DISTRIBUTED PERIMETER FIREWALL POLICY MANAGEMENT FRAMEWORK

A DISSERTATION IN
Computer Science
and
Telecommunications and Computer Networking

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment of
the requirements for the degree

DOCTOR OF PHILOSOPHY

by
MAHESH NATH MADDUMALA

M.Tech., Andhra University, India, 2007
B.Tech., M.L.Engineering College, India, 2004

Kansas City, Missouri
2017

DISTRIBUTED PERIMETER FIREWALL POLICY MANAGEMENT FRAMEWORK

Mahesh Nath Maddumala, Candidate for the Doctor of Philosophy Degree

University of Missouri-Kansas City, 2017

ABSTRACT

A perimeter firewall is the first line of defense that stops unwanted packets (based on defined firewall policies) entering the organization that deploys it. In the real world, every organization maintains a perimeter firewall between internet (which could be untrusted) and its own network (private network). In addition, organizations maintain internal firewalls to safeguard individual departments and data center servers based on various security and privacy requirements. In general, if we consider firewall setup in multinational organization's network environment, every branch has perimeter firewall and a set of internal firewalls. Every branch has its own security policies defined based on its specific security requirements, type of information, information processing systems, location-based compliance requirements, etc. As the branches of the multinational organizations span across the globe, managing the policies at every branch and ensuring the compliance and consistency of security policies are quite complex. Any misconfiguration of firewall policies even at a single branch may pose risk to the overall organization in terms of financial loss and reputation.

In this dissertation, we present our framework to automate the policy management of distributed perimeter firewalls of a multi-national organization. We introduce new categories of policies to support centralized management of distributed firewalls and to ensure consistency and compliance of organizational and location-based policies. We define procedures for the initialization of firewall policies and policy updates. Our scheme is highly automatic that needs minimum human intervention to incorporate a set of new policies or update existing policies in distributed firewalls.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Graduate Studies, have examined a dissertation titled ”Distributed Perimeter Firewall Policy Management Framework,” presented by Mahesh Nath Maddumala, candidate for the Doctor of Philosophy degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Vijay Kumar, Ph.D., Committee Chair
Department of Computer Science Electrical Engineering

Lein Harn, Ph.D.
Department of Computer Science Electrical Engineering

Cory Beard, Ph.D.
Department of Computer Science Electrical Engineering

Praveen Rao, Ph.D.
Department of Computer Science Electrical Engineering

Yongjie Zheng, Ph.D.
Department of Computer Science Electrical Engineering

CONTENTS

ILLUSTRATIONS

Figure Page

TABLES

Table Page

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Vijay Kumar for his guidance, support and patience throughout my doctoral research. Dr. Kumar also supported me financially through National Science Foundation grant, which allowed me to dedicate most of my time to the research.

I sincerely thank my committee members Dr. Cory Beard, Dr. Lein Harn, Dr. Praveen Rao and Dr. Yongjie Zheng for their inputs and advices.

Finally, I would like to thank my family for their love and support throughout these years.

CHAPTER 1

INTRODUCTION

A firewall is the first line of defense in computer or network security that stops malicious or unwanted network packets entering the system. It controls the incoming and outgoing network traffic based on firewall rule specification. Firewalls are generally categorized as host-based firewalls and network firewalls. A host-based firewall is a software running on a single host that controls the incoming and outgoing network traffic of that host. A network firewall is typically a hardware security appliance controls the network traffic between two or more networks.

1.1 Firewalls Technologies

Since its inception in late 1980's, firewall technology has been evolving to meet the growing security requirements of safe communication. Recent advances in firewall technologies have deployed a number of techniques to meet security requirements. In this section, we will examine the various firewall technologies and their capabilities.

Packet Filtering: The main feature of the firewall is packet filtering. Firewall filters (stops packets entering to the network) the network packets based on the list of firewall rules. There are two types of filtering performed based on the direction of the flow of network packets – ingress and egress filtering. Ingress filtering is filtering the network packets that are entering the computer system or the network that firewall is protecting whereas egress filtering is filtering the packets that are leaving the computer system or network (Figure 1).

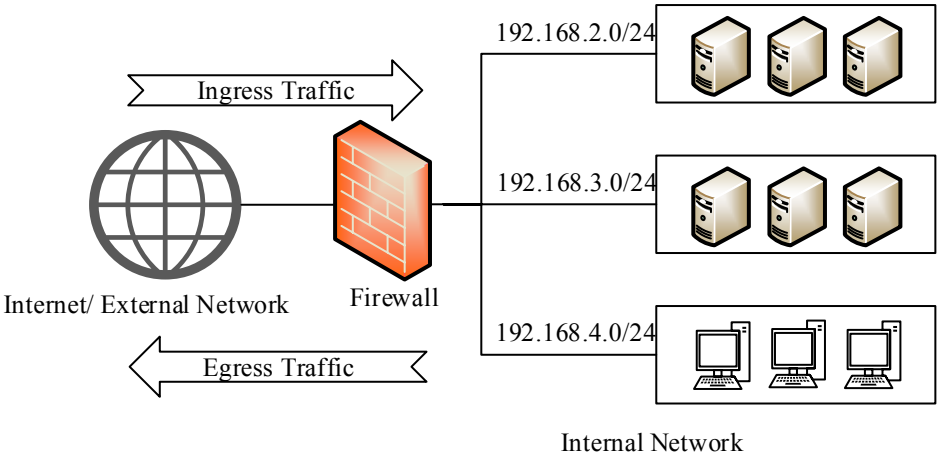

Figure 1. Firewall setup in a typical organization's network

Stateful Inspection: Stateful inspection examines the TCP header of the packets to monitor the state of each connection and records the connection state in a state table. Firewall uses this state table to check whether the incoming packets that are claimed to be part of the existing connection are valid or not. Stateless packet filtering is vulnerable to spoofing attacks due to the loopholes in the TCP/IP protocol stack. To prevent such attacks, it is required to first verify whether the incoming packet is part of an established connection or a new connection.

Application Layer filtering: Modern firewalls use application layer filtering to control the network access used by certain applications. Application layer filtering functions by inspecting the data at layer 7 or application layer of the protocol stack and filtering the packets based on application layer rules. To identify the application specific traffic, firewall inspects the header information of all the seven layers of protocol stack along with the payload of the packets.

In 2009, Gartner [36] coined the term "Next Generation Firewall" with the following capabilities at minimum:

1. Support in-line bump-in-the-wire configuration without disrupting network operations.

2. Standard first-generation firewall capabilities.

3. Integrated rather than merely collocated network intrusion prevention.

4. Application awareness and full stack visibility.

5. Extra firewall intelligence.

To provide security features such as stateful Inspection, continuous content inspection for malware detection, etc., firewalls have to be designed to be in-line security appliances. The in-line configuration reduces the performance of network communication and becomes bottleneck of the perimeter network. Next generation firewalls while supporting in-line bump-in-the-wire configuration should not disrupt network operations. For commercial purposes, vendors manufacture multiple devices to support multiple security feature such as packet filtering, intrusion detection and prevention, anti-malware, etc. Gartner [36] suggests that next generation firewalls should be capable of offering all these services at one place instead of distributed among multiple devices. The extra firewall intelligence capability in Gartner's [36] definition suggests to make a framework to generate dynamic policies to counter the unexpected or unanticipated and advanced attacks. This could be achieved by sharing and analyzing firewall packet statistics.

Though many commercial firewall vendors such as CISCO, checkpoint and Palo Alto have designed their next generation firewalls to meet most of the capabilities defined by Gartner [36], it is still a challenge to meet the requirements of ever-growing trends in network communications and a great amount of research has to be done in firewall technology.

1.2 Firewall Access Rules

Firewall access rules controls the network traffic by defining which network packets are allowed and which are denied. These rules are stored in firewall. When a network packet

arrives at the organization's network, firewall reads the contents of packets and compare against the elements of the firewall access rules sequentially from top to bottom. When a rule is matched, the action of that rule which can be allow or deny is applied and the subsequent rules are ignored. Thus, firewall always apply the action of the first matched rule. The last rule of any firewall rule set is a default rule that is applied when no previous rules are matched. The default rule is specified in either of two ways – "block by default" or "allow by default". The block by default rule blocks all traffic and rest of the rules define to allow desired traffic whereas allow by default rule allows all traffic and rest of the rules block unwanted traffic.

1.2.1 Layer 3 Rules

A layer 3 firewall rule is formed based on the protocol of the layer 3 of the OSI ISO protocol stack model. The rule at layer 3 consists of source and destination IP addresses, source and destination ports, protocol information and the action to be performed on the matched packets. Source IP address specifies the IP address or the network address of the origination of the traffic that is from the external network in incoming connection or from the internal network in outgoing connection. Destination IP address specifies the IP address or the network address where network traffic is destined. In the incoming connection, it is a part of the internal network and in the outgoing connection, it is a part of the external network. Similarly, source and destination ports specify the port information of the respective end machines. Protocol element specified the type of network traffic being communicated between two systems or networks. Action element species the action to be performed on the matched packets that is typically "allow" or "deny". Some firewalls also support "reject" action that is used to send an Internet Control Message Protocol (ICMP) response to indicate service is not available or not reachable.

We discuss the application of layer 3 firewall access rules with use cases. We define below a set of use cases that are chosen based on the network configuration (Figure 1).

Use case 1: Consider the network address 192.168.2.0/24 allocated to webservers and where only HTTP service is running (Figure 1). In this case, we want to allow only HTTP related incoming IP traffic to these webservers and block the rest of the IP traffic. We would like to have an access rule in our firewall to allow HTTP (port 80) based IP traffic and blocks the rest. Similarly consider the network address 192.168.3.0/24 allocated to FTP servers. In this case, we want to allow only FTP based traffic to these servers and block the rest of the IP traffic. We would like to have an access rule in our firewall that allows only FTP (port 25) based traffic and block the rest. The rules of these two requirements are defined in the Table 1. As we want to enforce these rules on inbound traffic, we assume that these rules are applied on the external interface of the firewall.

Table 1. Access rules for HTTP and FTP servers

Action	Protocol	Source IP	Source port	Destination IP	Destination port
Allow	TCP	Any	Any	192.168.2.0/24	80
Allow	TCP	Any	Any	192.168.3.0/24	25
Deny	Any	Any	Any	Any	Any

Use Case 2: Consider that the network address 198.168.4.0/24 is allocated for the generic purpose computers and only HTTP service is permitted to access the internet (Figure 1). In this case, we would like to have an access rule to allow the computers from the internal network to

access the internet or external network using HTTP service and block the rest of the communication. As we want to enforce these rules on outbound traffic, we assume these rules are applied on the internal interface of the firewall. The rules for this requirement is defined in the Table 2.

Table 2. Access rules to allow HTTP traffic

Action	Protocol	Source IP	Source port	Destination IP	Destination port
Allow	TCP	192.168.4.0/24	Any	Any	80
Deny	Any	Any	Any	Any	Any

1.2.2 Layer 7 Rules

To protect the internal systems or network from the malicious websites activities or to prevent access to unwanted sites or applications, firewall has to inspect the payload of the packets at the application layer (OSI layer 7 protocol stack model). To control the traffic at the application layer, firewall rules have to be defined at layer 7. Some examples of layer 7 rules are listed in Table 3. Suppose an organization decides to block the access to social networking websites to all its employees, it defines layer 7 rules (Rule 3 and Rule 4 from Table 3).

Table 3. Layer 7 rules

Rule No.	Action	Service Type	Value
1	Deny	Video Streaming	Youtube

Rule No.	Action	Service Type	Value
2	Deny	Video Streaming	Netflix
3	Deny	Website	Facebook
4	Deny	Website	Twitter

1.2.3 Temporal Policy Rules

Firewall temporal policy is a firewall policy that allows or denies a network packet based on specified day and time range of the policy in addition to the packet filtering rules. The time-based or temporal policies are necessary to control the traffic or internet based applications based on specific days and time duration. Table 4 lists some of examples of firewall temporal policies. . For example, an organization may decide to block access to social networking websites (Rule 3 and Rule 4 from Table 4) to its employees during working hours on weekdays in order to increase their productivity. Temporal policies can also be applied to control the bandwidth on certain days by restricting access to certain streaming websites or applications (Rule 1 from Table 4).

Table 4. Firewall temporal policies

Rule No.	Action	Service Type	Value	Days	Time
1	Deny	Video Streaming	Youtube	(Monday, Wednesday)	0800-1200
2	Allow	Video Streaming	Youtube	Anyday	Any time
3	Deny	Website	Facebook	Weekdays	0800-1700
4	Deny	Website	Twitter	Weekdays	0800-1700

1.2.4 Spatial Policy Rules

Attacks on an organization often originate from specific geo locations. It is necessary to stop all the traffic from such high unsafe locations. These locations are identified from the IP addresses. Though identifying locations with IP addresses is not a reliable option, but many commercial firewall vendors such as CISCO still continue to rely on IP addresses to block the certain or all the traffic from high unsafe geo locations.

1.3 Distributed Firewalls

The organizations with multiple subnetworks and diverse computing machines as shown in Figure 2 require multiple firewalls to be deployed in their perimeter network. A single firewall is not a viable solution in this case due to the following reasons.

1. Internal traffic cannot be trusted
2. Load balancing - To prevent a single firewall from being overloaded with firewall policies
3. Application aware firewalls have more granularity.

Every organization typically deploys two types of firewalls (a) perimeter firewall and (b) internal firewall. Perimeter firewall is deployed at the boarder of the perimeter network to protect the overall organization's network whereas internal firewalls are deployed to protect the individual subnetworks. Figure 2 illustrates the position of perimeter and internal firewalls in a typical organization.

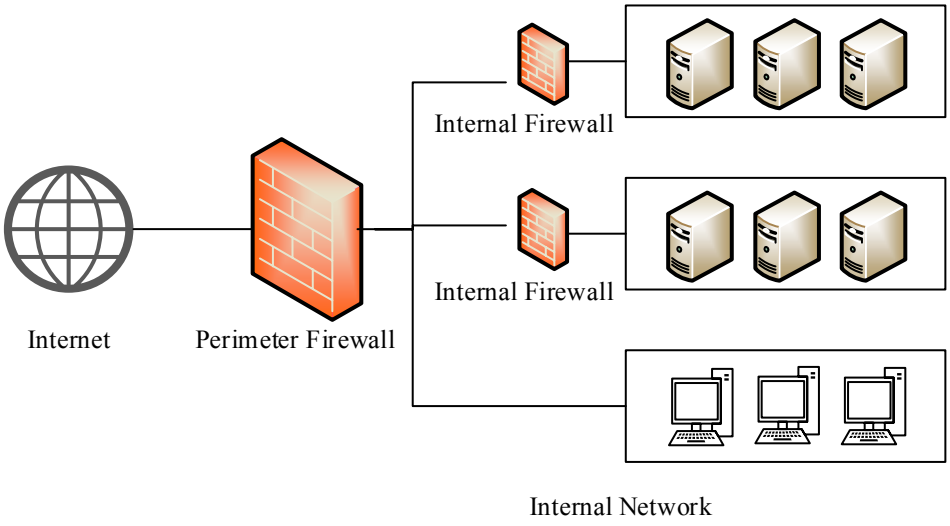

Figure 2. Distributed firewall setup in a typical organization

1.4 Scope and Contribution of the Dissertation

The main contribution of this dissertation is to provide a policy management framework for multi-national organizations to manage the firewall policies of distributed perimeter firewalls across the globe. Our framework addresses the critical policy management issues such as ensuring policy compliance, consistency checking and configuration of various categories of policies at every branch. The secondary contribution of this dissertation is to optimize the design of firewall temporal policies. We propose a mathematical approach to optimize the representation of temporal policies and we establish that our way significantly saves space and processing time to parse the temporal policies. The third contribution of this dissertation is to identify the unsafe locations of cyber-attacks in mobile communication. We consider the mobile cell global identity to identify the originated location of cyber-attacks and thus create a spatial policy to block internet traffic from high unsafe locations. To the best of our knowledge, so far the research on distributed perimeter firewalls has not appeared in the literature.

The scope of our policy management framework includes only perimeter firewalls of branches that belong to the same organization. Also our framework does not consider the policy management of internal firewalls.

1.6 Outline of the Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, we present a literature review of commercial firewalls and their policy specification formats. We also discuss the research work done by other researchers in distributed firewalls. In Chapter 3, we present our core research work on distributed perimeter firewalls policy management starting with categories of policies followed by global policy manager architecture and their procedures. In Chapter 4, we present efficient design of firewall temporal policies. In Chapter 5, we present the identification of unsafe locations to define the firewall spatial policies. Finally, we conclude our dissertation and briefly discuss the future research work in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

Most of the research on firewall is focused on packet classification, policy modeling, detection and resolution of policy conflicts, policy optimization and spatiotemporal policies in single firewall systems. However, significant research has not been done in distributed firewalls. Recently, researchers turned their attention to distributed firewalls to address the unique requirements of distributed computing based emerging technologies such as cloud computing, Internet of Things (IoT), Software Defined Networking (SDN), etc. In this chapter, we review related work on single firewalls in section 2.1 and related work on distributed firewalls in section 2.2.

2.1 Related Work on Single Firewalls

The basic functionality of a firewall is packet filtering. Firewall filters the packets by comparing and matching the various fields of packet headers with a set of firewall policy rules. To improve the performance of packet filtering, it requires efficient packet classification algorithms. There is a significant research work presented in [16, 17, 37, 39, 38, 40, 46] on this subject and as it is beyond the scope of our dissertation, we do not go in detail.

The policy description language plays a key role in understanding and analyzing the complex firewall policies to detect policy anomalies. Guttman [18] used LISP-like language to specify the global network access control policies. The author coined a new phrase "filtering postures" which is an assignment of inbound and outbound filtering constraints to router's

interface. The author presented algorithms to compute a set of filters for the individual routers from given network topology and then to compare these filters to global policies to check for any policy violations.

Hazelhurst et al. [22] tried to improve the firewall rules lookup latency by representing rule lists with Binary Decision Diagrams (BDD). The authors achieved this by converting rule sets into Boolean expressions and analyzing the rules by querying the Boolean expressions. Though the approach of optimizing the ruleset representation using binary expressions is effective, it is highly difficult for a human user to understand and interpret this representation in case of debugging. Yuan et.al. [49] also used the same approach of representing firewall rules with binary decision diagrams to develop their tool called Fireman.

Eronen and Zitting [16] designed a constrained logic programming-based system to model firewall policies that deals with conflict resolution only. Bandara et al. [5] also used logic programming to represent and resolve policy conflicts.

Some researchers used reverse engineering approach to extract firewall policies from firewall configuration files to create a platform-independent policy model. The firewall management tools such as Firmato [6], and Fang [29, 30] parse the various vendor-specific low-level firewall configuration files and generates an internal representation of the implied policy and network topology. They also provides an interface to query the policy database. Tongaonkar et al. [45] proposed a method to infer the higher-level security policies from the lower-level firewall rules by extracting hosts information, protocols and types of services.

An extensive research work has been done in the area of detection and resolution of policy conflicts in a single firewall. The research work presented in [20, 21, 1, 2, 9, 12, 14, 28, 32, 33, 41] proposed various methods to detect and resolve policy conflicts. Zhang et al. [50]

proposed a unique method of conflict-free policy specification so that the need of detection and resolution is avoided.

Hongxin et.al. [23] designed and implemented a framework of robust firewalls for software-defined networks. The main outcome of their work is detection and resolution of flow policy violations. They used the detection mechanism of examining flow path spaces against the authorization spaces that is composed from firewall rules. To detect flow policy violation, they defined entire violation and partial violation based on level of inclusion of tracked space in denied authorization space. They tried to resolve flow policy violations using four strategies - dependency breaking, update rejecting, flow removing, and packet blocking. The main drawback in their design is limiting the flow parameters to source IP and destination IP address. In addition, the scope of their framework is limited to single firewall in software-defined networking and could not be adopted to distributed firewalls.

2.1 Related Work on Distributed Firewalls

Most of the research contributions either are on single firewall or distributed firewalls that are located within the same network perimeter. There is no significant research on policy management of distributed perimeter firewalls. We review some of the related works.

Cremonini et al. [11] present an XML based formalism of semantic-aware perimeter protection. They present an XML syntax based filtering rules which do not go into much detail, nor consider enforcing centralized policies that react dynamically to varying environmental conditions. Zhang et al. [50] presented conflict-free policy resolution approach for multi-domain networks in a single firewall.

In order to avoid processing overheads and a single point of failure, Bellovin [7] proposed distributed firewalls. His definition of a distributed firewall is to have a centrally

configured organizational policy to be distributed to all the hosts residing within the organization. Every host has its own firewall to enforce the policies. Ioannidis et al. [24] provide an implementation of [7] using an open BSD. This implementation uses KeyNote, a system that uses public key cryptography for authentication and decentralized trust management. Unlike [7, 24], in our framework, we define distributed firewalls as the perimeter firewalls which are connected to a centralized controller

Gangadharan et al. [13] proposed distributed micro-firewalls where all the hosts in a private network are equipped with the micro-firewalls and these micro-firewalls are connected to a centralized policy manager. The functionality of the policy manager is to supervise the distributed intrusion system and alert the hosts with any policy change. Additionally gateway firewall is deployed at the gateway of the network and configured with the global policies. Though our work has similarities with this line of work in the area of dynamic update of policies, we developed a method of creating dynamic policies by analyzing firewall packet statistics instead of relying on third party IDS as in case of [13].

To avoid the weakness of the software-based firewalls, Payne et al. [35] presented an architecture of distributed embedded firewalls. They proposed firewalls are embedded on host's Network Interface Card (NIC) independent of host's operating system. The policy distribution is taken care by a policy server and in turn an individual embedded firewall has to report the audit information back to the policy server. They also discussed the applications and uses of embedded firewalls.

Meredith [31] presented an architecture of automatic distributed firewalls in which all the firewalls are connected to a centralized policy server. The main functionality of the policy server is to manage policies of all the distributed firewalls. Policies are automatically generated

based on the services used by the clients. Further, network traffic matching a firewall rule is audited and sent to the policy server. Here the scope of the automation of policies only limited to the initial discovery of policies and have not considered the audit information of the network traffic which plays an important role to improve firewall policies.

Al-Shaer et al. [3, 4] presented the conflict classification and analysis in cascading firewalls where firewalls are placed in a sequential manner starting with perimeter firewall followed by the internal firewalls. In our framework, we consider only perimeter firewalls but not the internal firewalls and the central controller manages all the perimeter firewalls of an organization distributed across the globe. CISCO and other firewall vendors also support the configuration of group policies which can be applied for a group of subnetworks within the same network perimeter. Gouda et al. [15] proposed Firewall Decision Diagrams [FDD] based verification of distributed firewalls. However, their approach is not suitable for firewalls that are distributed across multiple geo-locations.

2.2 Related Work on Temporal and Spatial Firewall Policies

Firewall Time-based filters are widely in use to control network traffic based on time. Vendors such as CISCO [10] and Palo Alto [34] equipped their firewalls with time based policies. Temporal policy specification is also available in IP tables [19].

There has not been significant research done on the design of time-based firewall policies. To the best of our knowledge, only the works reported in [43, 42] used time-based policies in resolving conflicts. Thanasegaran et al. [43] used BIt-vector based Spatial CALculus (BISCAL) [41] and characterization vectors to detect the conflicts by analyzing the n+1dimensional topological relationship between the firewall policies. Although they used

time-based approach, the representation of week day's list is not optimized, rather they only focused on detection of conflicts associated with space and time parameters of the policy.

Thanasegaran et al. [42] improvised their work [43] by proposing two phase approach to detect conflicts in firewall time-based policies. In first phase, they used a mapping mechanism to remove the repetition of the periodic filters like every specific day of the week. In the second phase, they used time divisor method to decompose time into intervals to analyze conflicting policies within each time interval. The authors mainly focused on detecting the conflicts in time-based policies effectively and not on designing the time-based policies efficiently.

In our work, we have introduced (a) an optimized way of representing temporal policies by using numeric approach and (b) set operations over weekdays for anomaly detection which reduces the processing time to compare list of weekdays.

Bhatti et.al. [8] developed the policy mapper, an administrative tool to administer location-based access control policies at both conceptual and logical levels. They identified the requirement of providing location-based access in addition to role-based access in certain scenarios and developed conceptual level of access control based on GeoRBAC model, a geospatial extension of the Role Based Access Control (RBAC) model.

CHAPTER 3

DISTRIBUTED PERIMETER FIREWALL POLICY MANAGEMENT

The security requirements of every organization, small or big, frequently change with its internal reorganization and its business dynamics. Successful timely implementation of appropriate firewall policies to meet the new security requirements by updating the firewalls is necessary to protect the organization from new threats. Currently, such policy updates are more or less, manual and are inserted by security staff. This approach has never very satisfactory and could compromise organizational security, especially for new platforms such as cloud. Recognizing the urgency of this requirement, we present a management scheme that offers dynamic firewall update facility needing minimum or no human intervention. Our scheme efficiently handles perimeter firewalls and policy updates for the entire network and individual zones as well.

Every organization typically deploys two types of firewalls (a) perimeter firewall and (b) internal firewall. Perimeter firewall is deployed between internet and organization's network. It protects the entire organization's network by filtering internet packets entering the organization. Internal firewalls are deployed between subnetworks or subdivisions of an organization. They protect the individual subnetworks which could serve individual department or division or a branch. A multi-branch organization deploys a single perimeter firewall at every branch and multiple internal firewalls within a branch. Figure 3 illustrates setup.

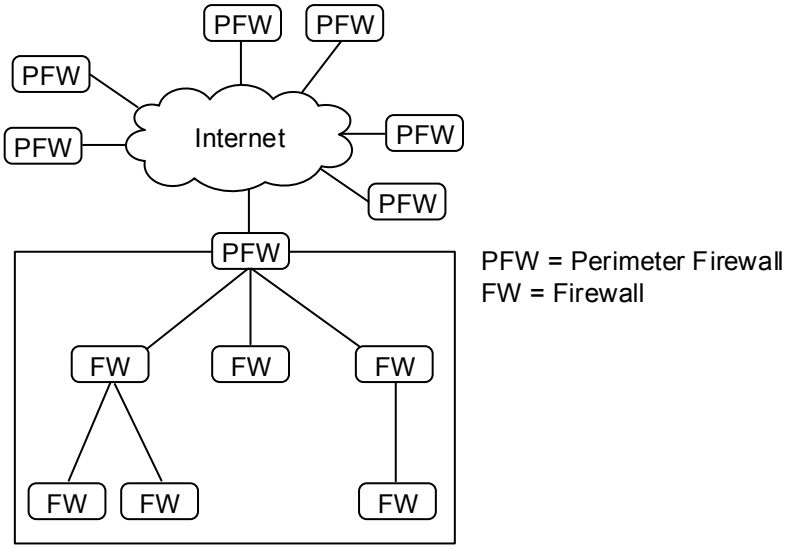

Figure 3. Distributed firewalls in a multi-branch organization

3.1 Categories of Policies

In a multiple branch organization setup, defining security policies for every branch and ensuring compliance, consistency and correctness of these policies are quite complex. Even a small error in policy configuration may result in huge financial loss and loss of reputation. In our framework, we address the above-mentioned security issues using a hierarchical structure of policies which ensures their compliance, consistency and correctness. Our firewall policies have three categories (a) compliance policies, (b) group policies and (c) local policies. These are based on the scope of the security requirements of the organization. Under these categories, our framework represents the entire firewall policies (complete firewall policies) as,

Complete Firewall Policies = {compliance policies} + {group policies} + {local policies}

Compliance policies: In our framework, we define two types of compliance policies – organizational compliance policies and location-based compliance policies. Organizational compliance policies are the mandatory policies that comply with the organization's internet

regulations. Location-based compliance policies are the policies based on the internet regulations governed by the individual countries where a branch of an organization is located. There are two types of location-based compliance challenges - one is the enforcement of data residency regulations (certain sensitive information must not leave the country) and the other is the enforcement of internet censorship (certain censored information must not enter the country). The first case has been addressed by various governmental compliance policies such as Health Insurance Portability and Accountability Act (HIPAA), Payment Card Industry Data Security Standard (PCI DSS), etc. The second case is enforced by the individual countries such as China, Iran, Saudi Arabia, etc., using firewalls and other filtering mechanisms. For example Iran [48] blocks almost 50% of the top 500 visited websites worldwide including YouTube, Facebook, Twitter, and Google Plus. China [47] governmental authorities not only block websites but also monitor the internet access of individuals. Unfortunately, such restrictions present serious compliance challenges. In this section, we deal with second case of location-based compliance issues.

An organization may have global organization level policies, national level policies, and state or province level policies and so on. We have represented this relationship in a hierarchical structure with organization level compliance policies at level 1 (L1) and national level compliance policies at level 2 (L2), … level n (Ln). This mode of representation offers an efficient and error-free way of manipulating (updating, removing, etc.) security policies. Compliance policies at any level is the combination of its own local compliance policies and compliance policies of its proper ancestors. For example, a branch of an organization located in Kansas city, Missouri, USA has to adhere to the organization compliance policies, USA (national) compliance policies and Missouri (state) compliance policies if any. Consider for

instance, “Blocking adult sites” can be an organization level policy and should be applied to all the business units or branches working under that organization. A branch of that organization in a country may have a national compliance policy of “Block the Facebook” which should be followed by all the business units working in that country. A branch in another country of the same organization may not have this policy

In Figure 4, *oc* represents organizational compliance policies, *nc1* and *nc2* represents national compliance policies of two different nations, sc1 and *sc2* are state compliance policies belonging to two separate nations and *f1* to *f5* represent the perimeter firewalls of the different branches of the same organization located in different geographical regions.

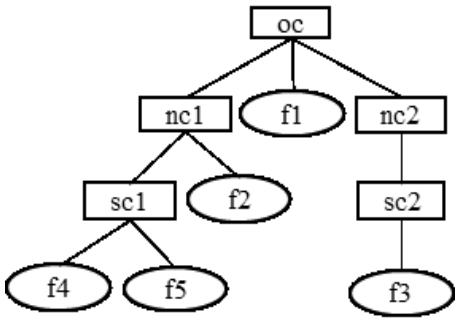

Figure 4. Hierarchical model of compliance policies

The concept of compliance policies is explained below with various cases using Figure 4.

Case 1: countries with stringent internet regulations may have restrictions on certain type of network traffic. Every business unit of a multinational established in multiple countries have to comply with those restrictions. Additionally, individual states or provinces of these countries may have their own restrictions on a different type of network traffic. Business units in those

states must comply with the state-wise restrictions in addition to the nation level restrictions. Perimeter firewalls *f3*, *f4* and *f5* come under this category where compliance policies at *f3* is

{oc} + {nc2} + {sc2} and compliance policies at *f4* and f5 are *{oc} + {nc1} + {sc1}*.

Case 2: If the restrictions are only at national level and not at the state level, then the business units or branches of the organization must comply only with national compliance policies. Compliance policies at *f2 = {oc} + {nc1}* is an example of this case.

Case 3: If there are no restrictions on network traffic at national and state levels where a branch of an organization is established then only the organizational compliance policies would be considered. Compliance policies at *f1 = {oc}* is an example of this case.

Changes and effects: If there is a change in compliance policies at any level then a policy update has to be propagated to all the perimeter firewalls of the branches located at and below that level. Suppose if there are compliance policy changes at nc1, then those changes have to be propagated to the perimeter firewalls *f2*, *f4* and *f5*.

Group Policies: We define group policies as security policies created for a set of requirements with common attributes. Common attributes may include, but not limited to, branches working with the same client, branches with similar type of operations such as financial, datacenters, etc. All participating business units must adhere to these group policies. Further, a business unit may participate in more than one group. In general, total group policies of a business unit is the union of individual group policies. Thus,

Total group policies = {group policies of *G1*} U {group policies of *G2*} U ... U {group policies of *GK*}, where group, *Gi* is a set of participating business units and *Gi* ⊆ {all business units of an organization}.

Group policies are explained in the following examples:

Example 1: Suppose *FW1* to *FW6* are perimeter firewalls associated with groups *G1, G2 and G3* and group *G1* is composed of {*FW1, FW2, and FW3}, G2* is composed of *{FW1, FW4}* and *G3* is composed of *{FW3, FW5, and FW6}*. Wwe observe that *FW3* participates in both groups *G1* and *G3*. So, the total group policies at *FW3* = group policies of *G1* and *G3*.

Example 2: Consider a university with departments *{CSE, TCN, Physics, Medicine}*, Libraries *{CSE Library, TCN Library, Med Library}* and Labs *{CSE Lab, TCN Lab, Phy Lab, Med Lab}*. Here the groups can be defined as:

Library group G1 = {CSE Library, TCN Library, Medicine Library}

CSE group G2 = {CSE, CSE Library, CSE Lab}

Medicine group G3 = {Medicine, Med Library, Med Lab}.

In this case, total group policies of CSE Library consists of group policies of library group *G1* and CSE group *G2*.

Changes and effects: If there is a change in common policies of a particular group then that change has to be applied to all firewalls of that group. Consider Example 1, if there is any change in policies of group *G2* then that change has to be applied to *FW1* and *FW4* firewalls.

Local Policies: We define local policies as the perimeter firewall policies to meet the security requirements of a business unit. In addition to the compliance policies and group policies, every business unit maintains its own local policies. They are solely defined and owned by that business unit.

Changes and effects: Since the scope of local policies is limited to the firewall itself, changes in local policies do not affect other firewalls.

3.2 Standalone vs Connected Firewalls

In a conventional approach of a multi or a single organization, firewall policies are manually configured at every location. If there is a change in an organization's compliance polices or group policies then this policy change has to be reflected at every firewall. The addition of new policies could lead to policy conflicts and could affect consistency and completeness of policies. Furthermore, handling and resolving these conflicts manually at an individual firewall is a big challenge to every organization.

In order to take care of these challenges, in our approach, we propose that all firewalls must be connected to a centralized controller so that the centralized controller would handle tasks such as configuration of firewalls, policy changes, policy conflict detection and resolution. This would avoid the erroneous manual configuration of policies and ensure the consistency and completeness of organization's security policies. The other advantage of connected firewalls is that every firewall can share its statistics to the centralized controller and thus helps in identification of potential network attacks.

3.3 Decentralized vs Centralized Topology

In connected systems, choice of topology plays an important role in overall performance of the system. The four common topologies that are in use in the internet are centralized, decentralized, hierarchical, and ring topologies. However, only centralized and decentralized topologies are feasible for the distributed perimeter firewall setup. In this section, we compare centralized and decentralized topologies with respect to distributed perimeter firewalls and evaluate their advantages and disadvantages in terms of security and performance. We do this to develop our firewall management scheme.

In a decentralized topology, all perimeter firewalls are connected to each other forming a fully connected network so that every perimeter firewall can communicate to every other perimeter firewall. The main advantages of a fully connected decentralized system is its availability. As perimeter firewalls in our framework exchange only firewall statistics and do not depend on each other for their basic operations, failure of an individual perimeter firewall does not affect the rest of the perimeter firewalls. Further, the application of decentralized topology to distributed perimeter firewalls has three main disadvantages (a) any compromised perimeter firewall could compromise all other firewalls leading to a serious security violations (b) overhead of information exchange among perimeter firewalls (c) requires more processing power for a perimeter firewall to process the statistics from every other perimeter firewalls. The first disadvantage can be explained with a use case of identification of an unsafe location. In decentralized setup, if a perimeter firewall determines a location L1 as a High Unsafe Location (HUL) then it blocks the traffic from that location and informs this to all other perimeter firewalls. Eventually, every other firewall starts blocking the traffic coming from the location L1. In this case, if a perimeter firewall is compromised and treats a safe location L2 as a HUL and informs the same to all other perimeter firewalls then every perimeter firewall in the network starts blocking the traffic coming from the location L2 even though L2 is a safe location. Thus, any compromised perimeter firewall could compromise the entire system. The other use case is an attacker can use a compromised perimeter firewall to flood every other perimeter firewall of an organization.

In a centralized topology, every perimeter firewall is connected to a central node (a controller). In this set up, unlike decentralized topology, perimeter firewalls could only communicate to a centralized controller and the centralized controller in turn propagates policy

changes to all perimeter firewalls in the network. This approach avoids the drawbacks of the decentralized system. We take the same example of identification of unsafe location to explain how a centralized system overcomes the drawbacks of a decentralized system. Consider a perimeter firewall determines the attack location as HUL on a condition that the attack count from a particular location reaches to a threshold value (*T*). In the centralized scheme, a perimeter firewall reports attack count to the controller at a predefined interval. Suppose '*T*' is a threshold count value and '*n'* is the number of intervals then an interval is computed as $x = T/n$. The perimeter firewall reports to the controller whenever attack count becomes *x, 2x, 3x*, until it reaches *T*. In this case, although an individual perimeter firewall is compromised, it cannot affect the other perimeter firewalls in the network because only the controller has the sole authority to set a location as HUL and enforce the policies at all perimeter firewalls in the network. The other advantage of centralized scheme is relieving the perimeter firewall from processing the firewall statistics and thus making the perimeter firewall cost effective.

3.4 Static vs Dynamic Firewall Policies

The firewall policies in general are predefined and configured by the security administrator based on security requirements of an organization. Although the static firewall policies could prevent known attacks, they could not stop the unknown attacks. This brings the need of dynamic firewall policies. We discuss dynamic firewall policies with a use case of dynamic IP blacklist.

3.4.1 Dynamic IP Blacklist

IP blacklist is a list of blocked IP addresses that are suspected of sending malicious packets. Security administrators maintain a static IP blacklist to block the traffic from certain IP addresses and the list is updated by importing latest blacklists from various security sources,

often vendors. However, there is a problem with the static blacklists. Firewalls could not prevent the attacks coming from IP addresses which are not listed in the blacklists. We developed an algorithm to generate IP blacklists dynamically by identifying the frequency of the attacks originated by an IP address.

Figure 5 illustrates the entire process of generating dynamic IP blacklist. Initially when packet arrives, Source IP address will be read and verified in the final black-list. If there is a match then the packet will be discarded, otherwise packet will be analyzed by the firewall for any malicious content apart from the firewall policies. If any malicious content is found, then the source IP address of the packet is added to the potential blacklist. A separate counter is maintained for each entry of a source IP address in potential blacklist and is incremented whenever a new attack is detected from the same IP address. When the counter value reaches the threshold limit T, IPA will be removed from potential blacklist and moved to final blacklist. Thereafter packets from the IP addresses of final blacklist are completely blocked.

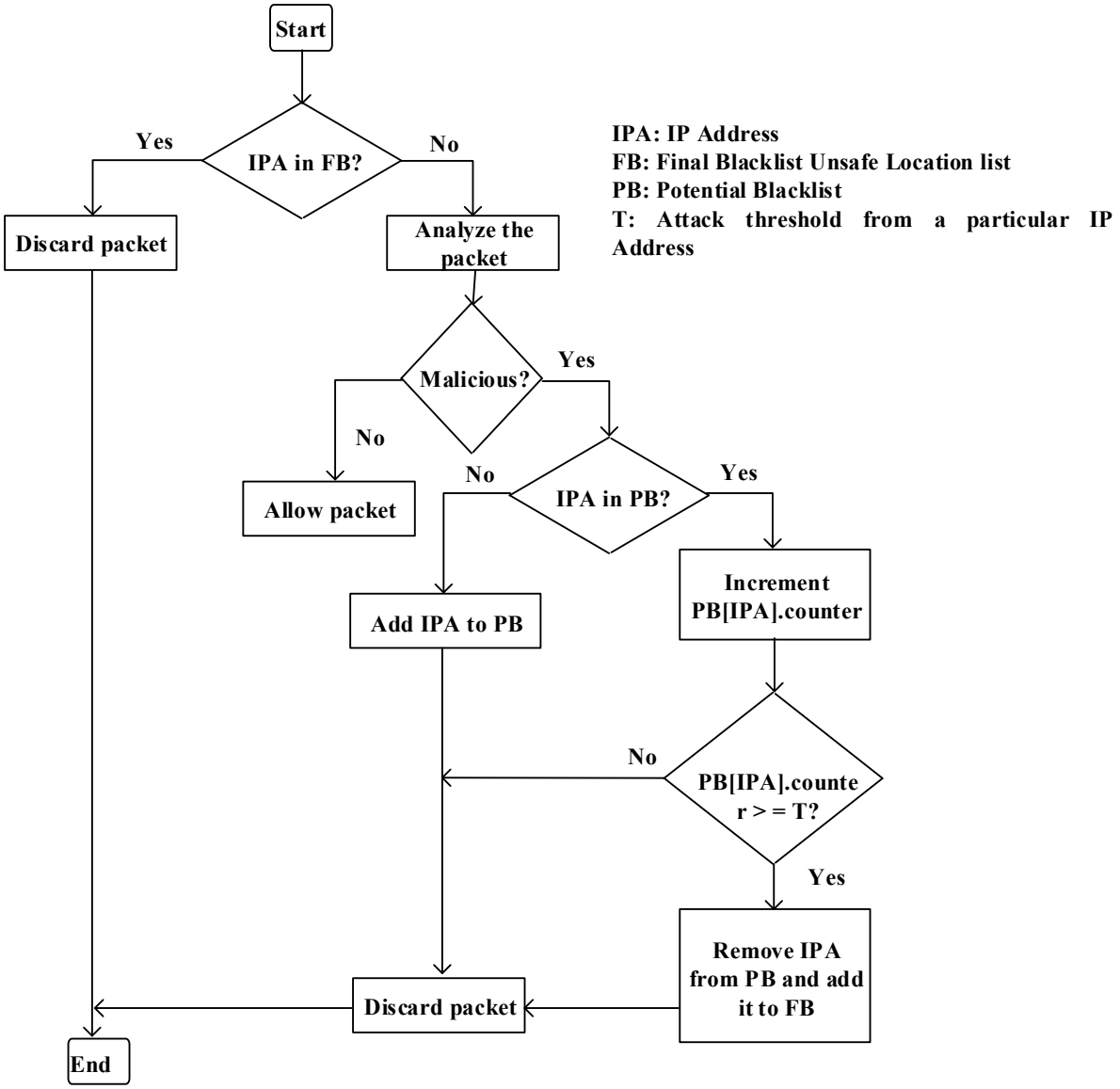

Figure 5. Flowchart of dynamic creation of IP Blacklist

3.5 Global Policy Manager

The core part of our framework is the centralized controller called Global Policy Manager (GPM). In our framework, the GPM is located at the headquarters of a multinational organization and every perimeter firewall is connected to the GPM as shown in Figure 6.

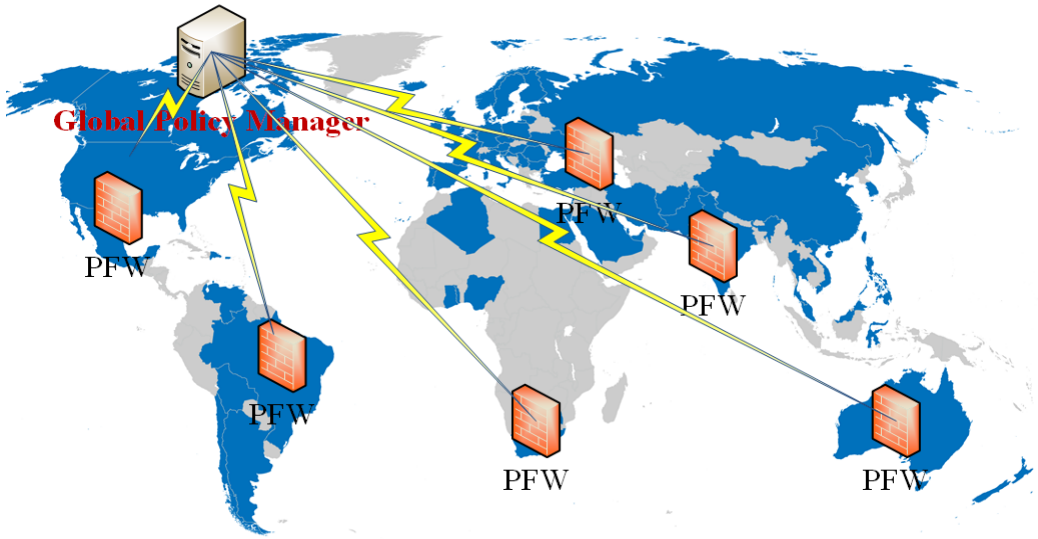

Figure 6. Distributed perimeter firewall setup

3.5.1 Dual Mode

GPM operates in two modes simultaneously - manual and automatic. In manual mode, security administrator can update the firewall policies of any firewall or group of firewalls in the global policy base. In automatic mode, GPM receives statistics or/and policy updates from the individual firewalls and updates the global policy base accordingly. In the above two cases, GPM applies the policy changes to the relevant firewalls if necessary.

3.5.2 Global Policy Manager Architecture

The global policy manager consists of four modules: *policy injection interface, PFW interface, Policy manager* and *Global policy base* as shown in the Figure 7.

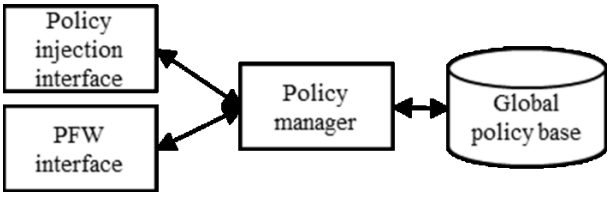

Figure 7. Global policy manager architecture

Policy injection interface: In manual mode, security administrators inject the compliance and group policies into the global policy base. This injection occurs in two cases (a) during initial configuration of the framework and (b) whenever there are changes in compliance and group policies. This interface is made available only to the security administrator who is accountable for security policies of all the branches of an organization.

PFW interface: The perimeter firewall interface that communicates with the perimeter firewalls of all branches of an organization, works in an automatic mode. This interface is used to collect the statistics and policy changes from the perimeter firewalls.

Policy manager: This is the key module which manages all other modules of the GPM. The activities of the policy manager include, (a) analyzing the policies received from the policy injection interface and PFW interface, (b) storing the policies in global policy base, (c) fetching the policies from the global policy base, (d) ensuring compliance, consistency and correctness of the policies and (e) communicating the policy updates to the applicable perimeter firewalls.

Global policy base: The policies of all the perimeter firewalls of an organization are stored in global policy base along with the compliance and group policies.

3.5.3 Policy Configuration Procedures

Policy configuration is required to be performed in two scenarios: Policies configured during initial setup of firewall and when there is any policy update due to change in existing policies.

Initial policy configuration: When a perimeter firewall is setup for the first time, policies should be configured as per the security requirements. The procedure of initial policy configuration is illustrated in seven steps (Figure 8). Every PFW has to define the local policies based on its business unit security requirements. PFW also share location details to the GPM to apply appropriate compliance polices. PFW has to specify the group IDs if it wants to participate in any group. GPM creates a complete list of policies in a precedence order starting with compliance policies followed by group policies and finally adds the local polices as shown in the Figure9. Precedence among different types of policies is *Compliance policies* > *Group policies* > *Local policies*.

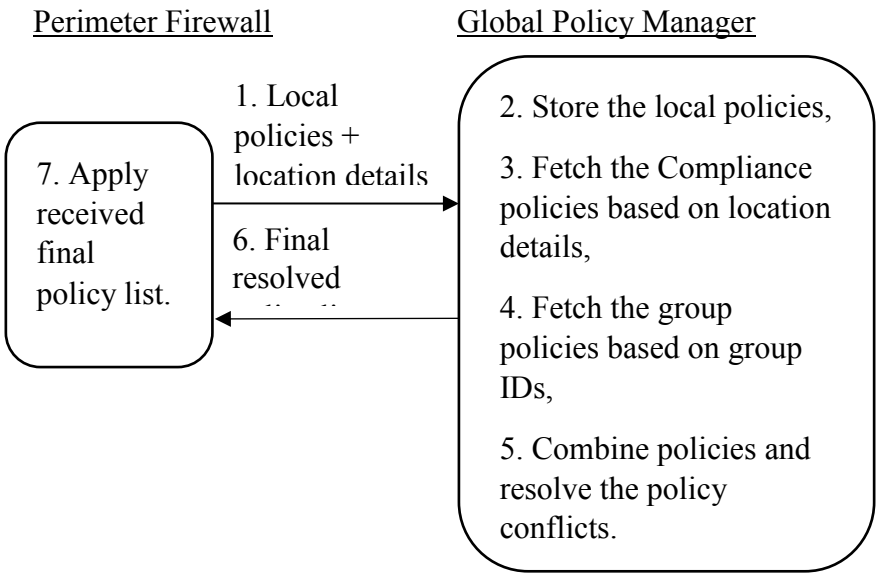

Figure 8. Initial policy configuration

Compliance policies Level 1
Compliance policies Level 2
…
Compliance policies Level n
(group policies 1) U (group policies 2) U … (group policies n)
Local policies

Figure 9. Precedence of policies

Policy update: This framework supports dynamic policy updates. In other words, policies can be dynamically created, modified or removed. This will be done in two ways, (a) GPM initiated update and (b) Peer initiated update.

GPM initiated update: This policy update happens when a policy change decision is made at the organization level and initiated by GPM itself.

Peer initiated update: GPM may propagate updated policies to a set of firewalls. These policies are derived from the statistics or other polices received from firewalls.

3.5.4 Detection of Anomalies in Policy Configuration

The main and critical task of a security administrator is to configure the firewall policies. Due to the complex nature of policy specifications, it often leads to misconfiguration of policies which results in inconsistent and inaccurate policies. There are mainly two types of anomalies that can occur in policy configuration – Redundant and conflict anomalies.

Redundant anomaly: When two policies represent the same set of packets with the same action, then they are said to be in redundant anomaly.

Conflict anomaly: When two policies represent the same set of packets with different actions, then they are said to be in conflict anomaly.

In our framework, we deal with three categories of firewall policies as discussed in section 4.1, compliance policies, group policies and local policies. In this section, we address the anomalies associated with these policies.

To generate a complete list of policies for an individual perimeter firewall, GPM has to combine the compliance, group and local policies. As the policies in different categories may overlap with each other, they result in redundant and conflict anomalies. We identified three possible cases where anomalies can occur in the multi-category policies – inter-category anomalies, intra-category anomalies and inter-group anomalies.

Intra-category anomalies: In this case, the anomalies occur within the same category. For example, policies of local policy category may overlap with each other. This may happen due to the security administrator's inadequate knowledge of policy configuration. The algorithm for the detection of intra-category anomalies is given in the algorithm 1.

Algorithm 1. Intra-Category_Policy_Anomaly_Detection

Input: Policy list P.

Output: TRUE or FALSE, and List of conflict and redundant policies.

RedundantPolicies = []; ConflictPolicies = []; status = FALSE

for *i in 1 to n-1* **do**

    for *j in i+1 to n* **do**

        if *pi.service = pj.service and pi.value R pj.value* **then**

            if pi.action = pj.action **then**

                RedundantPolicies.append(pi,pj);

            else

                ConflictPolicies.append(pi,pj);

            end

            status = TRUE

        end

    end

end

return status

Inter-group anomalies: In this case, the anomalies occur between the policies of two or more groups. For example, if a PFW requires policies of two or more groups then GPM has to check for the anomalies that could occur between the policies of two or more groups. The detection of inter-group anomalies is given in the algorithm 2.

Algorithm 2. Inter-group_Policy_Anomaly_Detection

Input: Policy list G1, G2,..Gn.

Output: TRUE or FALSE, and List of anomaly groups.

anomaly_policies = [ ], status = FALSE

for *i in 1 to n-1* **do**

    for *j in i+1 to n* **do**

        if *Intra-Category_Policy_Anomaly_Detection(Gi+Gj) == TRUE* **then**

            anomaly_groups.append(Gi,Gj)

            status = TRUE

        end

    end

end

return status

Inter-category anomalies: In this case, the anomalies occur between the policies of two or more categories. For example, policies of compliance policy category may overlap with policies of local and/or group category policies and vice-versa. These anomalies are resolved by using precedence of the categories. For example, if there are conflicts between local and group policies, then the conflicts are resolved by retaining group policy and removing local policies. The algorithm for the policy configuration by GPM is given in the algorithm 3.

Algorithm 3. GPM_Policy_configuration

Input: Local Policy list LP, Group IDs Gi and location L.

Output: List of anomalies if any, otherwise complete list of total policies TP

if *Intra-Category_Policy_Anomaly_Detection (LP) == TRUE* **then**

    report anomalies

else if *Inter-group_Policy_Anomaly_Detection (Gi) == TRUE* **then**

    report anomalies

else

    G = G1 + G2 + .. + Gn

    if *Anomalies(LP,G) == TRUE* **then**

        resolve(LP,G)

    elseIf *Anomalies(LP + G,COMP) == TRUE* //COMP – compliance policies **then**

        resolve(LP+G,COMP)

    else

        TP = LP + G + COMP

        SendToPFW(TP)

    end

end

3.6 Implementation and Evaluation

The objective of our implementation is to validate our framework and algorithms. In this validation we have used synthesized datasets of high level firewall policies. We validate

them by measuring the time taken to process the policies and by detecting expected policy anomalies. The validation confirms that (a) our anomaly detection algorithms detect every anomaly correctly and (b) the framework correctly process the policies.

The prototype of our framework is developed in virtual environment. We used Python language to create policy injection interface, perimeter firewall interface and policy manager. We developed communication protocols using Google protocol buffers to exchange policies and statistics between perimeter firewalls and global policy manager. We used ECLiPSe prolog to store and process the global policy base. We created multiple knowledge bases to store information about group of firewalls, global policy list, etc. For our testing, we created our own policy sets that reflect the real time security requirements.

3.6.1 Detection of Policy Anomalies in Local Policies

To validate intra-category policy anomaly detection (algorithm 1), we choose local category policies as a part of initial policy configuration procedure. Anomalies are injected in every dataset of local policies. In this setup, PFW sends local policies, group IDs and location details to the GPM. As there exists anomalies in local policies, GPM reports the PFW as specified in algorithm 3. We calculate the processing time for local policies as,

$$PT_{local} = RTT + GPM_{local}$$

Where RTT is the Round Trip Time of a perimeter firewall to send its local policies and group and location details to GPM and to receive the anomaly information about the policies, GPM_{local} is the time taken by the GPM to process the local policies for checking the anomalies. We observe that RTT remains same for every policy dataset and GPMlocal increases with the increase in number of policies (Figure 10a). Even with the policy set of 500 policies, the response time by the GPM is less than unnoticeable delay of 150ms. We also observe that the

number of anomalies detected by the GPM is equal to the number of injected anomalies (Figure 10b).

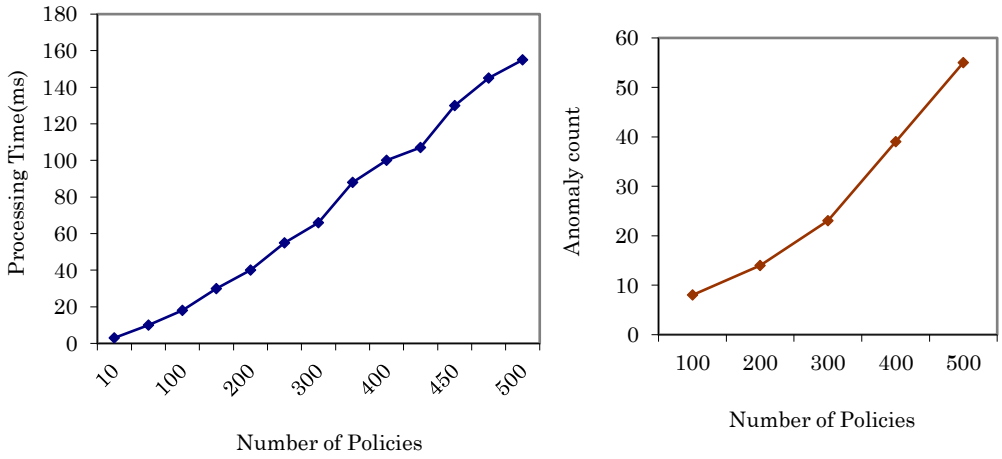

Figure 10. Detection of policy anomalies in local policies (a) Processing time (b) Anomaly count

3.6.2 Detection of policy anomalies in group policies

To validate algorithm 2 which detects inter-group policy anomaly, we chose multiple combination of groups with different number of anomalies. In this setup, PFW sends local policies, group IDs and location details to GPM. As group anomaly detection comes after local policy detection as mentioned in the algorithm 3, we choose anomaly free local policies (count 100) and tested with multiple combination of groups starting from two to six (Figure 11). We calculate the processing time for group policies as $PT_{group} = RTT + GPM_{local} + GPM_{group}$, where PT_{GPM} is the time taken by the GPM to process the group policies for checking the anomalies based on the group IDs requested by a PFW. Here, we observed that *RTT* and GPM_{local} remains constant and GPM_{group} increases with the increase in number of policies. To

check the correctness of anomaly detection, we test the algorithm 2 with all possible combination of groups and plotted only minimum and maximum anomaly combination of the groups. We observe that the difference in anomaly count between minimum and maximum conflict combinations increases with the increase in number of groups.

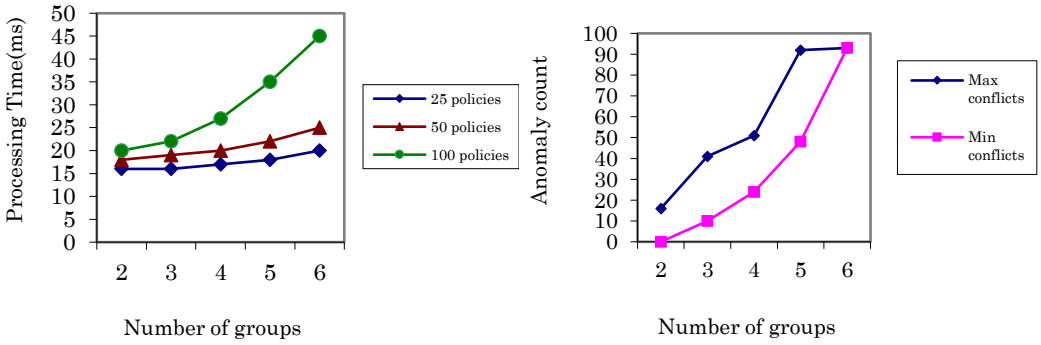

Figure 11. Detection of policy anomalies in multiple group policies (a) Processing time (b) Anomaly count

3.7 Summary

We have presented dynamic firewall policy management framework, which automates the policy management of distributed perimeter firewalls using a central controller called global policy manager. We defined procedures for initial configuration of a perimeter firewall and policy updates. Our main goal of the framework is to ensure consistency and enforce compliance policies across distributed perimeter firewalls of a single multi-national organization.

CHAPTER 4

EFFICIENT DESIGN OF FIREWALL TEMPORAL POLICIES

Firewall policies have been evolved and continue to evolve and expand their scope to address ever-changing security requirements of an organization. One of such new policy is temporal policy. Firewall temporal policies enhance network security by limiting access to certain services or applications during a particular day and time. For example, accessing sensitive information such as patient's records or business critical information can be allowed only during office hours. Temporal policies can also be used to utilize the network bandwidth efficiently by restricting certain less important applications or services such as social networking or video streaming during working hours.

Firewall vendors such as CISCO [10] and Palo Alto [34] have already featured firewall temporal policies in their security products. Temporal policy specification is also available in IP tables [19]. One of the design principles of a firewall is maintaining high throughput and it is also one of the deciding factors to choose a firewall from number of available commercial firewalls. Inclusion of temporal policies in firewall policies results in additional overhead for storing and scanning firewall policies and thus reducing the firewall throughput. While providing the fine control over network traffic, firewall temporal policies may have become bottleneck of the firewall throughput, if not designed efficiently. This necessitates the efficient design of firewall temporal policies. .

In this chapter, we present an innovative and efficient method for representing temporal policies that we have developed in [27], which includes compact representation of temporal policies and detection of anomalies using set operations.

4.1 Firewall Temporal Policies

Firewall temporal policy is a firewall policy that allows or denies a network packet based on specified day and time range of the policy in addition to the packet filtering rules. We consider periodic time (week day and time) parameters to specify temporal policies. In our design, time range of a policy is expressed as start and end time in 24 hour military time format and the day field is expressed as a list of days which is a subset of {Mon, Tue, Wed, Thu, Fri, Sat, Sun, Weekdays, Weekends, Anyday}. The value 'Weekdays' represents {Mon, Tue, Wed, Thu, Fri}, 'Weekends' represents {Sat, Sun} and 'Anyday' represents all the days of the week. For example, a temporal policy can be specified as, "Block the Facebook on weekdays from 0800 to 1700", which means restricting the Facebook access during office working hours on weekdays and only allowing it outside the office hours.

A temporal policy is said to "match" if the packet arrival day and time falls within the specified day and time range respectively. We explain a match using the example policy "Block the Facebook on weekdays from 0800 to 1700." If a Facebook page request arrives to the firewall of the organization on any of the weekdays during the hours 0800 to 1700, an ordered list of policies is searched to find a match. If there is a match, an associated action is performed on the request, which is deny in this example. As it is practiced, a default policy of "deny everything" is listed at the end of policies. Table 5 illustrates some sample policies.

Table 5. Sample temporal policies

P#	Action	Service	Days	Time
1	Deny	Video Streaming	(Monday, Wednesday)	0800-1200
2	Deny	Video Streaming	Any day	Any time
3	Allow	Video Streaming	(Wednesday, Friday)	1200-1500
4	Deny	Facebook	Weekdays	0800-1200
5	Deny	Facebook	Weekdays	1300-1700

4.2 Anomalies in Temporal Policies

A typical organization may have hundreds of firewall policies. When a packet arrives then the entire ordered list of policies is searched for a match and appropriate decision is taken. In cases when a network packet matches more than one policy then only the first match is considered and the rest of the matches are ignored. This approach often leads to erroneous configuration of policies and violates their consistency.

Anomalies are caused due to the misconfiguration of policies. An anomaly exists if two conflicting outcomes are listed in the ordered list of policies. For example, if a Facebook access is requested between 0800 to 1700 and a search of the policy list finds two rules one says block the packet and the other says allow the packet then a policy anomaly is said to occur. We identify two types of anomalies in temporal policies: *conflict* and *redundant.*

Conflict anomaly between two temporal policies *Px* and *Py* occurs when a packet's arrival day *'d'* and time *'t'* match the day and time range of policies *Px* and *Py*, but their decisions are *different*.

Definition 1 (Conflict anomaly): Two temporal policies *Px* and *Py* are said to *conflict* if $d \in \{Px.days \cap Py.days\}$ *and* $(Px.start\_time \leq t \leq Px.end\_time)$ *and* $(Py.start\_time \leq t \leq Py.end\_time)$ *and* $(Px.action \neq Py.action)$.

Redundancy between two temporal policies Px and Py occurs when a packet arrival day *'d'* and time *'t'* matches the day and time range of policies Px and Py whose actions are *same*.

Definition 2 (Redundant anomaly): Two temporal policies Px and Py are said to be *redundant* if $d \in \{Px.days \cap Py.days\}$ and $(Px.start\_time \leq t \leq Px.end\_time)$ and $(Py.start\_time \leq t \leq Py.end\_time)$ and $(Px.action = Py.action)$.

Consider a sample firewall temporal policies from Table 5. Suppose when a video streaming packet arrives on Wednesday at 1300 hours, it matches polices 2 and 3. However, as per first-match rule, action of the policy 2 (deny) is performed on the packet which is different from the policy 3 (allow). In this case policy 2 is said to be in *conflict* with policy 3. Similarly, when a Facebook access is requested on Wednesday at 1100 hours, policies 1 and 2 are matched and as actions of the both policies are same, they are *redundant* to each other. Note that source and destination domains are assumed to be same for all the policies and omitted in the sample policies of Table 5 (Policy number = P#) as the focus is more on temporal policies. Such occurrences are common in manual management of firewalls.

Anomalies in temporal policies can be detected by analyzing the relationship between any two policies with respect to day and time fields. Anomalies between any two policies may occur if any one of the following conditions exists with respect to day and time: *Subset (⊂), Superset (⊃), Equal (=)* and *Overlap (Δ)* and when their packet filtering rules are not disjoint.

Suppose policy *Px* precedes policy *Py* and considering the relationships with respect to the day alone, the conditions are explained below using sample policies of Table 5.

Subset: P1.days ⊂ P2.days

Superset: P2.days ⊃ P3.days

Equal: P4.days = P5.days

Overlap: P1.days Δ P3.days

As the temporal policies are represented in both week day and time, we need to consider all possible combinations of relationships between every two polices with respect to day and time in order to discover anomalies. Here two policies Px and Py are compared only if Px precedes Py in the policy list. Table 6 presents a complete list of all possible combinations of relationships between two policies and anomalies.

Table 6. Comprehensive list of combinations of relationships and Anomalies

Day	**Time**	**Action**	**Anomaly**
Subset	Subset	Same	Redundant
Subset	Equal	Same	Redundant
Subset	Superset	Same	No anomaly
Subset	Overlap	Same	No anomaly
Subset	Subset	Different	Conflict
Subset	Equal	Different	Conflict
Subset	Superset	Different	Conflict

Day	Time	Action	Anomaly
Subset	Overlap	Different	Conflict
Equal	Subset	Same	Redundant
Equal	Equal	Same	Redundant
Equal	Superset	Same	Redundant
Equal	Overlap	Same	No anomaly
Equal	Subset	Different	Conflict
Equal	Equal	Different	Conflict
Equal	Superset	Different	Conflict
Equal	Overlap	Different	Conflict
Superset	Subset	Same	No anomaly
Superset	Equal	Same	Redundant
Superset	Superset	Same	Redundant
Superset	Overlap	Same	No anomaly
Superset	Subset	Different	Conflict
Superset	Equal	Different	Conflict
Superset	Superset	Different	Conflict
Superset	Overlap	Different	Conflict
Overlap	Subset	Same	No anomaly
Overlap	Equal	Same	No anomaly

Day	Time	Action	Anomaly
Overlap	Superset	Same	No anomaly
Overlap	Overlap	Same	No anomaly
Overlap	Subset	Different	Conflict
Overlap	Equal	Different	Conflict
Overlap	Superset	Different	Conflict
Overlap	Overlap	Different	Conflict

Table 7 presents a summary of all anomalies. Although redundant is mentioned as an anomaly, it does not violate policy definitions. However, it is still mentioned as anomaly as it increases the number of policies unnecessarily and thus reducing the performance of processing the firewall rules. The main concern here is the conflict anomaly as it creates the conflict between two policy definitions and hence violates the consistency of the policies.

Table 7. Summary of Anomalies in Temporal Policies

Day	Time	Action	Anomaly
Any*	Any*	Different	Conflict
Subset	Subset/Equal	Same	Redundant
Equal	Subset/Equal/ Superset	Same	Redundant
Superset	Equal/Superset	Same	Redundant

* "Any" means "any relation except disjoint".

4.3 Efficient Representation of Temporal Policies

We optimized the design of temporal firewalls in two phases. In the first phase, we introduce the numeric representation of week days which reduces the storage space used by specification of list of week days and also reduces the time taken in scanning the firewall polices and comparing the list of weekdays to detect anomalies. In second phase, we propose the idea of grouping same day policies into policy sets which results in reduced set of policies. This approach reduces the time taken to match the policies.

4.3.1 Numeric Representation of Temporal Policies

To represent days in temporal policies, additional storage space is required. With the list of days' representation, it consumes significant amount of space and also incurs additional processing time to find a relationship between pair of polices to detect anomalies. To achieve cost-effective representation of days, we introduce the idea of a numeric representation. Here, instead of specifying set of days, a unique numeric value is assigned to every unique subset of week days. The week days are positioned in an order from Monday to Sunday as {Mon, Tue, Wed, Thu, Fri, Sat, Sun}. A binary "1" is assigned to each week day present in the days field of the policy and a binary "0" is assigned to the each week day absent in the days field of the policy. The assignment of binary values 1 and 0 to the week days follows the order from Monday to Sunday to form a 7 bit sequence which is used to calculate the decimal value. This decimal value is used to represent the days field of the policy.

Consider a day field of a policy is {Wednesday, Friday}. A binary 1 is assigned to Wednesday and Friday and a binary zero is assigned to rest of the week days. This generates the representation {0, 0, 1, 0, 1, 0, 0} which forms a 7 bit binary sequence $(0010100)_2$

equivalent of decimal value 20. Table 8 presents some sample conversions from week days to binary form to decimal value.

Table 8. Numeric representation of weekdays

P#	Days	M	T	W	Th	F	S	S	DV*
1	{M, T, W}	1	1	1	0	0	0	0	112
2	{Sat, Sun}	0	0	0	0	0	1	1	3
3	{M, W, F}	1	0	1	0	1	0	0	84
4	{Anyday}	1	1	1	1	1	1	1	127

*DV = Decimal values

By using numerical representation of the day field, temporal policies of Table 5 can be represented as shown in Table 9.

Table 9. Decimal representation of days from Table 5

P#	Action	Service	Days	Time
1	Deny	Video streaming	80	0800-1200
2	Deny	Video streaming	127	Any time
3	Allow	Video streaming	20	1200-1500
4	Deny	Facebook	124	0800-1200
5	Deny	Facebook	124	1300-1700

One of the important task is to check if a packet arrival day is a member of policy's day set. This can be done by performing bitwise AND operation on specific day and policy's day set. If a packet arrival day is '*d*' and policy's day set is *P.days* then membership function is defined as "*If (d∧ P.days) is non-zero then d is a member of P.days else d is not a member of P.days",* where '∧' is a bitwise operation. For example, to check if 'Wed' is a member of policy 1's day's list in the Table 8, we have to perform binary ∧ operation between value of Wed and value of day's list of policy 1. Here value of the Wed is 16 and value of day's list of policy 1 is 112. Bitwise operation ∧ of 16 and 112 yields non-zero value. So Wed is a member of day's list of policy 1.

The possible relationships between each pair of policies with respect to weekdays are: *subset, superset, overlap, equal and disjoint.*

4.3.2 Algorithm to Find Day Relationship between a Pair of Policies

As set of week days are represented using decimal value, finding relationships between pair of policies is not straightforward. We have taken the advantage of set operations to find the relationship between two sets of week days.

In the algorithm, *Px.days* and *Py.days* are the decimal values of days set of policies *Px* and *Py* respectively. "AND" is a bitwise AND operation which is an equivalent to set intersection operator.

Algorithm *Relationship (Px.days, Py.days)*

Input: Px.days, Py.days

Output: Relation

Begin

If (Px.days == Py.days) **then**

Relation = “EQUAL”

Else if (Px.days AND Py.days) == 0 **then**

Relation = “DISJOINT”

Else if (Px.days AND Py.days) == Px.days **then**

Relation = “SUBSET”

Else if (Px.days AND Py.days) == Py.days **then**

Relation = “SUPERSET”

Else

Relation = “OVERLAP”

End if

Return Relation

End

4.3.3 Correctness of the Relationship algorithm

To prove the correctness of the relationship algorithm, we have to prove that the corresponding relationship conditions are correct.

Equal: The condition for equal relation does not need proof as the condition (Px.days == Py.days) is a direct comparison of two decimal values for equivalence.

Disjoint: we need to prove that Px.days and Py.days are disjoint if (Px.days AND Py.days) == 0. Assume that Px.days and Py.days are not disjoint. Then, there should be an element x in both sets Px.days and Py.days. Since x is in both sets Px.days and Py.days, (Px.days AND Py.days) will be not be 0 which is a contradiction to our condition. So, Px.days and Py.days are disjoint.

Subset: we need to prove that Px.days is a subset of Py.days if (Px.days AND Py.days) == Px.days. Assume that Px.days is not a subset of Py.days. Then, there is an element x in Px.days that is not in Py.days. Since x is not in Py.days, (Px.days AND Py.days) will not include x. Thus, (Px.days AND Py.days) ≠ Px.days. But our condition is (Px.days AND Py.days) == Px.days, which is a contradiction. So Px.days is a subset of Py.days.

Superset: The proof of superset condition is similar to the subset condition.

Overlap: If all the above conditions are false, then only the left over possible relation is overlap. Once the relationship is determined, Table 7 is used to detect the possible anomaly and same to be reported.

4.3.4 Policy Sets based on Week Day

The design of temporal policies can be optimized by filtering out the policies corresponding to each weekday and group them as a policy set. In this way, we can have a separate policy sets for each weekday. As we have seven weekdays (Monday, Tuesday,

Wednesday, Thursday, Friday, Saturday, Sunday), we get seven policy sets. Every day, corresponding policy set is chosen and applied to the firewall. This reduces the processing time to process all the firewall policies. With this optimization, only time has to be verified. The firewall system can apply the corresponding policy set every day at time 0000 hrs. and the policy set is valid till the time 2359 hrs. (Time 2359 hours represent the time between 23 hours 59 minutes 00 seconds and 23 hours 59 minutes 59 seconds).

Consider the sample policies from Table 5, policy set for Friday consists of policies P2, P3, P4 and P5, whereas policy set for Saturday consists of only one policy P2. In later case, when a packet arrives on Saturday, only one policy will be scanned, though there are five policies in the policy list.

4.4 Implementation

We used prolog based logic programming language ECLiPSe [44] to represent our optimized design of temporal policies. ECLiPSe is an open source platform for developing constraint-logic-programming applications. Declarative nature of logic programming makes it easy to specify the temporal policy rules. As logic programs are used to describe relations, it is a better choice to represent and analyze relations of temporal policies.

There are three basic constructs in Prolog: facts, rules, and queries. Facts and rules are used to create knowledge bases. We represented temporal policies as facts, relations and anomalies as rules and finally we used queries to detect the anomalies.

In our implementation, a temporal policy, which could be represented as a fact is represented by *pTime(policyNo,policyName,days,start_time, end_time, action),* where *policyNo* is policy number in an increasing order, *policyName* is the name of a policy, *days* is the decimal value of week days*, start_time* and *end_time* indicate the time range to apply the

policy and finally *action* is a binary decision of allow or deny. For instance, policy 1 in Table 9 is represented as *pTime(1, deny_video_1, 80, 800, 1200, deny).* Note that *policyName* is not mentioned in the table.

The day relation predicate *dayRel* is used to determine the relation between two policies with respect to week days. Px and Py are the policy names of policy x and policy y, Nx and Ny are the policy numbers, DaysX and DaysY are the days fields. The prolog statements for *dayRel* predicate are as follows,

Equal: *dayRel(equal,Px,Py):- pTime(Nx,Px,DaysX,_,_,_), pTime(Ny,Py,DaysY,_,_,_),Nx < Ny, Px \== Py, (DaysX =:= DaysY -> true;false).*

Subset: *dayRel(subset,Px,Py):- pTime(Nx,Px,DaysX,_,_,_), pTime(Ny,Py,DaysY,_,_,_), Nx < Ny, Px \== Py, DaysX =\= DaysY, ((DaysX /\ DaysY) =:= DaysX -> true;false).*

Superset: *dayRel(superset,Px,Py):- pTime(Nx,Px,DaysX,_,_,_), pTime(Ny,Py,DaysY,_,_,_), Nx < Ny, Px \== Py, DaysX =\= DaysY, ((DaysX /\ DaysY) =:= DaysY -> true;false).*

Disjoint: *dayRel(disjoint,Px,Py):- pTime(Nx,Px,DaysX,_,_,_), pTime(Ny,Py,DaysY,_,_,_), Nx < Ny, Px \== Py, ((DaysX /\ DaysY) =:= 0 -> true;false).*

Time relation predicate *timeRel* is used to determine the relation between two policies with respect to time. StartX and EndX are the starting time and ending time of the policy x and StartY and EndY are the starting time and ending time of policy y. The prolog statements for *timeRel* predicate are as follows,

Equal: *timeRel(equal,Px,Py):- pTime(Nx,Px,_,StartX,EndX,_), pTime(Ny,Py,_,StartY,EndY,_), Nx < Ny, Px \== Py, ((StartX =:= StartY, EndX =:= EndY) -> true;false).*

Subset: *timeRel(subset,Px,Py):- pTime(Nx,Px,_,StartX,EndX, _), pTime(Ny,Py,_,StartY,EndY,_), Nx < Ny, Px \== Py, (((StartX >= StartY, EndX < EndY); (StartX > StartY, EndX =< EndY)) -> true;false).*

Superset: *timeRel(superset,Px,Py):- pTime(Nx,Px,_,StartX,EndX,_),pTime(Ny,Py,_,StartY,EndY, _), Nx < Ny, Px \== Py, (((StartX =< StartY, EndX > EndY); (StartX < StartY, EndX >= EndY)) -> true;false).*

Disjoint: *timeRel(disjoint,Px,Py):-pTime(Nx,Px,_,StartX, EndX,_),pTime(Ny,Py,_,StartY,EndY, _), Nx < Ny, Px \== Py, ((StartX >= EndY ; EndX =< StartY) -> true;false).*

Anomaly predicate *anomaly* is used to find redundant and conflict anomalies between every two policies. The prolog statements for *anomaly* predicate are as follows,

Redundant Anomaly Predicates:

anomaly(redundant,Px,Py):- dayRel(subset,Px,Py), timeRel(TimeRel,Px,Py), (TimeRel = subset; TimeRel = equal), pTime(_,Px,_,_,_,ActionX), pTime(_,Py,_,_,_, ActionY), ActionX = ActionY.

anomaly(redundant,Px,Py):- dayRel(equal,Px,Py), timeRel(TimeRel,Px,Py), (TimeRel = subset ; TimeRel =equal;TimeRel=superset),pTime(_,Px,_,_,_,ActionX),pTime(_,Py,_,_,_, ActionY), ActionX = ActionY.

anomaly(redundant,Px,Py) :- dayRel(superset,Px,Py), timeRel(TimeRel,Px,Py), (TimeRel = equal ; TimeRel =superset),pTime(_,Px,_,_,_,ActionX),pTime(_,Py,_,_,_, ActionY), ActionX = ActionY.

Conflict Anomaly Predicate:

anomaly(conflict,Px,Py):- dayRel(DayRel,Px,Py), timeRel(TimeRel,Px,Py), DayRel \= disjoint, TimeRel \=disjoint,pTime(_,Px,_,_,_,ActionX),pTime(_,Py,_,_,_, ActionY), ActionX \= ActionY.

To find out the anomalies, we have to query the above statements with, *findall((X,Px,Py),anomaly(X,Px,Py), Anomalies)*.

We have not included the type of service in our implementation as it is assumed to be same or related for all the policies. We have tested our implementation by considering synthesized temporal policies. The policies are chosen in such a way that all the day and time relations are covered as specified in Table 6. Our implementation is able to find all the anomalies associated with the given sample policies.

4.5 Summary

Although firewall temporal policies provide fine control to security administrators over network traffic, they affect the firewall performance in terms of additional storage and processing time. To lessen the impact of temporal policies on overall performance of firewall, we choose to represent week days in numerical representation which requires only one byte to store week days of a temporal policy, thus optimizing the storage requirements of temporal policies. However, this optimization throws a challenge of detecting policy conflicts in numerical representation. To solve this problem, we used set relations and identified possible relations that cause conflicting policies.

The second optimization technique in our design is to reduce the policy scanning time. Firewall filters network packets by comparing the packet contents with firewall policies sequentially from top to bottom. Scanning entire policy list takes considerable amount of time.

To improve the scanning time without losing policy consistency, we used the method of splitting the entire policy list into smaller policy lists based on week day so that the firewall would use smaller policy list, which has policies defined for that particular day. With these optimizations, the impact of firewall temporal policies on overall performance of firewall could be reduced.

CHAPTER 5

IDENTIFICATION OF UNSAFE LOCATIONS IN IP AND CELLULAR BASED NETWORKS

Attacks on an organization's network come from many locations (local or international). We can categorize these locations as (a) unsafe and (b) safe. A location from where attacks frequently originate is defined as an unsafe location. In reality, there is no ideal safe location. We differentiate between locations from where a large number of serious attacks occur and locations from where relatively less number of attacks (less serious and less frequently). One of our objectives of our dissertation is to identify these locations (static or varying) and develop a database of unsafe locations that can be used to detect and protect an organization from these attacks by reconfiguring the firewalls dynamically as necessary.

We categorize network traffic origination locations as (a) Hard Unsafe Location, (b) Mild Unsafe Locations, and (c) Clean Location.

Hard Unsafe Location: A location from where a large number of serious attacks originate with a high frequency. Any of these attacks can severely affect the security and integrity of a system and recovery may be time consuming and costly. It is quite possible that less serious attacks may originate from serious locations, but they are treated as serious attacks. If the firewall detects that an attack is mounted from a serious location then it immediately eliminates this attack. For example, a Trojan is a serious attack and it is immediately eliminated by the firewall.

Mild Unsafe Location: A location from where relatively less number of less serious attacks originate. These types of attacks do not significantly affect the system performance and integrity and the system can continue to function while the attack is taken care of. For example, some mild virus can just scare people without harming the computer. This type of virus may be found on music sharing platform where the virus just obstruct music sharing with suspicious users. The firewall may decide to let it enter the system.

Clean Location: A location from where no attack originates. The firewall may apply minimal security checks to messages coming from these locations.

There is a good amount of research dedicated to geo-locate the origination of attack based on IP addresses. However, the dynamic nature of allocating IP addresses and the vulnerabilities of spoofing IP addresses make it unfavorable as a geo-location identifier. IPs assignment is region specific and a set of IPs can identify a location with some granularity. If an attacker moves around in a location while attacking an installation then the unit will have the same IP address within different points inside the location. When an attacker moves from location *li* to location *lj* then the IP address will also change. However, the IP address will remain the same if the movement is confined to *li*. The later situation is more difficult to handle algorithmically alone. For this, the organization that wants to protect its computers must set a set of IP assignment policies.

An attacker can mount an attack from its current location. To hide the identity and to protect itself being caught, the attacker generally use a proxy. For example, an attacker located at a location u may use a gateway located at l as a proxy (Figure 12). The TRACERT (Trace Route) command is not helpful because it cannot go beyond the proxy. Therefore, finding the origin of an attack by tracing an IP address is not possible.

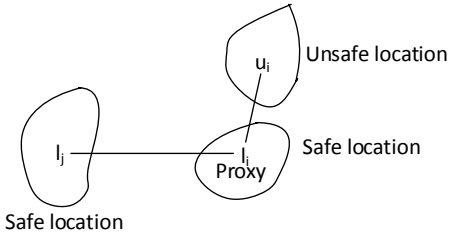

Figure 12. Hiding unsafe location behind the proxy

Currently, the security measures taken by the organizations with respect to mobile apps use authentication and encryption. Although authentication and encryption solve security issues to some extent, they still have some limitations. For instance, an attacker can easily mount an attack on data centers with stolen credentials. Firewall at the data center allows this attack to happen as the request comes from the authenticated user and there is no way of knowing who the attacker is. Our approach which is presented in [25, 26] will identify the geo-location of the attacker even though the credentials are compromised.

5.1 Our Approach

Allocation of IP addresses is dynamic and not bound to any specific location. Also IP addresses can be spoofed in internet communication. Using this an attacker can exploit IP addresses to hide his/her identity which makes it hard to identify the attacker's location based on IP address. In order to reach to the attacker hiding behind the proxy, we propose the following approach. The location of a mobile phone in a cellular network can be identified by knowing its *cell global identity*. The scope of this approach is limited to the mobile devices that uses cellular network.

The location of a mobile device in a cellular network is given by *cell global identity*. For example, if the *cell global identity* in an IP packet is MCC = 310, MNC = 410, LAC =

3450 and CI = 118541125, then it represents a Cell in Kansas City of Missouri in United State of America. Here MCC (310) represents the country United States of America, MNC (410) represents AT&T network and LAC and CI codes represents a unique cell area in United States of America (Figure 13).

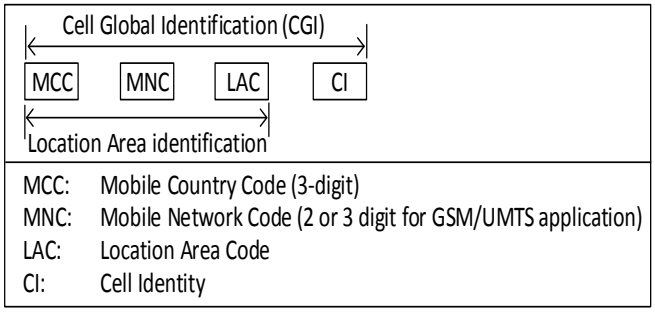

Figure 13. Cell global identity structure

5.2 Extended IP Header

At present cell global identity information is not available in IP packets coming from mobile devices. Our proposal is to extend the structure of an IP packet and include cell global identity information in IP packets. IP packet header contains optional field. It can be used to store the *cell global identity*. This will help us to identify the location of the mobile unit mounting the attack; directly or through a proxy. This will help us to program the firewall accordingly, which then can block the attack from an unsafe location.

The IP Header format has an option field that is used whenever it is necessary (Figure 14). As per RFC791 standard, option field is of variable length and two types of formats are available: (a) a single octet of option type and (b) an option-type octet, an option-length octet, and the

actual option-data octets. In this proposal, we consider second type of option field format with one octet of type, one octet of length and 6 octets of data which forms the *cell global identity.*

<table>
<tr><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td>Time to Live</td><td colspan="2">Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
<tr><td>Type</td><td colspan="2">Length</td><td colspan="2">MCC</td></tr>
<tr><td colspan="3">MNC</td><td colspan="2">Location Area Code</td></tr>
<tr><td colspan="5">Cell Identity</td></tr>
</table>

Figure 14. Extended IP Header

As per RFC 971 standard, the option-type octet is viewed as having 3 fields: 1 bit-copied flag, 2 bits-option class and 5 bits-option number. In type field of proposed option header has the values of 1 as flag, 1 as class and 1 as the number. So, Option type = 10100001, i.e. 161. MCC is Mobile Country Code which is of 2 octets, MNC is Mobile Network Code which is of 2 octets, LAC is Location Area Code which is of 2 octets and Cell Identity is of 4 octets.

5.3 Firewall Logic

Initially when packet arrives, location area code will be read and verified in the HUL (Hard Unsafe Location) list. If there is a match then the packet will be discarded, otherwise packet will be analyzed by firewall for any malicious content apart from the firewall policies. If any malicious content is found, then LAC of the packet is recorded in MUL (Mild Unsafe Location) list. A separate counter is maintained for each entry of LAC in MUL and will be incremented whenever a new attack is detected from the same location.

When the counter value reaches the threshold limit T, it will be removed from MUL list and moved to HUL list. Thereafter packets from these HUL are completely blocked. Note that HUL is the final list of unsafe locations which we want to avoid.

Figure 15 illustrates the entire process of determining unsafe locations. It is an ongoing process of finding unsafe locations based on number of attacks originating from a specific location.

LAC is a Location Area Code from where packet originates and

T is Threshold limit of attacks from a particular location.

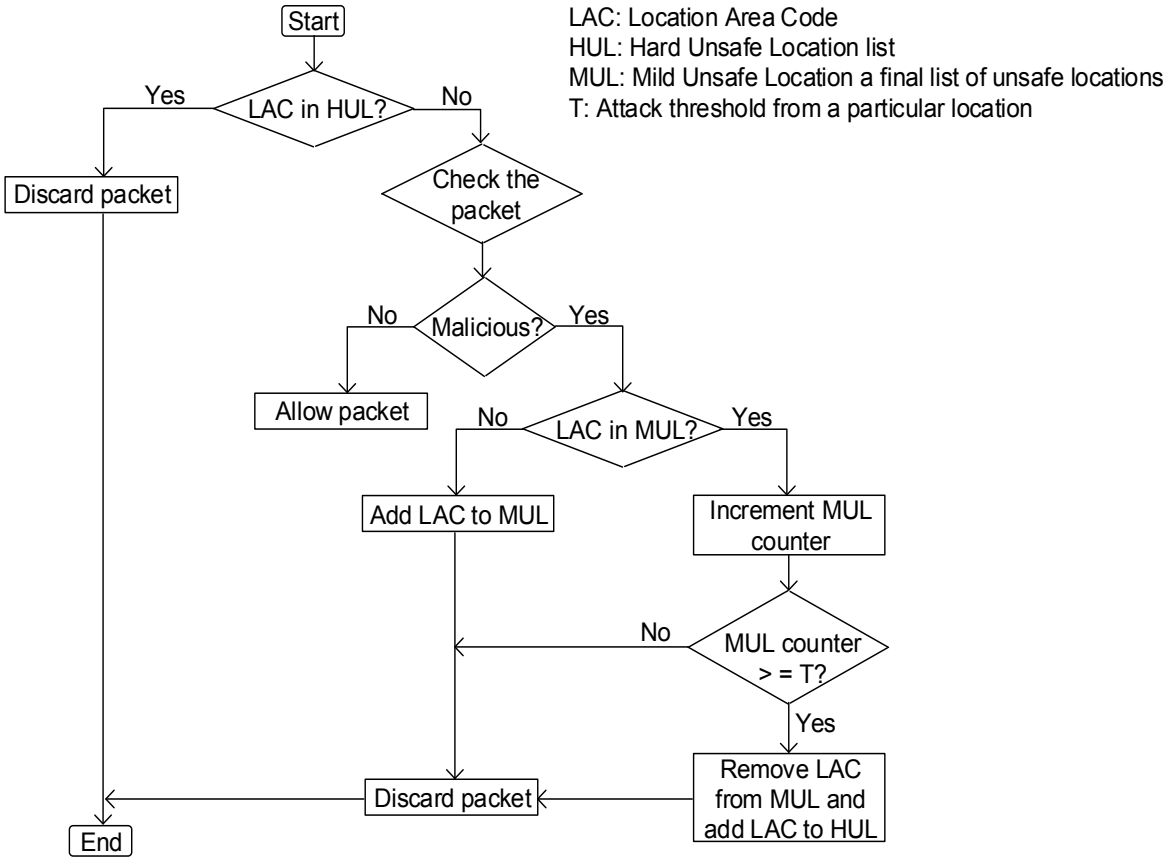

Figure 15. Firewall logic flowchart

How to geo-locate the attacker? Once an attack is identified by the Firewall then attacker's geolocation can be identified by converting the cell global identity to the GPS coordinates. Google provides the APIs to convert Cell Global Identity information the GPS coordinates. It can also be mapped on the Google Maps.

5.4 Implementation and Evaluation

The Client at Mobile side is responsible for integrating *cell global identity* information with the IP packet. This information should be encrypted to safeguard the privacy of the user and to protect from the man-in-the-middle attack (Figure 16). Encryption technique is incorporated

in the mobile client implementation logic. The Cell Global Identity information should be made available in every packet that is going to the organization.

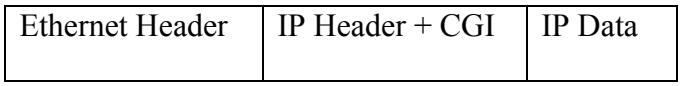

Figure 16. Integration of CGI with IP Header

.

Another approach is to make the *cell global identity* available in the IP packets only at the time of authentication of user by firewall that is while mobile client sending credentials to the firewall in the authentication process. This reduces the size of the total length of the packets sent to the firewall compared to the previous approach. Also on the other side, firewall does not have to check and compare the Location Area Code for every packet. This reduces the load on the Firewall. However, there is a disadvantage to this approach. When user moves from one location to another or one location area to another location area then this new location area information is not made available to the firewall until user logins again. There is an inconsistency between user's Location Area known to the Firewall and actual user's Location Area. So, by making Cell Global Identity available in every outgoing packet reduces these inconsistencies and avoids any loopholes that can be exploited by the attacker.

We evaluated our approach by developing mobile app, which acts as mobile client and implemented the firewall logic in python, which acts as a Firewall. We loaded mobile app on various android devices and tried accessing server that has our Firewall program running on that server. Firewall program able to extract the IP packets, reads the Cell Global Identity information of various locations, and compared against the existing list of Location Area Codes. To categorize a location *li* as an unsafe location, firewall should be able to detect

malicious content in the packets originating from the location *li*. We used port scanning activity as a malicious activity and tested our algorithm by

CHAPTER 6

CONCLUSION AND FUTURE WORK

Policy management is one of the main challenges of distributed firewalls. Manual configuration of policies often leads to policy inconsistencies and non-compliance issues. To address these problems, we introduced hierarchical compliance based policies to define organization wide compliance policies and geo-location wide compliance policies. We identified the requirement of various categories of policies in order to ensure consistency among distributed firewalls. As policy conflicts are inevitable in network access control policies, we developed algorithms to detect policy conflict within intra-category and inter-category policies. We also designed efficient representation of firewall temporal policies and developed an algorithm to detect temporal policy conflicts.

In our future work, we would like to extend our framework to Internet of Things and we planned to use machine-learning algorithms to identify the attack patterns based on network packet statistics.

REFERENCES

[1] Alfaro, Joaquin G., Frederic Cuppens, and Nora Cuppens-Boulahia. "Aggregating and deploying network access control policies." In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pp. 532-542. IEEE, 2007.

[2] Alfaro, Joaquin Garcia, Nora Boulahia-Cuppens, and Frédéric Cuppens. "Complete analysis of configuration rules to guarantee reliable network security policies." International Journal of Information Security 7, no. 2 (2008): 103-122.

[3] Al-Shaer, Ehab, Hazem Hamed, Raouf Boutaba, and Masum Hasan. "Conflict classification and analysis of distributed firewall policies." IEEE journal on selected areas in communications 23, no. 10 (2005): 2069-2084.

[4] Al-Shaer, Ehab S., and Hazem H. Hamed. "Discovery of policy anomalies in distributed firewalls." In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, pp. 2605-2616. IEEE, 2004.

[5] Bandara, Arosha K., Antonis C. Kakas, Emil C. Lupu, and Alessandra Russo. "Using argumentation logic for firewall configuration management." In *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on*, pp. 180-187. IEEE, 2009.

[6] Bartal, Yair, Alain Mayer, Kobbi Nissim, and Avishai Wool. "Firmato: A novel firewall management toolkit." In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pp. 17-31. IEEE, 1999.

[7] Bellovin, Steven M. "Distributed firewalls." *Journal of Login* 24, no. 5 (1999): 37-39.

[8] Bhatti, Rafae, Maria Luisa Damiani, David W. Bettis, and Elisa Bertino. "Policy mapper: Administering location-based access-control policies." *IEEE internet Computing* 12, no. 2 (2008).

[9] Capretta, Venanzio, Bernard Stepien, Amy Felty, and Stan Matwin. "Formal correctness of conflict detection for firewalls." In *Proceedings of the 2007 ACM workshop on Formal methods in security engineering*, pp. 22-30. ACM, 2007.

[10] CISCO. "Creating and Applying Group Policies," accessed April 4, 2016, https://documentation.meraki.com/MX-Z/Group_Policies_and_Blacklisting/Creating_and_Applying_Group_Policies.

[11] Cremonini, Marco, Ernesto Damiani, and Pierangela Samarati. "Semantics-aware perimeter protection." In *Data and Applications Security XVII*, pp. 229-242. Springer US, 2004.

[12] Eronen, Pasi, and Jukka Zitting. "An expert system for analyzing firewall rules." In *Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec 2001)*, pp. 100-107. 2001.

[13] Gangadharan, Muralidaran and Kai Hwang. "Intranet security with micro-firewalls and mobile agents for proactive intrusion response." In *Computer Networks and Mobile Computing, 2001. Proceedings. 2001 International Conference on*, pp. 325-332. IEEE, 2001.

[14] Gouda, Mohamed G., and X-YA Liu. "Firewall design: Consistency, completeness, and compactness." In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pp. 320-327. IEEE, 2004.

[15] Gouda, Mohamed G., Alex X. Liu, and Mansoor Jafry. "Verification of distributed firewalls." In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pp. 1-5. IEEE, 2008.

[16] Gupta, Pankaj, and Nick McKeown. "Algorithms for packet classification." *IEEE Network* 15, no. 2 (2001): 24-32.

[17] Gupta, Pankaj, and Nick McKeown. "Packet classification on multiple fields." *ACM SIGCOMM Computer Communication Review* 29, no. 4 (1999): 147-160.

[18] Guttman, Joshua D. "Filtering postures: Local enforcement for global policies." In *Security and Privacy, 1997. Proceedings, 1997 IEEE Symposium on*, pp. 120-129. IEEE, 1997.

[19] Herve, Eychenne. "iptables (8) - Linux man page," accessed April 4, 2016, http://linux.die.net/man/8/iptables. [Accessed: April 4, 2016].

[20] Hamed, Hazem, and Ehab Al-Shaer. "Taxonomy of conflicts in network security policies." *IEEE Communications Magazine* 44, no. 3 (2006): 134-141.

[21] Hari, Adiseshu, Subhash Suri, and Guru Parulkar. "Detecting and resolving packet filter conflicts." In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1203-1212. IEEE, 2000.

[22] Hazelhurst, Scott, Adi Attar, and Raymond Sinnappan. "Algorithms for improving the dependability of firewall and filter rule lists." In *Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on*, pp. 576-585. IEEE, 2000.

[23] Hu, Hongxin, Wonkyu Han, Gail-Joon Ahn, and Ziming Zhao. "FLOWGUARD: building robust firewalls for software-defined networks." In *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 97-102. ACM, 2014.

[24] Ioannidis, Sotiris, Angelos D. Keromytis, Steve M. Bellovin, and Jonathan M. Smith. "Implementing a distributed firewall." In *Proceedings of the 7th ACM conference on Computer and communications security*, pp. 190-199. ACM, 2000.

[25] Jaiswal, Chetan, Mahesh Nath, and Vijay Kumar. "Location-Based Security Framework for Cloud Perimeters." *IEEE Cloud Computing* 1, no. 3 (2014): 56-64.

[26] Maddumala, Mahesh Nath, and Vijay Kumar. "A Logic-Based Security Framework for Mobile Perimeter." In *Mobile Data Management (MDM), 2015 16th IEEE International Conference on*, vol. 2, pp. 30-33. IEEE, 2015.

[27] Maddumala, Mahesh Nath, and Vijay Kumar. "Efficient Design of Firewall Temporal Policies." In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 2, pp. 449-454. IEEE, 2016.

[28] Martínez, Salvador, Joaquin Garcia-Alfaro, Frédéric Cuppens, Nora Cuppens-Boulahia, and Jordi Cabot. "Model-driven extraction and analysis of network security policies." In *International Conference on Model Driven Engineering Languages and Systems*, pp. 52-68. Springer, Berlin, Heidelberg, 2013.

[29] Mayer, Alain, Avishai Wool, and Elisha Ziskind. "Fang: A firewall analysis engine." In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pp. 177-187. IEEE, 2000.

[30] Mayer, Alain, Avishai Wool, and Elisha Ziskind. "Offline firewall analysis." *International Journal of Information Security* 5, no. 3 (2006): 125-144.

[31] Meredith, Lynn M. "A summary of the autonomic distributed firewalls (ADF) project." In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, vol. 2, pp. 260-265. IEEE, 2003.

[32] Mouelhi, Tejeddine, Franck Fleurey, Benoit Baudry, and Yves Le Traon. "A model-based framework for security policy specification, deployment and testing." *Model Driven Engineering Languages and Systems* (2008): 537-552.

[33] Nelson, Timothy, Christopher Barratt, Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. "The Margrave Tool for Firewall Analysis." In *LISA*, pp. 1-18. 2010.

[34] nrice. "How to Schedule Policy Actions," accessed April 4, 2016, https://live.paloaltonetworks.com/t5/Management-Articles/How-to-Schedule-Policy-Actions/ta-p/56338.

[35] Payne, Charles, and Tom Markham. "Architecture and applications for a distributed embedded firewall." In *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, pp. 329-336. IEEE, 2001.

[36] Pescatore, John, and Greg Young. "Defining the Next-Generation Firewall", *Gartner RAS Core Research Note G00171540*, 12 October 2009, R3210 04102010.

[37] Singh, Sumeet, Florin Baboescu, George Varghese, and Jia Wang. "Packet classification using multidimensional cutting." In *Proceedings of the 2003 conference*

on Applications, technologies, architectures, and protocols for computer communications, pp. 213-224. ACM, 2003.

[38] Song, Haoyu, and John W. Lockwood. "Efficient packet classification for network intrusion detection using FPGA." In *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, pp. 238-245. ACM, 2005.

[39] Srinivasan, Venkatachary, Subhash Suri, and George Varghese. "Packet classification using tuple space search." In *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 4, pp. 135-146. ACM, 1999.

[40] Taylor, David E., and Jonathan S. Turner. "Classbench: A packet classification benchmark." *IEEE/ACM Transactions on Networking (TON)* 15, no. 3 (2007): 499-511.

[41] Thanasegaran, Subana, Yi Yin, Yuichiro Tateiwa, Yoshiaki Katayama, and Naohisa Takahashi. "A topological approach to detect conflicts in firewall policies." In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1-7. IEEE, 2009.

[42] Thanasegaran, Subana, Yuichiro Tateiwa, Yoshiaki Katayama, and Naohisa Takahashi. "An improved conflict detection system with periodic cycle treatment for time-based firewall policies." In *2010* Proceedings *of 19th International Conference on Computer Communications and Networks*. 2010.

[43] Thanasegaran, Subana, Yuichiro Tateiwa, Yoshiaki Katayama, and Naohisa Takahashi. "Simultaneous Analysis of Time and Space for Conflict Detection in Time-

Based Firewall Policies." In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pp. 1015-1021. IEEE, 2010.

[44] "The ECLiPSe Constraint Programming System," [Online]. Available: http://eclipseclp.org/. [Accessed: April 4, 2016].

[45] Tongaonkar, Alok, Niranjan Inamdar, and R. Sekar. "Inferring Higher Level Policies from Firewall Rules." In *LISA*, vol. 7, pp. 1-10. 2007.

[46] Vamanan, Balajee, Gwendolyn Voskuilen, and T. N. Vijaykumar. "EffiCuts: optimizing packet classification for memory and throughput." In *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 207-218. ACM, 2010.

[47] Wikipedia. "(2017) Internet Censorship in China," accessed Feb 28, 2017, https://en.wikipedia.org/wiki/Internet_censorship_in_China.

[48] Wikipedia. "(2017) Internet Censorship in Iran," accessed Feb 28, 2017, https://en.wikipedia.org/wiki/Internet_censorship_in_Iran.

[49] Yuan, Lihua, Hao Chen, Jianning Mai, Chen-Nee Chuah, Zhendong Su, and Prasant Mohapatra. "Fireman: A toolkit for firewall modeling and analysis." In *Security and Privacy, 2006 IEEE Symposium on*, pp. 15-pp. IEEE, 2006.

[50] Zhang, Bin, Ehab Al-Shaer, Radha Jagadeesan, James Riely, and Corin Pitcher. "Specifications of a high-level conflict-free firewall policy language for multi-domain networks." In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pp. 185-194. ACM, 2007.

VITA

Mahesh Nath Maddumala received an M.Tech. in Computer Science and Technology from Andhra University, Visakhapatnam, India in 2007 and a B.Tech. in Computer Science and Information Technology from M. L. Eng. College, India in 2004.. He worked as a senior software engineer at Aricent, Bangalore and USA from 2007 to 2012 and later joined S. V. P. Eng. College as an Assistant Professor in 2013. He joined the PhD program at UMKC full-time in 2014. His research areas are in the area of Computer and Network Security, Ethical Hacking, Digital Forensics and Malware Reverse Engineering.