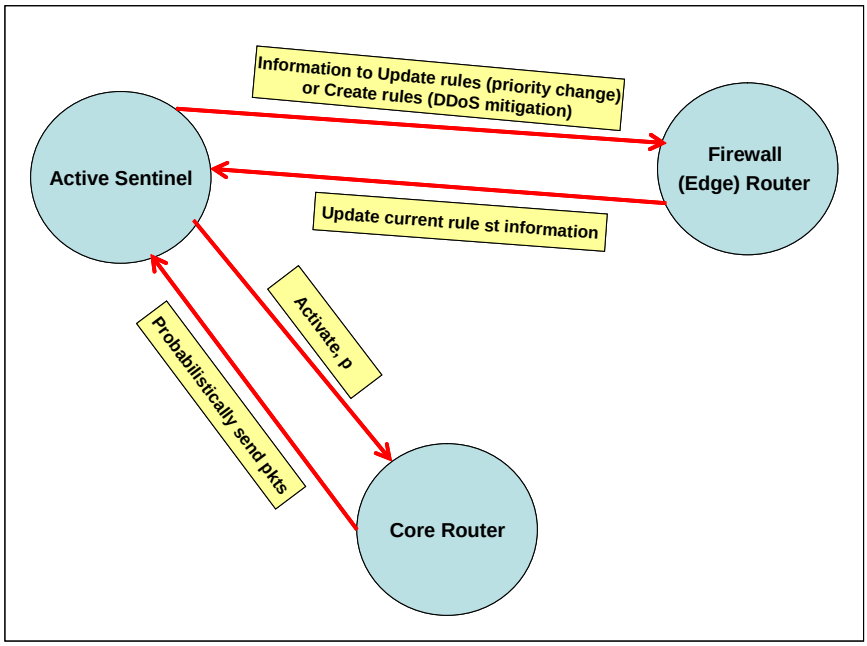

Figure 39: Basic Message in CDA

Given a graph $G = (V, E)$, where V represents the set of vertices and E the set of edges, the DS process divides V into a collection of subsets $V_1, V_2...V_n$, such that $V = \bigcup i = 1...n$ V_i and each subset V_i induces a connected subgraph of G. Notice that subsets need not be disjoint. Furthermore, if the graph induced by S is connected, it is called a connected dominating set. The computation of a DS of minimum cardinality for arbitrary graphs is known to be NP-complete [10]. Several heuristics have been proposed to produce a near-optimal solution for a variety of problems [13]. These heuristics, however, fall short in addressing the major requirements of proposed defense system. The vulnerability of sentinels to attacks, coupled with potential sentinel failures or network disconnection, imposes a variety of conditions that must be satisfied in order to produce a resilient, secure and fault-tolerant IDS defense. In order to ensure a high level of redundancy and resiliency against attacks directed at the defense system, we ensure that each core node in $V - S$ be dominated by at least r sentinels in S, for a fixed positive integer r. Furthermore, in order to ensure that the communication delay between core routers and sentinels is bound, we require that each core node $V - S$ be within distance d of at least r vertex in S, for a fixed positive integer d. We refer to this problem as the $(r, d) - OSP$ problem [31, 32]. The problem of computing $(r, d) - OSP$ of minimum cardinality for arbitrary graphs is *NP-Complete*. In our future research we will attempt to design heuristics to compute an approximate solution to this problem.

7.3.2 Probabilistic Packet Inspection

The approach used in this research to reduce the packet inspection workload builds on the observation that effective intrusion detection can be achieved based on a reduced amount of packet sampling and logging. The proposed research direction is similar to the work presented in [18] and uses random sampling to inspect packets. To illustrate this process, the following simple, traffic sampling model is used as discussed. Assume that a flow is considered suspicious if its bandwidth usage exceeds a threshold percentage, $t\%$ (t is specified by the network administrator) of the link capacity. These flows are labeled as delinquent

and become subjected to thorough inspection. In order to reduce false alarms, packets are sampled randomly such that each byte of the packet has equal probability, p, of causing the flow to become delinquent. At the end of the sampling period, flows that have been identified as suspicious can be inspected more thoroughly before alerts are generated. In our future research we aim to design a distributed probabilistic packet inspection algorithm for the proposed collaborative defense infrastructure.

7.3.3 Dynamic Collaborative Packet Filtering

In this thesis we have developed effective, traffic-aware optimizations to improve the operational cost of firewalls for both centralized and de-centralized firewall operation. The proposed techniques "adapts" rule-set configurations to the dynamically changing network traffic characteristics, while maintaining policy integration across the different networks. A unique feature of the approach is its adaptive anomaly detection and countermeasure mechanism, used to dynamically alter the firewall rule-set to improve performance.

The strong motivation to the online dynamic packet filtering approach is the observation that the major portion of the network traffic matches only a subset of the field values in the security policy rules. This is known as the $80-20$ rule, which states that 80% of the traffic is filtered by 20% of the rules. One of the important traffic characteristics commonly observed in our analysis of large number of Internet traces is the skewness of the traffic matching in the policy, which reveals that the majority of inbound or outbound packet is matched against a small subset of filtering field values which exists in the security policy implemented by the firewalls. What is important of this property is that the traffic skewness property is unlikely to change over a short period of time, and the total number of different skewness property is unlikely to be large in a firewall policy. As future research we aim to embed our adaptive dynamic packet optimization technique developed in this thesis to the proposed *Collaborative Defense* infrastructure. The proposed architectures and algorithms are imported to the Firewall Enabled edge routers to perform efficient traffic-aware filtering and attack mitigation.

These approaches mitigate *out-of-network* or *denial-of-service (DoS)* attacks. In order to handle distributed in-network attacks (DDoS) the firewall routers depend upon the collaborative action taken by the smart router nodes. We aim to apply various traffic characteristic information to determine the distributed attacks for such network systems.

The basic operation of an *Active sentinel* is depicted in Figure 40. In the following we discuss the steps of the collaborative defense action. In the first step of the defense operation we aim at developing a novel node placement algorithm in the proposed separate security plane.

In the next step we aim at the design and implementation of a simple probabilistic packet inspection mechanism amongst the smart defense node in the separate security plane. The probabilistic approach in packet inspection is due to the large volume of the network traffic and the fast link speed operations of *Tier-1 ISPs*.

In the third step we aim to import all the de-centralized traffic aware optimizations as discussed in Chapter 5 into the edge firewall nodes to enable fast and traffic-aware packet filtering and attack mitigation. The implementation of these approaches help to mitigate *out-of-network* and *denial of service* anomalies/attacks. This step however does not answer to the *in-network* attacks.

In the last step our goal is the design of efficient but simple techniques to recognize the presence of *in-network (stealth)* anomalies in the distributed collaborative framework. The approach is proposed to be based on distributed statistical inference of traffic characteristics amongst the smart defense nodes. The smart nodes would aggregate the information received from one another and monitor (check) to detect any possible threshold violations. We propose that the threshold information be specified by the network administrator. If a violation of threshold is detected, the smart nodes immediately inform the edge firewall (filtering) nodes to take appropriate action, either to *promote* the filter or *create* a new filter corresponding to the misbehaving packet and thus limit or mitigate the developing anomaly.

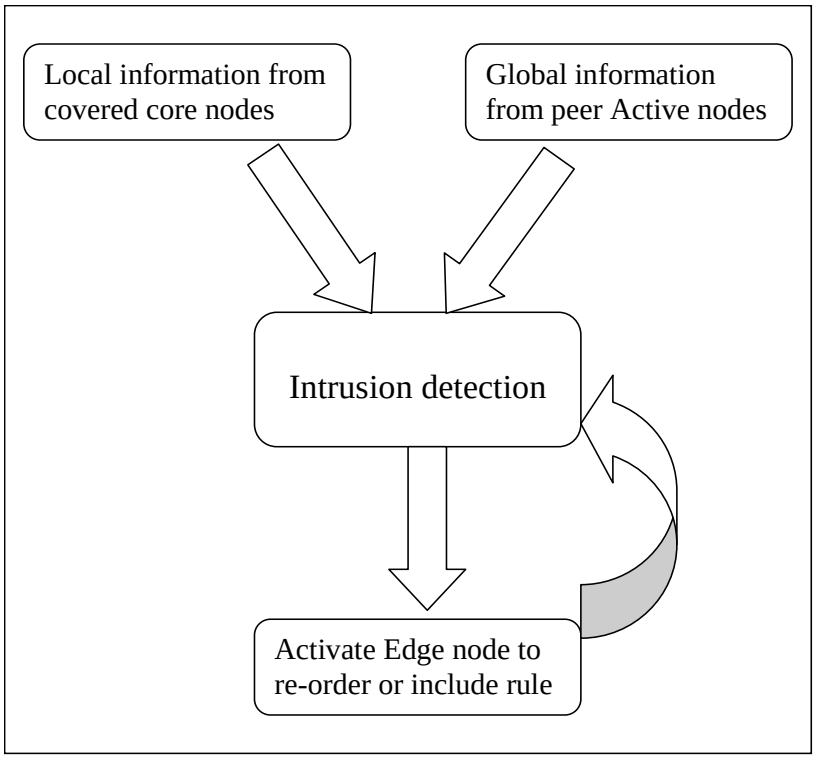

Figure 40: Basic Active sentinel Operation

This in consequence helps to limit the distributed attack.

7.4 SUMMARY

In the chapter we present a future research direction aimed at the design of a dynamic collaborative defense infrastructure to detect and limit distributed attacks. We propose the design of a separate security plane to manage and collaborate information between the various network elements. We believe that such a global and co-ordinated approach would be real-time and cost-effective with very minimal changes to the current network infrastructure. Moreover it would also addresses scalability and promises ease of deployment. Along with the collaborative defense model, *CDA* benefits from the dynamic optimized packet filtering approaches proposed in the thesis research. The dynamic de-centralized firewall optimizations helps to enable dynamic defense in such large scale network environments. Our long term future direction is for the *CDA* infrastructure to be deployed in a real network environment.

BIBLIOGRAPHY

[1] Symantec internet security threat report, http://eval.symantec.com.

[2] Linux ipchains, http://people.netfilter.org/ rusty/ipchains.

[3] Denial of service, http://www.cert.org/homeusers/ddos.html.

[4] Checkpoint ngx firewall, http://www.checkpoint.com.

[5] S. Acharya, M. Abliz, B. Mills, A. Greenberg, T. Znati, Z. Ge, and J. Wang. Optwall: A hierarchical traffic-aware firewall. 14th Annual Network and Distributed System Security Symposium, San Diego, CA, February, 2007.

[6] S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greenberg. A Traffic-Aware Framework and Optimization Strategies for Large Scale Enterprise Networks. *Technical Report*, pages 1–20, September 2005.

[7] S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greenberg. Traffic-aware firewall optimization strategies. In *IEEE International Conference on Communications*, Istanbul, Turkey, June 2006.

[8] Ehab Al-Shaer and Hazem Hamed. Modeling and management of firewall policies. *IEEE Trans. Network and Service Management*, 1(1), Apr 2004.

[9] F. Baboescu and G. Varghese. Scalable packet classification. Proceedings of ACM SIGCOMM, 2001.

[10] M. R. Barey and D. S. Johnson. *Computer and Intractability.* Freeman, San Fransisco, 1978.

[11] P. Brucker. On the complexity of clustering problems. In *in Optimization and Operations Research.* Springer-Verlag, pp. 45-54, 1977, 1997.

[12] Moses Charikar, Sudipto Guha, Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. ACM Symposium on Theory of Computing, 1999.

[13] G. Chen, F. Nocetti, J. Gonzalez, and I. Stojmenovic. Connectivity-based-k-hop clustering in wireless networks. Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS 02), IEEE Computer Society, 2002.

[14] C. Douligeris and A. Mitrokotsa. Ddos attacks and defense mechanims: classification and state of the art. Computer Networks, Vol. 44(5), pp. 643-666, 2004.

[15] T. Dubendorfer, A. Wagner, and B. Plattner. An economic damage model for large-scale internet attacks. Proceedings of the 13th International Workshop on Enabling Technologies, June, 2004.

[16] Adel El-Atawy, Taghrid Samak, Ehab Al-Shaer, and Hong Li. Using online traffic statistical matching for optimizing packet filtering performance. INFOCOM, 2007.

[17] Pasi Eronen and Jukka Zitting. An expert system for analyzing firewall rules. In *Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec 2001)*, pages 100–107, Copenhagen, Denmark, November 2001.

[18] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. ACM SIGCOMM Internet Measurement Workshop, 2001.

[19] A. Feldmann and S. Muthukrishnan. Tradeoffs for packet classification. In *IEEE INFOCOM*, March, 2000.

[20] Errin W. Fulp. Optimization of network firewalls policies using directed acyclic graphs. In *Proceedings of the IEEE Internet Management Conference*, 2005.

[21] Errin W. Fulp. Parallel firewall designs for high-speed networks. In *IEEE INFOCOM High Speed Networking Workshop*, 2006.

[22] P. Gupta and N. McKeown. Packet classification using hierarchical intelligent cuttings. In *Proceedings of Hot Interconnects*, 1999.

[23] P. Gupta and N. McKeown. Algorithms for packet classification. in IEEE Network, Vol. 15, No. 2, pp. 24-32, 2001.

[24] P. Gupta, B. Prabhakar, and S. Boyd. Near optimal routing lookups with bounded worst case performance. In *IEEE INFOCOM*, 2000.

[25] Pankaj Gupta and Nick McKeown. Packet classification on multiple fields. In *Proceedings of SIGCOMM*, 1999.

[26] Hazem Hamed and Ehab Al-Shaer. Dynamic rule-ordering optimization for high-speed firewall filtering. In *ASIACCS*, 2006.

[27] Hazem Hamed, Adel El-Atawy, and Ehab Al-Shaer. Adaptive statistical optimization techniques for firewall packet filtering. In *IEEE INFOCOM*, April, 2006.

[28] Susan Hinrichs. Integrating changes to a hierarchical policy model. In *Proceedings of 9th IFIP/IEEE International Symposium on Integrated Network Management*, Nice, France, 2005. IEEE.

[29] http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html. Symantec security response w32.blaster.worm, 2003.

[30] http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html. Symantec security response w32.sasser.worm, 2004.

[31] N. Jariyakul and T. Znati. On the internet delay-based clustering. Proceedings of the 38th Annual Simulation Symposium, April 2005.

[32] N. Jariyakul and T. Znati. A clustering-based selective probing framework to support internet quality of service routing. Distributed Computing - IWDC 2005, in Proceedings of the 7th International Workshop, Kharagpur, India, December, 2005.

[33] T. V. Lakshman and D. Stidialis. High speed policy-based packet forwarding using efficient multi-dimensional range matching. In *Proceedings of SIGCOMM*. ACM Press, 1998.

[34] R. Lemos. Msblast epidemic far larger than believed. http://news.com/msblast, 2004.

[35] A. J. McAulay and P. Francis. Fast routing table lookup using cams. Proceedings IEEE INFOCOM, 1993.

[36] J. Mirkovic, G. Prier, and P. Reiher. Attacking ddos at the source. Proceedings of ICNP, November 2002.

[37] J. Mirkovic and P. Reiher. A taxonomy of ddos attack and ddos defense mechanisms. Computer Communications Review, Vol. 34(2), pp. 39-52, 2004.

[38] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. IEEE Security and Privacy, July, 2003.

[39] W. Noonan and I. Dubrawsky. *Firewall Fundamentals: An introduction to network and computer firewall security.* June, 2006.

[40] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan. Cossack: Coordinated suppression of simultaneous attacks. DARPA Information Survivability Conference and Exposition, Washington, DC, 2003.

[41] Jiang Qian, Susan Hinrichs, and Klara Nahrstedt. ACLA: A framework for access control list (acl) analysis and optimization. In *Communications and Multimedia Security*, 2001.

[42] Lili Qiu, George Varghese, and Subhash Suri. Fast firewall implementations for software-based and hardware-based routers. In *SIGMETRICS '01: Proceedings of the 2001 ACM*

SIGMETRICS international conference on Measurement and modeling of computer systems, pages 344–345, New York, NY, USA, 2001. ACM Press.

[43] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, pages 91–92, New York, NY, USA, 2002. ACM Press.

[44] Summet Singh, Florin Baboescu, George Varghese, and Jia Wang. Packet classification using multidimensional cutting. In *SIGCOMM*, 2003.

[45] V. Srinivasan, S. Suri, and G. Varghese. Packet classification using tuple space search. In *Proceedings of SIGCOMM*. ACM Press, 1999.

[46] V. Srinivasan and G. Varghese. Fast address lookups using controlled prefix expansion. ACM Transactions on Computer Systems, Vol. 17, No. 1, 1999.

[47] Stephen J. Tarsa and Errin W. Fulp. Trie-based policy representations for network firewalls. In *Proceedings of the IEEE International Symposium on Computer Communications*, 2005.

[48] T. Y. C. Woo. A modular approach to packet classification: Algorithms and results. in Proc. of IEEE INFOCOM, March 2000.

[49] Yang Xiang and Wanlei Zhou. Intelligent ddos packet filtering in high-speed networks. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, October, 2005.

[50] Xiao-Ling Zhao and Ji-Zhou Sun. A parallel scheme for ids. Proceedings of the Second International Conference on Machine Learning and Cybernetics, November, 2003.