Detection of Firewall Policy Anomalies in Real-time Distributed Network Security Appliances

Ameya Hanamsagar
Department of Computer Engineering, Sinhgad Institute of Technology & Science, Narhe, Pune 411041

Bhagyashree Borate
Department of Computer Engineering, Sinhgad Institute of Technology & Science, Narhe, Pune 411041

Ninad Jane
Department of Computer Engineering, Sinhgad Institute of Technology & Science, Narhe, Pune 411041

Aditi Wasvand
Department of Computer Engineering, Sinhgad Institute of Technology & Science, Narhe, Pune 411041

Santosh Darade
Department of Computer Engineering, Sinhgad Institute of Technology & Science, Narhe, Pune 411041

ABSTRACT
With the advent of emerging technologies like cloud computing, the security of confidential data is of prime importance. Firewalls are widely used as the most basic security device used to protect a network from unauthorized access and network intrusions. Network Administrators define some rules to filter incoming and outgoing packets which form the security policy of the firewall. The large size of firewall policies create complex interactions between policies of the same firewall as well as between multiple firewalls. In this paper, we extend the currently known classification for firewall policy anomalies. Further, we propose a tool which obtains these rules from security devices in real-time environment, detects the anomalies present in them according to the underlying network topology and propagates the consistent rules with the consent of administrator. Currently, the tool can only be used with Cisco security devices; however, it can be extended to incorporate the syntax of other vendor's devices as well.

General Terms
Computer Networks, Network Security

Keywords
Firewalls, ACL, rules, anomaly, Firewall Policy, Policy conflicts

1. INTRODUCTION
Due In any organization, firewalls are used as basic security devices to protect internal resources from outside attacks. The firewall(s) deployed at the network boundary act as the first line of defense. However, the need to protect organization network from internal attacks gave rise to the concept of 'Distributed Firewalls.' The basic idea of 'Distributed Firewalls' is to make every device present in the network, a firewall that filters traffic to and from itself [11]. Firewalls, especially the packet-filtering firewalls contain certain predefined rules which form the security policy of the firewall. The firewall matches all the incoming and outgoing packets passing through it with this policy and take the respective action as defined in the policy. In this paper, we will refer these policies as Access Control Lists (ACL's). As defined in [6], an ACL is of the form {Predicate} → {Decision} where {Predicate} is represented over certain predefined packet fields. Typically, Cisco IOS-based security devices [13, 15] contain the fields ACL ID, sequence number, protocol, source IP address, source wild card mask, destination IP address, destination wild card mask, destination port range and additional options (refer table 1). Every packet passing through the firewall is matched with these predicates and if match is found, then the corresponding {Decision} is taken which can be 'permit' or 'deny.'

In an organization, firewall(s) have hundreds or even thousands of ACL's. The huge size of these rules create complex interactions between them which lead to anomalies between the rules. For example, a network administrator wants to add an ACL to allow traffic from the IP address 'A.B.C.D'; however, he isn't aware weather the particular ACL already contains a rule for IP address 'A.B.C.D.' Manually, scanning the ACL to locate the rule is cumbersome/practically impossible. Thus, inserting a new rule may create anomalies in the existing policy.

1.1.Motivation
Adding or changing firewall security policy is an error-prone task due to the following reasons (considering Cisco devices) [13]:

- The rules within an ACL are sensitive to rule order. The security devices match packets based on their order in that particular list. For example, if a packet is filtered by a rule, the device decides the fate of that packet by consulting the 'action' or 'decision' field of that rule. The device doesn't explore the remaining rules after a match is found.
- Huge size of ACL's create complex interactions between rules of the same device as well as between rules of other connected devices.
- If cycles are present in a network, then multiple paths exist from source to destination which may permit some packets which should be denied or vice versa.

Recently, detection of conflicts between the firewall rules has received attention from researchers [2, 3, 4, 5, 7, 9, 10 and 11]. Several methods have been proposed from detecting and resolving the anomalies to verifying the correctness of the rules [6, 12].

1.2.Key Contributions
The challenges presented in [1] include detection of policy anomalies in real-time environment containing distributed security devices. In this paper, we tackle this issue of obtaining policy rules (ACL's) from real Cisco devices along

with information about multiple interfaces of the device. Also, we detect anomalies in inter-firewall environment having multiple paths from source to destination. Physical network security devices contain multiple interfaces, each having different set of policy rules. This classification of intra-firewall anomalies have not been explored in previous work.

Table 1. ACL representation according to Cisco security devices

ACL ID	Seq .No	Protocol	Source IP	Source Wild Card Mask	Dest. IP	Dest. Wild Card Mask	Dest. Port	Dir.	Action
5	**10**	IP	192.168.2.0	0.0.0.255	160.12.0.2	0.0.0.0	any	OUT	Deny
101	**20**	IP	192.168.1.2	0.0.0.0	200.1.0.0	0.0.255.255	80	OUT	Permit
101	**25**	TCP	any	any	192.168.0.0	0.0.0.255	22	IN	Deny

1.3.Paper Organization

The rest of the paper is organized as follows. Section 2 discusses the classification of Firewall Policy Anomalies in brief section 3 presents the types of firewall anomalies valid with respect to Cisco security devices. Section 4 gives an overview of how to access Cisco devices in real-time environment. Section 5 demonstrates how to explore the linkages in the organization's network. Section 6 and 7 provides algorithms to detect intra-firewall and inter-firewall anomalies in real-time environment along with their resolution options available. Section 8 gives a brief overview of previous work done in the detection of firewall policy anomalies.

2. FIREWALL POLICY ANOMALY CLASSIFICATION

Firewall policy anomalies were first classified by Al-Shaer et al. [3]. Firewall policy anomalies arise due to the conflicts between the firewall rules when a packet matches more than one rule. Firewall anomalies not only give rise to incorrect packet filtering, but also wastes the available space for storing ACL's. As already discussed, security devices scan rules sequentially; more number of anomalies implies more inconsistent rules which increases the time to scan through the rules to match a packet.

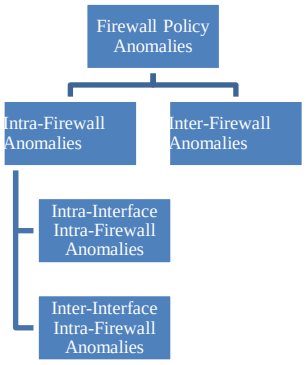

Figure 1: Classification of Firewall Policy Anomalies

2.1. Intra-Firewall Anomalies

Intra-Firewall Anomalies are the conflicts between the rules of the same security device. In other words, when the same packet match two or more rules within the same firewall, an intra-firewall anomaly is said to be present. In practice, security devices like routers, firewalls, layer 3 switch have multiple interfaces. We further categorize intra-firewall anomalies into intra-intraface and inter-interface anomalies as per the presence of multiple interfaces.

2.1.1. Intra-Interface Intra-Firewall Anomalies

These anomalies arise due to conflicts between the rules defined in the same interface, typically within one ACL; or in certain cases the anomaly can be between rules of different ACL's, but allocated to the same interface.

2.1.2. Inter-Interface Intra-Firewall Anomalies

Inter-Interface anomalies arise due to conflicts between the rules defined in different ACL's allocated to different interfaces of the same device. These anomalies occur when the ACL of one interface allows an incoming packet, but the ACL defined on the other interface through which the packet is supposed to pass is denying it or vice versa.

2.2. Inter-Firewall Anomalies

Inter-Firewall Anomalies are the conflicts between the rules of two or more devices in the same network. Inter-Firewall anomaly arise when one device permits a packet while other device in its path to destination denies it (as per the security policy defined for the device). Inter-Firewall Anomalies can be between two adjacent devices (within one subnetwork) or between the devices located in different subnetworks.

3. PREFACE TO FIREWALL POLICY ANOMALIES

Several types of firewall anomalies have been identified earlier in [3, 4 and 9]. Here, we briefly describe the anomalies considering their detection in Cisco ACL's:

3.1 Exact Match

This is the most basic type of anomaly which can occur intra-firewall as well as inter-firewall environment. Exact match anomaly occurs when two or more rules match the same set of packets except that the rules differ in 'action' field (in case of intra-interface anomalies) and/or 'direction' field (in case of inter-interface and inter-firewall anomalies).

3.2 Shadowing

The difference between exact match and shadowing anomalies is that the later occurs when a preceding rule match all the packets as that of succeeding rule. In other words, the succeeding rule 'R2' is a subset of the preceding rule 'R1' (R2 $\subset$ R1), such that the rule 'R2' is never activated. Shadowing anomaly causes incorrect 'action' to be taken. Moreover, it wastes the space allocated to ACL's of a device and increases the processing time of filtering the packets due to increase in rules.

3.3 Correlation

Correlation anomaly arises when both the preceding and succeeding rules match certain set of packets, but their 'action' differs. So, the fate of the packet is decided based on

position of the rules in ACL. Correlation is a warning and administrator should decide the positions of the rules.

3.4 Generalization

Generalization is the opposite of shadowing anomaly. In generalization, the preceding rule is a subset of the succeeding rule. Generalization is used when an administrator wants different 'action' for certain specialized packets for which a general rule is defined. Generalization too is a warning to notify the administrator about the specialized rule.

4. OBTAINING INFORMATION FROM CISCO DEVICES

Cisco devices use IOS (Internetworking Operating System); thus, all Cisco devices follow the same syntax for configuring interfaces and ACL's (except for minor changes between the IOS versions). Telnet/SSH (along with authentication) must be enabled on the device so that the system can access its configuration data.

The network administrator must provide IP address of at least one interface of each security device present in the network. Our system connects to the respective interfaces and obtains the configuration data of the entire device which contains information of remaining interfaces (if any), its ACL's, access groups, etc. In Cisco IOS, access groups specify the ACL applied to an interface along with their direction (inbound/outbound). The ACL's which are not applied to any interface, can be ignored.

Cisco IOS defines two types of ACL's, 'standard' and 'extended.' Standard ACL's allows an administrator to filter packets based only on source address while extended ACL's are used to filter packets based on source and destination addresses, protocol and destination port number. Both standard and extended ACL's have to be considered together to detect firewall policy anomalies. So, while comparing a standard ACL with an extended one, we consider standard ACL's destination address, protocol and port as 'any.'

Our system uses python at the back end to establish communication with Cisco devices. Currently, we are using python package 'telnetlib' to establish Telnet connection with the Cisco IOS. Regular expressions are used to detect interface information, ACL's and other information and store them respectively in the relational database. For extending support for security devices from other vendors, regular expressions to match the syntax of the respective vendor-specific operating system have to be constructed.

5. EXPLORING NETWORK CONNECTIONS

We device a recursive algorithm to find the connections (links) between the already explored security devices. These links are used to detect inter-firewall anomalies. Further, we have also used these explored connections to provide administrator with a pictorial view of the network topology using python's package 'NetworkX.'

The algorithm scans through the database containing interface details of the devices to determine the two devices which are present in the same network. To prevent the algorithm from looping through the cycles present in the network, we have used an array 'exploredList.' Whenever the algorithm finds a match for a particular interface (i.e. the devices containing the two interfaces are in the same network), 'exploredList' is updated with the addresses of both the interfaces. Thus, the algorithm doesn't re-explore the interface if 'exploredList' contains the address of that interface.

5.1.Working

The algorithm matches the IP/mask pair received as arguments with each ip/mask pair present in the database. When a match is found (i.e. both the interfaces are in the same subnetwork), the link is stored in the database. Then, it obtains details of other interfaces with the same hostname as that of current IP/mask pair and call the function recurrsively with the new IP/mask pair if the interface is not explored previously.

Algorithm 1: EXPLORE($name, ip, mask$)

Input: IP Address and Subnet Mask from the previous step.
In first call, the first IP Address and Subnet Mask obtained from database is taken.
Output: Stores the linkages between distributed network devices in the database.

for each *new ip/mask* ∃ *database*
do
  if *new ip/mask* ∈ *current ip/mask network*
  then
    Insert the linkage into database
    $exploredList[i] \leftarrow ip$
    $i \leftarrow i + 1$
    for each *ip/mask* ∈ *hostname* ≡ *name* ∃ *database*
    do
      if *new ip* ∉ *exploredList*
      then EXPLORE($new\ name, new\ ip, new\ mask$)

Figure 2: Algorithm: exploration of network topology

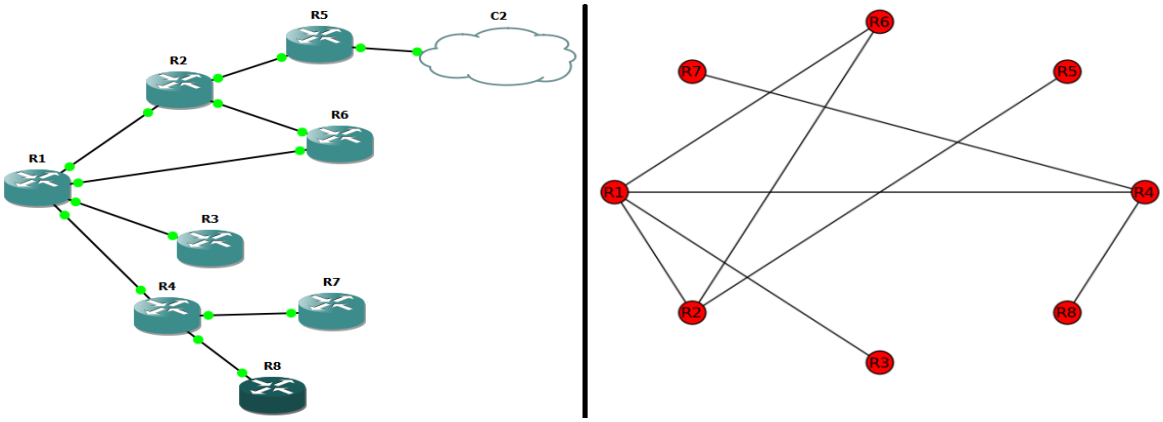

Figure 3: Actual topology designed in network emulator GNS3 and the topology generated using algorithm given in Figure 2

5.2.Advantages

As stated earlier, the discovered links are used to construct a pictorial representation of the topology which is further used to detect inter-firewall anomalies. Pedditi et al. [2] defined 'Parent-Child Relationships' to detect inter-firewall anomalies; however, it involves having a single 'root' firewall (firewall at the port of entry i.e. interface of organization's internal network with outside network). Thus, the Parent-Child Relationships wont work if multiple root firewalls are deployed which is usually the case if an organization has multiple leased lines.

6. DETECTING INTRA-FIREWALL ANOMALIES

As discussed in previous sections, Intra-Firewall Anomalies are the anomalies present in the firewall policies of the same device. We detect exact match, shadowing and correlation anomalies for intra-interface as well as inter-interface. For intra-interface anomaly detection, the order of rules matter for anomaly resolution (except in exact match).

6.1. Examples of Intra-Firewall Anomalies and Resolution Strategies

Figure 4 represents a sample set of ACL's taken from a Cisco router by connecting to it via console.

```
Standard IP access list 1
   10 permit 200.1.0.2
   20 deny   200.1.0.0, wildcard bits 0.0.255.255
   30 permit any
Extended IP access list 100
   10 deny ip 200.1.0.0 0.0.255.255 any
   20 deny 80 host 140.69.1.2 any
   30 permit ip host 200.1.0.2 any
   40 permit ip any any
Extended IP access list 101
   10 deny ip host 192.168.104.2 host 10.1.1.2
   20 permit ip any any
Extended IP access list 102
   10 deny ip 10.6.2.0 0.0.0.255 host 192.168.101.2
   20 deny 21 6.6.0.0 0.0.255.255 any
   30 permit 21 host 192.168.106.1 host 192.168.108.2
   40 deny 21 192.168.106.0 0.0.0.255 host 192.168.108.2
   41 deny 21 host 192.168.106.1 host 192.168.108.2
   50 permit ip any any
Extended IP access list 103
   10 deny 80 110.0.0.0 0.0.0.255 192.168.103.0 0.0.0.255
   20 deny 80 110.0.1.0 0.0.0.255 host 192.168.103.2
   30 permit ip any any
Extended IP access list 104
   10 deny ip 118.1.10.0 0.0.0.255 any
   20 permit ip any 160.12.0.0 0.0.255.255
   30 permit ip any any
```

Figure 4: Example configuration of ACL's containing Intra-Firewall Anomalies

The sequence numbers 30 and 41 of extended ACL 102 are *exact matches* with different 'action' fields of which one has to be deleted to resolve the anomaly. The sequence number 30 of extended ACL 100 is *shadowed* by sequence number 10 of the same ACL. Thus, either one of them has to be deleted or the sequence number have to be interchanged so as to create a specialized rule to resolve the anomaly. Sequence numbers 10 and 20 of standard ACL 1 and 30 and 40 of extended ACL 102 are examples of *generalization* anomaly warning which may be neglected or the specialized rule removed. Lastly, sequence numbers 10 and 20 of extended ACL 104 is an example of *correlation* anomaly. Here in this case, currently a packet with source IP address from network 118.1.10.0 with destination IP address from network 160.12.0.0 will be denied. If the sequence numbers are interchanged, the same packet will be permitted. To resolve this anomaly, the administrator has to either ignore it, interchange the sequence of the rules or add a specialized rule.

6.2. Algorithm for detecting Intra-Firewall Anomalies

Our Intra-Firewall Anomalies detection algorithm proceeds as follows. We detect Intra-Interface and Inter-Interface anomalies in a single scan through all the ACL's. We have used a function 'DetectAnomaly' to compare two rules from the same or different ACL. This function checks whether the provided two rules conflict with each other.

Working: First, we check for inter-interface anomalies; thus, we store all the ACL's applied to an interface in 'currentACL.' To reduce the number of calls made to 'DetectAnomaly,' we only compare the current rule with the succeeding rules (variable 'j' is used for that purpose).

For detecting inter-interface anomalies, first, we use 'findIndex' function to find all the interface addresses of that device except the current one. Thus, by obtaining the ACL's of remaining interfaces and comparing them with the current rule, we can determine anomalies present between the interfaces of the same device.

As per algorithm presented in Figure 6, 'DetectAnomaly' accepts two rules and returns the anomaly type (if any) between them. The two rules considered are 'acl1[index]' and 'acl2.' The anomalies are stored in database along with the two rules and the administrator is prompted to take further actions.

Algorithm 2 : DETECTINTRAANOMALIES($AllACL$)

Input: All Access Control Lists
Output: Stores the intra-firewall policy anomalies in the database.

comment: Detect Intra-Interface Anomalies

for each $acl \in AllACL$
do
- **if** $currentIP \equiv acl.ip$
  - **then** $currentACL[i] \leftarrow acl$; $i \leftarrow i+1$
  - $j \leftarrow 1$
  - **for each** $currentACL$
    - **do** **for** $k \leftarrow j$ **to** i **do** DETECTANOMALY($k, currentACL, acl$); $j \leftarrow j+1$
  - **comment:** Detect Inter-Interface Anomalies
  - **else** $currentIP \leftarrow acl.ip$
  - $index \leftarrow findIndex(acl.ip)$
  - $InterACL \leftarrow AllACL.ip \in index$
  - **for** $i \leftarrow 0$ **to** $len(InterACL)$ **do** DETECTANOMALY($i, InterACL, acl$)
  - $currentACL.clear()$
  - $i \leftarrow 0$

Figure 5: Algorithm: Detecting Intra-Firewall Anomalies

Algorithm 3 : DETECTANOMALY($index, acl1, acl2$)

Input: Two possibly conflicting ACL's.
Output: If the ACL's are conflicting, return its anomaly type.

comment: Check for Exact Match
if $acl1 \equiv acl2$
then { $anomaly \leftarrow exact\ match$; **return** ($anomaly$) }
comment: Check for shadowing
if $acl1 \subset acl2$
then { **if** $InterInterface$ and $acl1.direction \neq acl2.direction$ and $acl1.action \neq acl2.action$
then $anomaly \leftarrow shadowing\ InterInterface$
else $anomaly \leftarrow shadowing$
return ($anomaly$) }
comment: Check for correlation
if $acl1.src \subset acl2.src \wedge acl2.dest \subset acl1.dest$ $\vee\ acl2.src \subset acl1.src \wedge acl1.dest \subset acl2.dest$
then { $anomaly \leftarrow correlation$; **return** ($anomaly$) }

Figure 6: Algorithm: Check for conflicting rules

7. DETECTING INTER-FIREWALL ANOMALIES

As previously discussed, Inter-Firewall Anomalies are the anomalies between rules of two distinct security devices. Cisco security devices have two types of flows which are used to filter traffic: inbound and outbound. When an ACL is applied to an interface in 'inbound' direction, the packets are which are coming inside the interface are filtered. On the other hand, when an ACL is applied in 'outbound' direction, the packets going outside of the interface are filtered. If any ACL is not applied on an interface in either of the direction, the traffic flowing through that direction is not filtered. For instance, if any ACL is not applied 'outbound' of an interface, then the device allows all traffic flowing outside of the interface.

7.1. Conditions for existance of Inter-Firewall Anomalies

As per our observations, the conditions for inter-firewall anomaly to exist (with respect to the 'direction' attribute) include:

- For the two interfaces which belong to the same subnetwork, the direction cannot be the same i.e. we have to check outbound rules of one interface with the inbound of the other interface or vice versa for inter-firewall anomaly to exist.
- For all the other interfaces which belong to the same device which contains an interface belonging to the same subnetwork as that of the currently inspected interface, the direction attribute should be same.
- For all the other devices, the interfaces which are discovered first whilst traveling from the current interface to that particular device should be checked for opposite direction rules (outbound to be compared with inbound and vice versa) and for the remaining interfaces of that device, the rules having direction attribute same should be checked.

To understand the above conditions, consider a small network consisting of three devices.

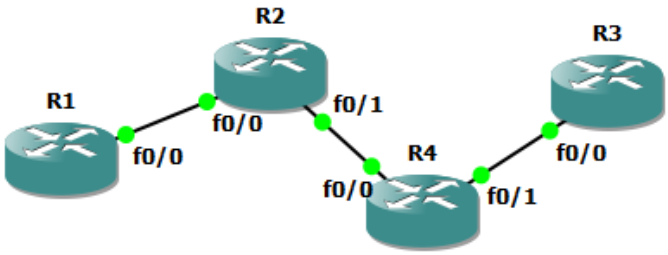

Figure 7: Example network with predefined inbound/outbound ACL's

R1, R2, R3 and R4 are the hostnames of devices shown in Figure 7 and f0/0, f0/1 are their respective interface names. Consider that only outbound rules are configured on device R1's interface f0/0. Now according to the first condition, inter-firewall anomalies can exist only when the outbound rules of {R1, f0/0} and inbound rules of {R2, f0/0} conflict. As per the second condition, inter-firewall anomalies exist only when the outbound rules of {R1, f0/0} and outbound rules of {R2, f0/1} conflict (same 'direction' attribute). Now, as per the third condition, the interface f0/0 of device R4 is discovered first when we travel from f0/0 of device R1 to device R4; thus, the outbound rules of {R1, f0/0} should be checked with inbound rules of {R4, f0/0} (opposite 'direction' attribute). Likewise, the outbound rules of {R1, f0/0} should be checked with outbound rules of {R4, f0/0}.

7.2. Algorithm for detecting Inter-Firewall Anomalies

Algorithm 4 : DETECTINTERANOMALIES($acl1, acl2$)

Input: ACL's of two interfaces
Output: Stores the inter-firewall policy anomalies in the database.

comment: Combine ACL's of two interfaces.
$acl \leftarrow acl1 \bigcup acl2$
for each $currentACL \in acl$
do { **for** $i \leftarrow 0$ **to** $len(acl)$
do { **if** $currentACL.direction \neq acl[i].direction$ and $currentACL.action \neq acl[i].action$
then DETECTANOMALY($i, currentACL, acl$)
comment: Find other interfaces with hostname acl[i].name
for each $interface \in hostname = acl[i].name$
do { MULTIINTERFACEANOMALY($currentACL, interface$)
comment: Check other devices in same n/w.
comment: Let AllInt contain all interfaces in the network.
for $i \leftarrow 0$ **to** $len(AllInt)$
do { **if** $interface \subset AllInt[i]$
then DETECTINTERANOMALIES($interface.acl$, $AllInt[i].acl$) } } }

Algorithm 5 : MULTIINTERFACEANOMALY($acl, interface$)

Input: One ACL and One Interface.
Output: Check for conflicts between ACL and Interface's ACL's.

comment: Let IntACL contain the given interface's ACL's
for each $currentACL \in IntACL$
do { **if** $acl.direction \equiv currentACL.direction$ and $acl.action \neq currentACL.action$
then DETECTANOMALY($0, acl, currentACL$) }

Figure 8: Algorithm: Detecting Inter-Firewall Anomalies

Our inter-firewall detection algorithm proceeds in three main stages.

7.2.1. First, the algorithm checks rules for conflicts in the ACL's between interfaces present in the same subnetwork using the 'DetectAnomaly' function previously discussed. Here, the 'direction' attribute should be opposite of each other (as the comparison is being made with ACL's of the current interface and the interface of other device which is discovered first traveling from the current interface to the other device being considered).

7.2.2. In the second stage, the algorithm searches for the remaining interfaces of the other device and executes 'MultiInterfaceAnomaly' which detects anomalies between ACL's of the current interface and the remaining interfaces. Here, the 'direction' attribute of both the ACL's should be same as per the conditions discussed earlier.

7.2.3. In the third stage, the algorithm checks for the interfaces in the entire network which is in the same subnetwork as that of the interfaces chosen for executing the second stage and calls the algorithm recurrsively using ACL's of the aforementioned interfaces.

7.3. Resolution of Inter-Firewall Anomalies

The anomalies for Inter-Firewall environment are similar to as that of Intra-Firewall environment except that in inter-firewall anomalies, we are dealing with two distinctly separate devices. So, we only discuss the resolution options provided by the implemented system.

Inter-Firewall *shadowing* anomaly can be resolved by either deleting or modifying one of the rules. Unlike intra-firewall intra-interface shadowing anomaly, we cannot resolve the anomaly by relocating the rules between two devices.

Like Intra-Firewall anomaly resolution for exact matches, inter-firewall *exact match* anomaly can be resolved by deleting one of the conflicting rules.

Inter-Firewall correlation anomaly warnings, unlike intra-firewall correlation anomaly warnings cannot be resolved as per the predetermined security policy as the traffic is going to pass through both the devices (i.e. if an upstream device permits a packet, the packet is going to pass through the downstream firewall where it will be denied since the rules of the devices are correlated). To resolve inter-firewall correlation anomaly warnings, we can either delete one of the rules, modify one of the rules or add a specialized rule for permitting/denying the traffic under consideration in both the devices.

8. RELATED WORK

We discuss the related work that intersects our work in: discovery of firewall policy anomalies, and distributed firewall policy management.

Several methods for detecting anomalies have been proposed. Al-Shaer and Hamed [3] first presented 'Firewall Policy Advisor' tool to detect the existing intra-firewall and inter-firewall anomalies. The tool uses state-diagram based approach which is not suitable for large distributed networks. Bartal et al. [5] designed a UML like language for representing firewall policy rules which are obtained from a model compiler which translates the rules into firewall configuration files. A. Mayer et al. [11] designed a tool to analyze firewall policy rules. However, the analysis is done offline i.e. the tool cannot be used to handle distributed security devices. Lihua Yuan et al. [3] introduced a tool 'Fireman' which is used for static analysis of distributed devices.

Several tools have been developed using Binary Decision Diagrams [8, 9] which are effective in terms of representing firewall policy anomalies; however, the proposed tools are incapable of obtaining ACL's from real-time distributed security devices. As discussed previously, our tool obtains ACL's from real-time security devices, detects anomalies between them and reflects the consistent rules back to the devices with the consent of the administrator.

Pedditi et al. [2] designed and simulated a novel protocol for firewall communication which can be used to add appropriate rules in case of outside network attacks. The paper suggests that the firewalls can automatically check for anomalies between their policy rules through message passing. However, the protocol is in the design stage and not implemented practically. Furthermore, the ways to resolve the firewall policy anomalies is unknown.

9. CONCLUDING REMARKS AND FUTURE WORK

In this paper, we have presented a tool which can be used to detect and resolve firewall policy anomalies in real-time environment. The tool presents the anomalies to the administrator with options available to resolve them. It also reflects the changes/resolved anomalies to the respective device(s) with the consent of the administrator. We have also classified intra-firewall anomalies further according to the interfaces present activated in the device and provided an algorithm to detect them. The tool can also be used— (a) when an organizational network has cycles (multiple paths exist between source and destination) present in the pre-deployed network; (b) when multiple 'root' firewalls are deployed in the network. We believe that this tool will be useful for network administrators to resolve policy anomalies in pre-deployed distributed networks.

Our future work includes incorporating syntax of operating systems provided by vendors other than Cisco (for e.g. JUNOS used in Juniper devices) so that the system can be used globally for any devices deployed in the network. Also, we would like to extend our anomaly detection to handle anomalies in application gateway policies.

10. REFERENCES

[1] Ameya Hanamsagar, Ninad Jane, Bhagyashree Borate, Aditi Wasvand and S. A. Darade, "Firewall Anomaly Management: A survey," International Journal of Computer Applications Volume 105 Number 18

[2] Sandeep Reddy Pedditi, Du Zhang, and Chung-E Wang, "FIEP: An Initial Design of A Firewall Information Exchange Protocol," IEEE 14th International Conference on Information Reuse and Integration (IRI), 2013

[3] E. Al-Shaer and H. Hamed, "Discovery of Policy Anomalies in Distributed Firewalls,"IEEE INFOCOM '04,vol. 4, 2004. pp. 2605-2616

[4] L. Yuan, H. Chen, J. Mai, C. Chuah, Z. Su, P. Mohapatra, and C. Davis, "Fireman: A Toolkit for Firewall Modeling and Analysis," Proc. IEEE Symp. Security and Privacy, 2006

[5] Y. Bartal, A.J. Mayer, K. Nissim, A. Wool, "Firmato: A novel firewall management toolkit," ACM Transactions on Computer Systems 22, 2004, pp. 381-420

[6] Suchart Khummanee, Atipong Khumseela and Somnuk Puangpronpitag, "Towards a New Design of Firewall: Anomaly Elimination and Fast Verifying of Firewall Rules," 10th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2013, pp. 93-98

[7] Chi-Shih Chao, "A flexible and feasible anomaly diagnosis system for Internet firewall rules," 13th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2011

[8] A. X. Liu and M. G.Gouda, "Firewall policy queries," IEEE Transactions on Parallel and Distributed Systems (TPDS), 20(6), pp. 766–777, 2009

[9] Hongxin Hu, Gail-Joon Ahn and Ketan Kulkarni, "Detecting and Resolving Firewall Policy Anomalies," IEEE Transactions on Dependable and Secure Computing, vol. 9, issue 3, pp. 318-331

[10] Alan Jeffrey and Taghrid Samak, "Model Checking Firewall Policy Configurations," IEEE International Symposium on Policies for Distributed Systems and Networks, 2009, pp. 60-67

[11] A. Mayer, A. Wool and E. Ziskind, "Offline firewall analysis," International Journal of Information Security 5 (3), 2005, pp. 125–144

[12] Alex X. Liu, "Firewall policy verification and troubleshooting," The International Journal of Computer and Telecommunications Networking, Vol 53 Issue 16, 2009, pp. 2800-2809

[13] Cisco ASA Series Firewall ASDM Configuration Guide, Cisco Systems Inc., updated March 31, 2014

[14] S. R. Pedditi, "An initial design of firewall information exchange protocol (FIEP)," MS Degree Project Report, Department of Computer Science, California State University, Sacramento, May 2012.

[15] Cisco Security Appliance Command Line Configuration Guide, Cisco Systems Inc., 2009