Improving Firewall Performance by Eliminating Redundancies In Access Control Lists

Ajay Krishna Vasu *ajay_krishna_v@yahoo.co.in*
Computer Science Department
Sri Venkateswara College of Engineering
Pennalur, 602117, India

Ashwin Ganesh *ariel_ash@yahoo.com*
Computer Science Department
Sri Venkateswara College of Engineering
Pennalur, 602117, India

Priya Ayyappan *appy178@gmail.com*
Computer Science Department
Sri Venkateswara College of Engineering
Pennalur, 602117, India

Anirudhan Sudarsan *anirudhan.sudarsan@gmail.com*
Computer Science Department
Sri Venkateswara College of Engineering
Pennalur, 602117, India

Abstract

A firewall is a network security device that works to protect an organization's internal network from both unauthorized and malicious users. It functions by examining all packets that enter any one of its incoming interfaces and comparing the structure of the packet against a set of predefined rules. Each rule specifies if a packet corresponding to the rule is to be permitted or denied. This set of rules is called an access control list (ACL) and it forms the basis of a firewall's policy. Incorrect configuration of the firewall can lead to redundant rules which cause performance degradation. We propose an algorithm to identify and eliminate redundant rules in an access control list during the configuration phase. The proposed work defines an access control list as a linked list data structure. A comparison of the proposed work and the conventional approach is also presented.

Keywords: Firewall, Access Control List, Network Security, Firewall Configuration, Firewall Policy.

1. INTRODUCTION

Firewalls serve as perimeter defence devices for private corporate networks ranging from small to huge. Such networks are under constant threat from attackers who attempt to penetrate them in view of financial or other forms of personal gain. Firewalls are hence installed to prevent such occurrences. These firewalls prevent packets both from being routed into the network and from being routed out of the network unless the source host device has the necessary permissions to access a particular resource.

A firewall achieves this by acting as a gatekeeper for the network i.e. they act as the entry or exit point of the network. All packets, whether incoming or outgoing have to pass through the firewall [1].

The firewall is best implemented through packet filtering technology [2]. A packet can be considered to be a structure with various attributes such as source IP address, destination IP address, source port and destination port. Each packet will correspond to a rule defined in the ACL or will not be mapped onto to any rule in which case, it will be mapped onto an implicit deny. A packet filtering firewall functions by comparing the attributes of each incoming packet against the rules defined in the access control list [3]. The firewall then takes a decision to either route the packet into the network or to drop the packet. Today's firewalls also include additional options to log the incoming packets for later analysis [4].

1.1 Overview of Various Firewall Technologies

The main reason why firewalls are utilized is because of the inefficiency of encryption algorithms in containing malicious packets from being routed into the private network [2]. Some of the common firewall technologies are presented below,

Stateful packet filter- A stateful packet filtering firewall inspects the state of existing network connections in order to make a decision on whether to forward a packet or filter it. If the incoming packet is a legitimate request for a new or part of an existing connection, it is routed through to the internal network [2]. The working of stateful packet filter is based on the concept that packets from the same source need not be examined repeatedly as long as they belong to the same connection. It is known as a dynamic packet filtering technology [5]. The advantage of this type of firewall is that it considerably reduces the average number of comparisons required before a packet is matched with the firewall rules. It provides a greater level of security and is also easy to utilize [3].

Stateless packet filter- A stateless packet filtering firewall is the simplest firewall to implement from the point of view of implementation complexity as well as functionality. It is the most widely used firewall technology. These firewalls do not store any connection related information. Each packet is treated as an independent entity [2]. The firewall examines each incoming packet and decides to either route it or drop it.

Application Gateways- Application gateways basically play the role of a proxy. They process service requests from external clients. An application level gateway performs more thorough inspections on a packet than the average packet filter. It functions even at the application layer by examining the format of the application contained in the packet. The application level gateway can hence detect and block packets if they carry viruses and other malicious code fragments, in addition to blocking them based on IP addresses [2]. When acting as a proxy, the application level gateway provides authentication mechanisms such as username password combinations. It can also provide a detailed log of all the actions it has taken on various packets. The drawbacks of this type of firewall are the implementation complexity which leads to both a complex as well as a slowed down operation as well as a lack of support for new applications and protocols [3 and 6].

Circuit Gateways- Also known as circuit level gateways, these firewalls predominantly function at the transport layer. They make the decision to route packets or filter them based on both IP addresses as well as the port numbers contained in the transport layer header. This header can be either a TCP or a UDP header. When combined with a packet filtering firewall, it is termed as a dynamic packet filter [2]. Circuit level gateways can examine and validate the formation of TCP connections by observing the three way handshake occurring [7]. It also maintains the connection state just like a stateful packet filter and permits packets only if they belong to one of the existing connections [2].

Despite all the technological advancements in firewall technologies, firewalls continue to have a few limitations including the following,

1) They do not deal with threats fully. They basically prevent only unauthorized users and applications from entering the trusted network while providing access to permitted users. A firewall fails, if an unauthorized user has already entered the network. In such cases, it

cannot prevent malicious activities from being carried out. There is a pressing need to also implement additional security measures such as sniffing and encryption [7].

2) An application gateway can identify viruses and malicious code, but by itself, it cannot destroy the source of the attack [5].

3) Modern networks are typically large, i.e. they contain several hundreds if not thousands of network devices including hosts, switches and routers. There is an increasing chance that an attack can be carried out by hosts on the internal network itself. Traditional firewalls are powerless in such scenarios [7].

There are several firewalls that are available in the market today such as the Checkpoint SPLAT, Cisco ASA and the freely available OpenBSD packet filter. The three aforementioned firewalls do a very good job of protecting the private network from intruders.

Performance testing of these three firewalls in a lab environment has indicated that the Cisco ASA exhibits the best performance for several indicators such as HTTP throughput, TCP throughput and UDP throughput. The HTTP throughput of the Cisco ASA was found to be nearly twice as much as that of the SPLAT and the OpenBSD packet filter. However, for indicators such as Concurrent Connections and Connections per second, the BSD was found to be the best performer.

There is a strong need to improve firewall technology as well as the performance due to an increasing amount of regulations such as the CobiT framework, the Payment-Card Industry Data Security Standard and the NIST standard [4].

Hereon, we use the term “access list” to refer to an access control list unless specified otherwise.

1.2 Working of An Access List In Stateless Packet Filtering Firewall

There are two types of access lists that can be created to define a firewall’s policy. They are,

- Standard Access List
- Extended Access List

We first define a simplified structure for an IP packet as follows,

struct packet {
Source IP address
Destination IP address
Source port number
Destination port number
}

The structure of the packet contains a source IP address, destination IP address, source port number and destination port number (transport layer). In reality, a packet will contain many more attributes such as version, internet header length, total length, identification, flags, time to live, etc. However, such additional fields have been ignored in the above defined structure as they play a very negligible role in the working of a stateless packet filter.

A standard access control list is one which uses only the *Source IP address* attribute in the packet’s structure to decide whether to forward the concerned packet or to filter it. The *Source IP address* of the packet is compared with each rule defined in the access list until a match is obtained. The other attributes are ignored. An extended access control list on the other hand, uses all four attributes defined in the packet’s structure to determine whether to forward the packet or not. Similar to the working of a standard access list, here also, the packet’s attributes are compared sequentially against the rules in the access list, until a match is obtained. In this

case, a match occurs if and only if every condition defined in the rule is matched successfully with the packet's corresponding attributes [8].

Comparisons usually occur until a rule is matched with the packet or none of the rules match with the packet. In case none of the rules can be successfully matched, then, an implicit deny is applied to the packet by default. This means that any packet not explicitly permitted by any rule in the access list will be denied by default. For the purpose of this paper's analysis, a Cisco based standard access list was considered.

FIGURE 1 shows an example of a network. It must be noted that the figure is not indicative of a production network. The cloud denotes an untrusted network such as the internet. The router r1 serves to represent a perimeter security device such as a firewall. The other devices are a part of the trusted network.

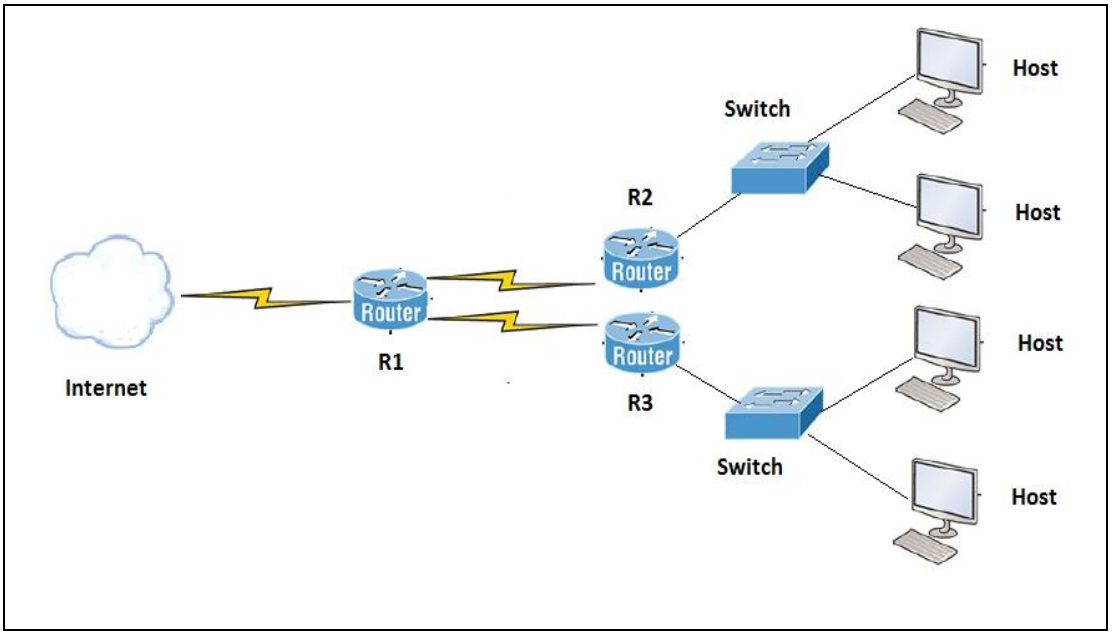

FIGURE 1: Example of a Network.

Private networks usually employ RFC 1918 addressing on their internal network devices and implement network address translation to get onto the internet. All class A, B and C addresses that are not a part of the RFC 1918 ranges are public IP addresses. The range of private address is as follows,

Class A: 10.0.0.0 /8 to 10.255.255.255 /8.
Class B: 172.16.0.0 /12 to 172.31.255.255 /12
Class C: 192.168.0.0 /16 to 192.168.255.255 /16

The command format for defining a standard ACL based on a Cisco router is as follows:
access-list [access-list number 1-99] [permits or deny] [source IP address] [wild card mask]

Once it has been defined, the access list must be applied to an interface of a firewall or a router which can also act as one. The access list can be applied in two directions in each interface- either outbound or inbound. When it is applied to the outbound interface, then the packet is first routed to that interface before it is compared against the rules in the access list. If the access list is applied on an inbound interface, the packet is first compared with the rules in the ACL before

being routed or dropped. The firewall policy is considered functional only after being applied to the interface [8]. The following is an example of an ACL.

1. access-list 10 permit 154.10.5.1 0.0.0.0
2. access-list 10 deny 154.10.0.0 0.0.255.255
3. access-list 10 permit 132.15.4.0 0.0.0.255
4. access-list 10 deny 132.15.0.0 0.0. 7.255
5. access-list 10 permit 15.248.27.4 0.0.0.0
6. access-list 10 permit 17.24.142.0 0.0.0.255

The above access list functions as follows,

a) Packets from the host address 154.10.5.1 are permitted but packets from the rest of the 154.10.0.0 /16 network are denied.
b) Packets from the 132.15.4.0 /24 network are permitted but packets from the remaining addresses in the 132.15.0.0 /21 network are denied.
c) Packets from the host address 15.248.27.4 are permitted.
d) Packets from the 17.24.142.0 /24 network are permitted.
e) An implicit deny is enforced by default at the end of the access list.

An important concept related to access list creation is rule ordering which has been the subject of a lot of significant research recently. Optimizing the ordering of the rules will lead to a much better performance by the firewall in terms of reduction in the number of comparisons required before a rule is matched to the packet. However, optimizing the cost of the packet matching process is an NP hard problem [9].

When configuring an access list, care must be exercised in ensuring that the rules are ordered correctly so as to prevent problematic behaviour by the firewall. This is based on the fact that a firewall works by comparing packets against the access list rules in a sequential manner [8]. One such a problem is illustrated below. The access list defined above has been modified by interchanging rules 1, 2 and rules 3, 4. However, the intent behind defining this access list is still the same. The altered access list is shown below,

1. access-list 11 deny 154.10.0.0 0.0.255.255
2. access-list 11 permit 154.10.5.1 0.0.0.0
3. access-list 11 deny 132.15.0.0 0.0. 7.255
4. access-list 11 permit 132.15.4.0 0.0.0.255
5. access-list 11 permit 15.248.27.4 0.0.0.0
6. access-list 11 permit 17.24.142.0 0.0.0.255

The altered access list functions as follows,

a) All packets from the 154.10.0.0 /16 network are denied.
b) All packets from the 132.15.0.0 /21 network are denied.
c) Packets from the host address 15.248.27.4 are permitted.
d) Packets from the 17.24.142.0 /24 network are permitted.
e) An implicit deny is enforced by default at the end of the access list.

As mentioned above, both access list 10 and 11 were defined with the same intent. However, access list 11 does not permit packets from the host address 154.10.5.1 while a packet from the same address is permitted by access list 10. This is because, when access list 10 is applied, the packet is first compared with the rule *access-list 10 permit 154.10.5.1 0.0.0.0* which routes the packet into the private network. However, when access list 11 is applied, the packet is compared with the rule *access-list 11 deny 154.10.0.0 0.0.255.255* before the rule *access-list 11 permit 154.10.5.1 0.0.0.0* which cause this packet to be dropped.

This logic can also be extended to rules 3 and 4 of the access list 10. When a packet arrives from an address in the 135.15.4.0 /24 network, it will be routed through to the internal network by access list 10. The same packet will however be dropped when access list 11 is applied as the packet will first be compared to the rule *access-list 11 deny 132.15.0.0 0.0.7.255* before the rule *access-list 11 permit 132.15.4.0 0.0.0.25*.

2. RELATED WORK

A measurable amount of research has been carried out in relation to firewalls, especially firewall performance. The stateless packet filter compares the structure of the incoming packets against rules in the access list sequentially until there is a match. This method is very inefficient as the worst case time complexity will become directly proportional to the number of rules in the access list thereby affecting its scalability. The number of rules in the firewall can grow larger due to complex user requirements and different networked applications. Some of the research work has focused on utilizing specialized data structures while others have attempted to suggest hardware based solutions [9 and 10].

[11] proposes a method based on the histograms of packet filtering to monitor firewall performance in real time and to predict the patterns of packet filtering in terms of rule order as well as rule field order. The rule order, the rule field order and the characteristics of packet flow have a significant impact on packet filtering time. This paper suggests an approach to optimize early acceptance and rejection path. This method uses histograms of both packet matching rule and packet not matching rule fields. The presented algorithm calculates the histograms in terms of packet matching and non-matching probabilities on a real time segment basis. The packet processing time is saved by 123% when compared to static rule ordering mechanism and 104% when compared to dynamic rule ordering mechanism.

[10] presents a method that performs early rejection of unwanted flows without affecting other traffic flows. The main aim here is to have a minimum number of early rejection rules but maximum discarding effect. This adaptive technique relies on the construction of a tree structure based on packet flow and packet field values. The constructed trees for each field are combined to create an optimal statistical matching tree of all rules defined in the policy. A packet entering a network is compared to a look up table of values containing packet field characteristics of the tree structure. Until a rule is matched, subtrees of the tree are recursively called. If there is no rule match, default filtering action takes place. The reduction in matching is maximal when the upcoming traffic distribution over field values matches the distribution of the constructed tree. However this is unlikely, because some flows start and others terminate with passage of time. Hence two types of updates are performed- those triggered by performance and those done periodically.

[12] suggests that the performance of a firewall based network can be substantially improved when the traffic behavior of packets is optimized. This paper attempts to solve the optimal rule ordering problem (ORO), which is to find a rule ordering among a given set of rules, where the relations with the previous rules has been preserved in such a way that only a minimum number of packets are matched to the rules. This paper considers the ORO problem as a Binary Integer Problem and thus solves it by using the 'branch and bound' problem along with gradient projection method, thus reducing the effect of combinatorial explosion. The space complexity of the proposed algorithm is linear and the time complexity is polynomial.

In [13], the firewall rule ordering is modified dynamically to obtain maximum efficiency by analyzing network traffic behavior and using packet matching statistics. This optimization process analyzes the traffic behavior and dynamically modifies the order of the filtering rules. The traffic is segmented into P segments of L packets and the matching ratio, mean and variance are calculated. Then, a quantity called the match ratio is calculated and the rules are dynamically reordered based on a matching rate coefficient. It is seen through simulations that the dynamic re-ordering reduces the filtering processing delays found in static rule order systems and

improves the performance of the firewall. The technique does not require any additional overhead and is scalable and easy to implement.

[14] proposes a method to optimize the early acceptance and rejection paths. It uses a set of statistical splaying filters with a binary search on prefix length (SSF-BSPL) technique to reduce packet filtering times. This technique combines three levels of filtering called the statistical Policy Filtering Level, field filtering level and cascaded filtering level. These three filtering levels are combined to enhance packet processing time. This multi-level packet filtering also helps to counter DOS attacks which target the default rule.

3. IMPLEMENTATION

The access control list is implemented in the form of a singly linked list data structure. However, we would like to make a point that using a binary search tree or an AVL tree would provide better results in terms of reducing the average number of comparisons per packet [15].

Each node in the linked list corresponds to one rule in the access list. When a packet arrives at the interface where this access list is applied, its structure will be compared with the attributes of each node until there is a match. The aim here is to reduce the number of comparisons by eliminating redundant rules. A rule in a firewall is redundant if removing that rule does not alter the functioning of the firewall. A more precise definition for a redundant rule is given in [1] - A rule r is redundant in a firewall **f** if and only if the resulting firewall **f'** after removing rule r is equivalent to **f**. There are two possible types of redundant rules that can be considered which are as follows,

Backward redundant rules- A rule R is said to be backward redundant if there exists a rule P appearing earlier than R in the access control list such that R is a subset of P.

Forward redundant rules- A rule R is said to be forward redundant if there exists a rule P appearing after R in access control list such that R is a subset of P and R and P perform the same actions while any rule Q defined between R and P is disjoint from R or performs the same action as R [16].

It has been observed in [16] that 7.8% of the rules are backward redundant while 7.2% of the rules are forward redundant leading to an overall redundancy of 15% on an average.

The algorithm that we propose in this section can be implemented internally in the firewall to affect the final configuration while the firewall is being configured. The working of this algorithm need not be known to administrators thus eliminating any potential administrative hassles.

3.1 Proposed Algorithm

We define the following variables and types for our algorithm,

List - access list defined as a linked list
Rule – A rule which is defined as a node in the **List**.
Function - An attribute of each **Rule** that takes two values- 0 and 1 corresponding to deny or permit respectively.
IP - An attribute of each **Rule** defined as an array that specifies the list of IP addresses for which the corresponding **Function** is defined.
List L - The defined access list.
Rule R1- New rule being inserted into List L.
Rule R- A rule already defined in the List L.

Algorithm **redundant-insert** (**Rule** R1, **List** L) *{*
check=0
For each Rule R in L {
If((R.Function ⊕ R1.Function)=0)

```
{
Array IP[] = R.IP[] U R1.IP[]
if (IP[]=R.IP[]) {
        Check=0
        Break
        }
else if (IP[]= R1.IP[]) {
        Remove(R)
        Insert (R1)
        Check=0
        Break
        }
else {
        Check=1
        Continue;
        }
} //end of if
} //end of Loop
if (check=1)
insert (R1)
} //end of code
```

This algorithm takes as input the rule to be added (Rule R1) and the access list to which the rule is to be added (List L). Rule R1 is then compared with each rule R defined previously in List L. The algorithm then checks if R1 and R both perform the same function. If the function performed is the same then, the IP addresses are compared.

If Rule R has defined IP addresses that include the ones defined in R1 as well as some additional addresses, R1 is considered redundant and not added to List L. If R1 contains all IP addresses contained in R and also a few more, then Rule R is removed from the list and R1 is inserted at the end. If neither of the cases occurs, then the loop proceeds until either of the cases occur or there are no more rules left to compare. If neither of the cases occurs till the very end, then the Rule R1 is inserted at the end of the list.

We define the following access list to provide a practical illustration of the algorithm. The rules are numbered for the purpose of ease of understanding.

1. access-list 25 deny 100.25.24.0 0.0.0.255
2. access-list 25 permit 154.63.0.0 0.0.255.255
3. access-list 25 deny 201.23.24.25 0.0.0.0
4. access-list 25 permit 220.0.25.0 0.0.255.255
5. access-list 25 permit 154.63.128.0 0.0.127.255
6. access-list 25 permit 154.0.0.0 0.255.255.255

FIGURE 2 represents this access list with six rules where rules 2, 5 are redundant as rule 6 performs their function.

FIGURE 2: Access list with 6 rules.

When the **redundant-insert** algorithm is implemented then the access list creation will proceed like in FIGURE 3 and FIGURE 4. When there is an attempt to insert rule 5 into the access list, it is

not added as rule 2 performs the function of rule 5 while also covering a wider range of addresses. When rule 6 is inserted into the access list, rule 2 is removed as rule 6 performs the same function as rule 2 while covering a larger range of addresses.

FIGURE 3: Access list after attempting to insert rule 5.

FIGURE 4: Access list after inserting rule 6.

3.2 Testing Setup

The following ACL with redundant rules was defined for the testing phase-

access-list 40 deny 164.27.48.0 0.0.7.255
access-list 40 permit 131.126.128.0 0.0.127.255
access-list 40 permit 8.8.16.0 0.0.0.255
access-list 40 permit 7.4.25.36 0.0.0.0
access-list 40 deny 27.0.0.0 0.15.255.255
access-list 40 deny 164.27.32.0 0.0.31.255
access-list 40 permit 18.14.0.0 0.0.255.255
access-list 40 deny 96.24.32.0 0.0.31.255
access-list 40 permit 180.64.72.0 0.0.3.255
access-list 40 permit 7.4.0.0 0.0.255.255
access-list 40 permit 195.254.124.128 0.0.0.127
access-list 40 permit 206.121.64.192 0.0.0.63
access-list 40 deny 96.24.0.0 0.0.127.255
access-list 40 deny 175.10.128.0 0.0.63.255
access-list 40 permit 100.1.1.1 0.0.0.0

The **redundant-insert** (Rule, List) algorithm was not used when defining this list. The following rules in access-list 40 are redundant. Only one of each pair is required for the firewall to function correctly.

a) *access-list 40 deny 164.27.48.0 0.0.7.255* and *access-list 40 deny 164.27.32.0 0.0.31.255*
b) *access-list 40 permit 7.4.25.36 0.0.0.0* and *access-list 40 permit 7.4.0.0 0.0.255.255*
c) *access-list 40 deny 96.24.32.0 0.0.31.255* and *access-list 40 deny 96.24.32.0 0.0.31.255*

In each of the above three cases, the second rule makes the first one redundant. The presence of three additional rules in the access list takes a considerable toll on the overall performance of the firewall in terms of the average number of comparisons required per packet before a rule is matched successfully. However, when the same access list is defined using the **redundant-insert** algorithm, we get the following completed ACL.

access-list 41 permit 131.126.128.0 0.0.127.255
access-list 41 permit 8.8.16.0 0.0.0.255
access-list 41 deny 27.0.0.0 0.15.255.255

access-list 41 deny 164.27.32.0 0.0.31.255
access-list 41 permit 18.14.0.0 0.0.255.255
access-list 41 permit 180.64.72.0 0.0.3.255
access-list 41 permit 7.4.0.0 0.0.255.255
access-list 41 permit 195.254.124.128 0.0.0.127
access-list 41 permit 206.121.64.192 0.0.0.63
access-list 41 deny 96.24.0.0 0.0.127.255
access-list 41 deny 175.10.128.0 0.0.63.255
access-list 41 permit 100.1.1.1 0.0.0.0

The access list 41 defined using the redundant-insert algorithm has only twelve rules as against fifteen rules in access list 40. This should in theory, lead to a twenty percentage improvement in firewall performance in terms of average number of comparisons for a packet match.

3.3 Testing Process

We define the term “unmatched packets” as the incoming packets that are not matched to any rules in the access list which means that such rules will be handled by the implicit deny in the access list.

Both C and Java programming were used for our implementation. The results obtained were found to be similar in both implementations. We simulated the generation of 10000 packets some of them that could be matched with a rule in the access list and some of them which did not match any rule in the list. The comparison is performed between two simulated firewalls- one that follows a conventional approach and one that implements the **redundant-insert** algorithm.

First the average number of comparisons per packet was calculated for the access list 40 that was not optimized by the **redundant-insert** algorithm. We also varied the percentage of unmatched packets from zero to fifty, with the aim of calculating the average number of comparisons under more realistic or practical scenarios were we would most likely witness several incoming unmatched packets.

Then, the average number of comparisons per packet was calculated for access list 41 which was defined using the **redundant-insert** algorithm. The optimized access list in theory would hold a cutting edge in terms of performance when compared to the access list 40. We again varied the percentage of unmatched packets from zero to fifty.

For both the non-optimized and the optimized access lists, the total number of comparisons was first calculated and then the average number of comparisons per packet was calculated and averaged out over fifty iterations of the simulation. The formula used to calculate the average number of comparisons was as follows,

Average number of comparisons per packet= Total number of comparisons/ Total number of packets
We represent the average number of comparisons as C_{avg}.

3.4 Results

The following results were observed after the implementation.

a) For the non-optimized access list 40, C_{avg} was found to be 7.988 with no packets going unmatched. When the percentage of unmatched packets was increased to 12.5, C_{avg} jumped to 8.824. When the number of unmatched packets was at 16.66%, C_{avg} was 9.17. When the unmatched packets percentage was increased to 25, C_{avg} increased correspondingly to 9.75. When the unmatched packet percentage was further increased to 33.33, C_{avg} increased to 10.324. C_{avg} increased to a final value of 11.497 when the unmatched packet percentage was amplified to 50.

b) For the optimized access list 41, C_{avg} was found to be 6.499 when there were no unmatched packets. When the unmatched packet percentage was increased to 12.5, the C_{avg} value proportionately increased to 7.281. The C_{avg} further underwent an increase to 7.44 with the rise of percentage of unmatched packets to 16.66. C_{avg} further increased to take values of 7.875, 8.348 and 9.25 as the percentages of unmatched packets grew to 25, 33.33 and 50 respectively.

The percentage increase in performance degradation in terms of an increasing C_{avg} was also observed and tabulated. ΔC_{avg} represents the percentage increase in performance degradation relative to no unmatched packets.

Unmatched packet percentage	C_{avg} – access list 40	C_{avg} – access list 41
0	7.988	6.499
12.5	8.824	7.281
16.66	9.171	7.44
25	9.75	7.875
33.33	10.324	8.348
50	11.497	9.25

TABLE 1: C_{avg} value for both the non-optimized and the optimized access list.

Percentage of unmatched packets	ΔC_{avg} for access list 40	ΔC_{avg} for access list 41
12.5	10.465	12.032
16.66	14.809	14.479
25	22.058	21.172
33.33	29.243	28.450
50	43.00	42.32

TABLE 2: ΔC_{avg} due to increase in unmatched packet percentage.

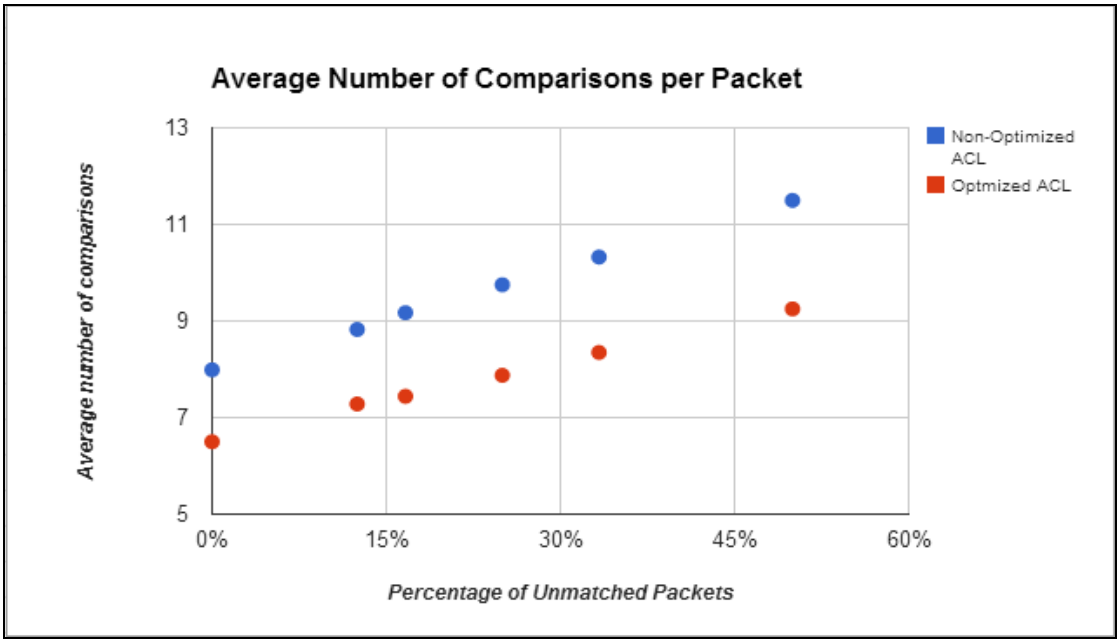

FIGURE 5: C_{avg} for various percentages of unmatched packets for both access lists.

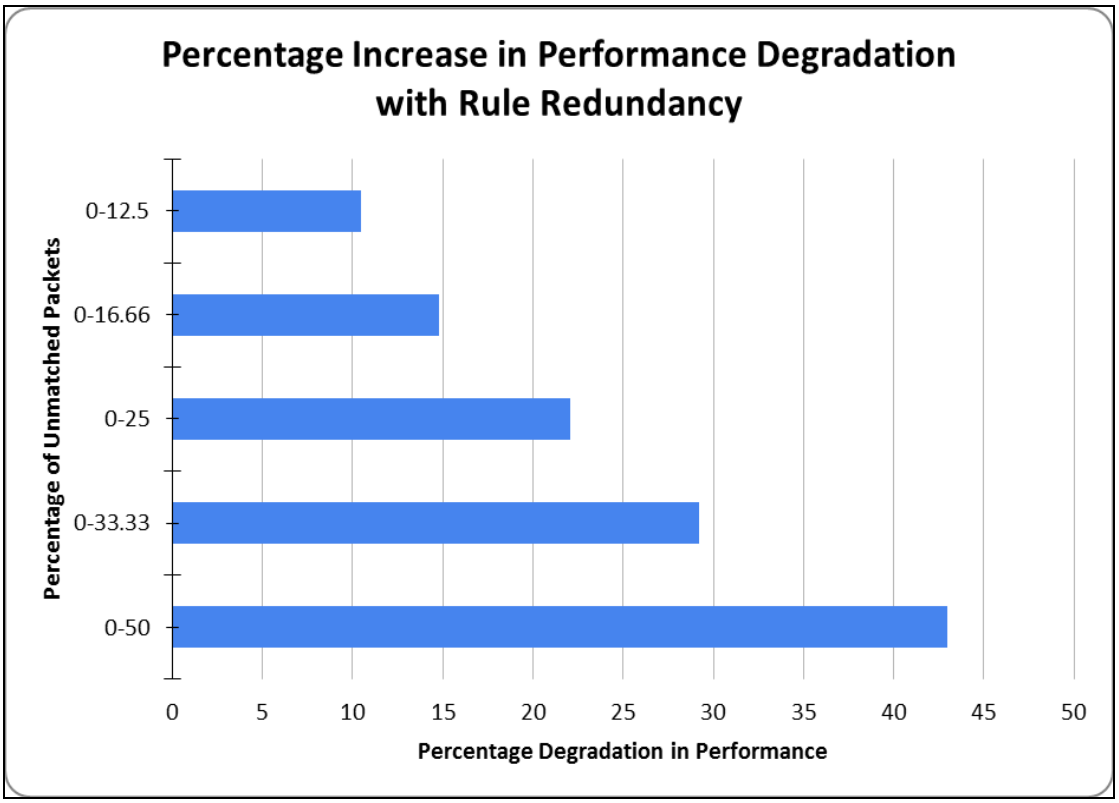

FIGURE 6: Increase in percentage degradation of C_{avg} for non-optimized access list 40.

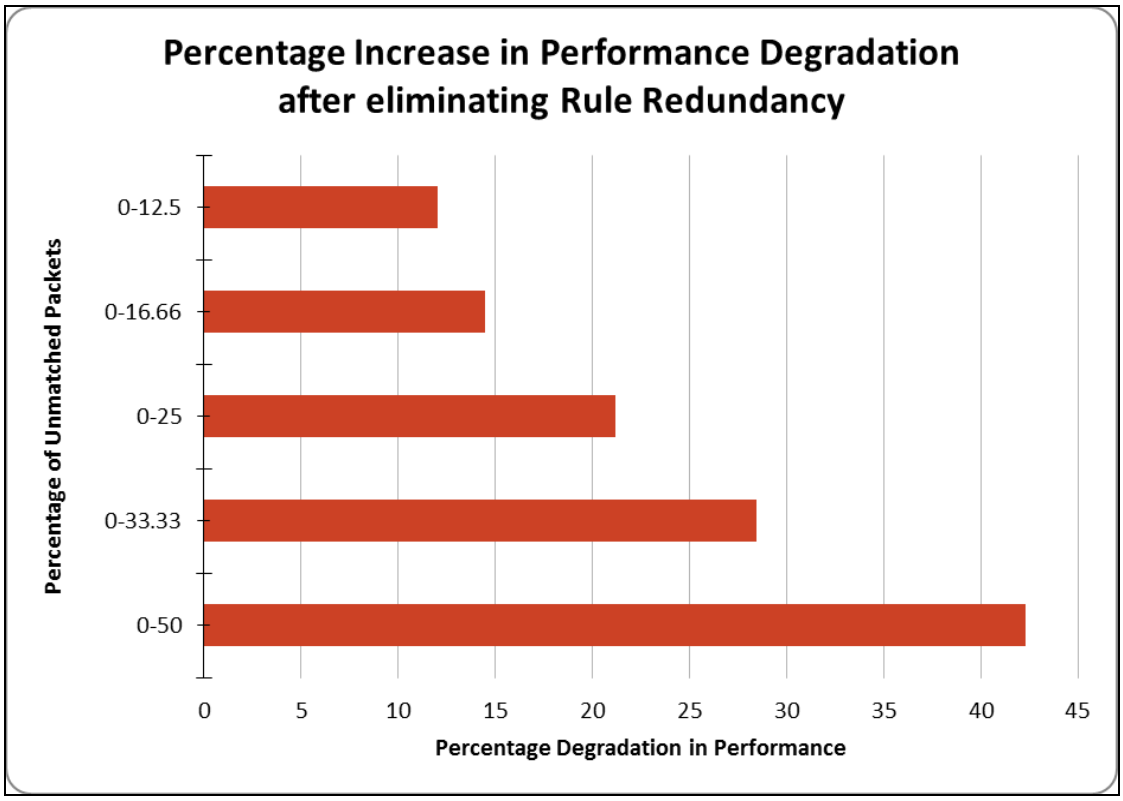

FIGURE 7: Increase in percentage degradation of C_{avg} for optimized access list 41.

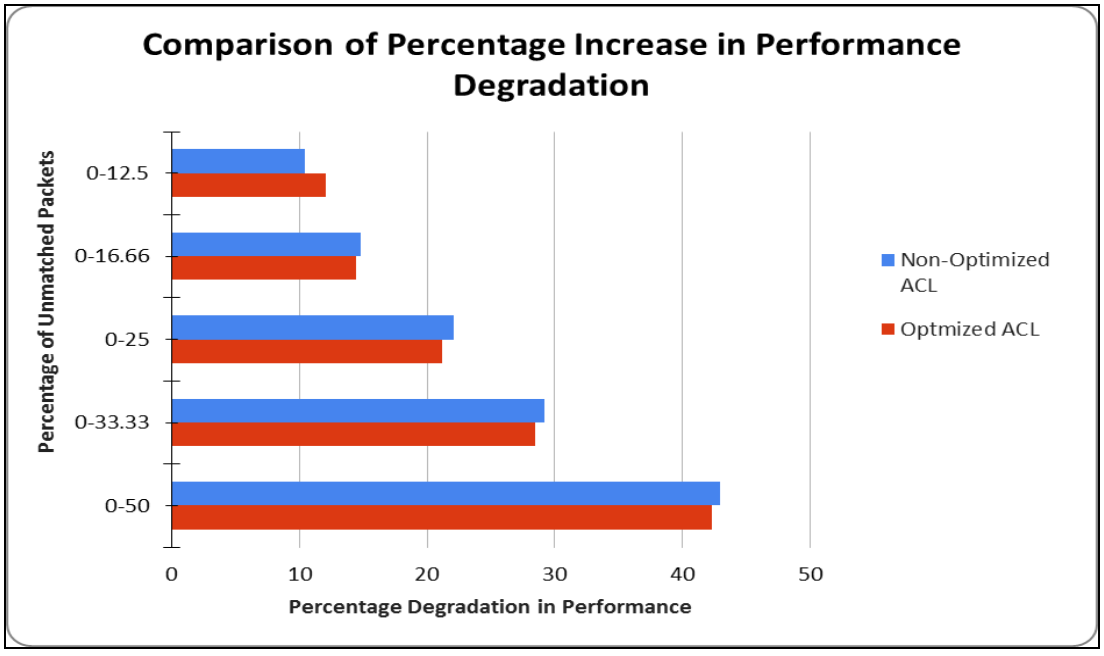

FIGURE 8: Comparison of increase in percentage degradation of C_{avg} for both access lists.

4. ANALYSIS AND DISCUSSION

It is extremely important to improve the performance of existing firewall technologies. This is because the firewall acts as the network's protector and prevents intruders from breaking in. However, the firewall's presence also causes inconvenience to devices within the network because of its negative impact on the network's performance. All packets attempting to enter the network have to go through the firewall. The rate of transmission is slowed because each packet has to wait until packets before it in the queue are matched to a rule in the firewall policy. This problem is further exacerbated by the presence of redundant rules. This is predominantly because in case there is one redundant rule, each packet that enters the network may have to be compared against at least one extra rule which leads to slower transmission.

While this negative impact may not be evident when the number of incoming packets is minimal, it becomes much more apparent when the number of incoming packets is numbering in the millions. Hence, eliminating even one redundant rule will have a visible impact on the firewall's performance because the lesser number of rules leads to lesser number of comparisons. For the sake of simplicity, if we assume that there are hundred rules in the access list which includes two redundant rules and each comparison takes one second, then removing the two redundant rules leads to performance improvement as it takes less time to compare against 98 rules than a hundred rules. In reality however, a comparison will require only a millisecond or even lesser. So the improvement will be estimable when the number of incoming packets is very high.

Also, it must be noted that only packets that match to rules defined after the redundant rules will be affected by their presence. Hence, any performance improvement will have an effect only on those packets. If all the redundant rules are at the end of the access list, keeping or removing them will have absolutely no impact on the firewall's performance.

The immediate observations show unambiguously that the optimized access list 41 defined by implementing the **redundant-insert** algorithm shows a considerably better performance in terms of a considerably reduced value for the average number of comparisons required per packet before a rule is matched.

However, as the percentage of unmatched packets increase, it is noticed that the rate of performance degradation in terms of the average number of comparisons per packet expressed as a percentage value is almost the same for both the optimized and non-optimized access lists which leads to a possible conclusion that the degradation in performance, which is expressed as $\mathbf{\Delta}\ C_{avg}$ is independent of the number of rules defined in the access list.

5. FUTURE WORK

The security of a network plays a huge role in its daily functioning. The firewall is the most fundamental security device in a network. Hence it is crucial to maximize the firewall's performance. There are several aspects of the firewall's functioning that can be given serious thought for research such as rule reordering, redundant rule elimination, impact of burst traffic and rule combination. This will be the subject of our future work.

6. CONCLUSION

Firewalls play an increasingly important role in network security across the world in thousands of enterprise networks today. The amount of packet traffic entering any enterprise network is ever increasing and hence there is considerable incentive in improving firewall performance. In this paper, we proposed an algorithm to identify and eliminate redundant rules in access lists during the configuration phase itself. The results of our implementation strongly indicated that a considerable amount of improvement in firewall performance could be obtained by eliminating redundancy in access control lists.

7. REFERENCES

[1] A. Liu, M. Gowda. “Complete Redundancy Detection in Firewalls." In the proceedings of the 19th annual IFIP WG 11.3 working conference on Data and Applications Security, 2005, pp. 193-206.

[2] H. Ling-Fang. “The Firewall Technology Study of Network Perimeter Security.” In Proceedings of the IEEE Asia-Pacific Services Computing Conference, 2012, pp. 410-413.

[3] L. Zhu, H. Mao and H. Qin. “A case study on Access Control Rules Design and Implementation of Firewall.” In Proceedings of the 8th International Conference on Wireless Communications, Networking and Mobile Computing, 2012, pp. 1-4.

[4] C. Sheth and R. Thakker. “Performance evaluation and Comparative Analysis of Network Firewalls.” In Proceedings of the International Conference on devices and communication, 2011, pp.1-5.

[5] H. Mao, L. Zhu and M. Li. “Current State and Future Development Trend of Firewall Technology.” In Proceedings of the 8th International Conference on Wireless Communications, Networking and Mobile Computing, 2012, pp. 1-4.

[6] M.Z.A Aziz, M.Y Ibrahim, A.M Omar, R.A Rahman, M.M.M Zan, & M.I Yusof. “Performance analysis of application layer firewall.” In Proceedings of the IEEE Symposium on Wireless Technology and Applications (ISWTA), 2012. pp. 182-186.

[7] A. Krishna and A. Victoire. “Simulation of Firewall and Comparative Study.” In Proceedings of the 3rd International conference on Electronics Computer Technology, 2011, pp. 10-14.

[8] T. Lammle. CCNA Routing and Switching Study Guide. Indianapolis, Indiana: Sybex, 2013, pp. 501-528.

[9] I. Mothersole and M. Reed. “Optimizing Rule Order for a Packet Filtering Firewall.” In Proceedings of the Conference on Network and Information Systems Security (SAR-SSI), 2011, pp. 1-6.

[10] H. Hamed, A. El-Atawy & E. Al-Shaer. “Adaptive Statistical Optimization Techniques for Firewall Packet Filtering.” In Proceedings of the 25th IEEE International Conference on Computer Communications, 2006, pp. 1-12.

[11] Z. Trabelsi, L. Zhang & S. Zeidan. “Packet flow histogram to improve firewall efficiency”, In Proceedings of the 8th International Conference on Information, Communication and Signal Processing, 2011, pp. 1-5.

[12] H. Hamed and E. Al-Shaer. “Dynamic Rule-ordering Optimization for High-Speed Firewall Filtering.” In Proceedings of the ACM symposium on Information, computer and communications security, 2006, pp. 332-342.

[13] Z. Trabelsi. Z. Sayed, H.E & Zeidan. “Firewall packet matching optimization using network traffic behavior and packet matching statistics.” In Proceedings of the Third International Conference Communications and Networking (ComNet), 2012, pp. 1-7.

[14] Z. Trabelsi & S. Zeidan. “Multilevel Early Packet Filtering Technique based on Traffic Statistics and Splay Trees for Firewall performance improvement.” In Proceedings of the IEEE International Conference on Communications (ICC), 2012, pp. 1074-1078.

[15] A. Sudarsan, A. Vasu, A. Ganesh, D. Ramalingam and V. Gokul. “Performance Evaluation of Data Structures in implementing Access Control Lists.” International Journal of Computer Networks and Security, vol. 24, issue 2, pp. 1303-1308, 2014.

[16] P. Gupta. “Algorithms for routing lookups and packet classifications.” PhD thesis, Stanford University, 2000.