A Simple Traffic Aware Algorithm To Improve Firewall Performance

Anirudhan Sudarsan *anirudhan.sudarsan@gmail.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

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

Vanaja Gokul *vanaja@svce.ac.in*
Computer Science Department
Sri Venkateswara College of Engineering
Pennalur, 602117, India

Abstract

Firewalls play an extremely important role in today's networks. They are present universally in almost every corporate network across the globe and serve to protect such networks from unauthorized access. The firewall is most commonly implemented as a packet filter. The packet filter works by comparing incoming packets against a set of predefined rules called an access control list (ACL). It is vital to improve the performance of packet filtering firewalls as much as possible. Most of the research work in this area barring a few has not focused on utilizing traffic characteristics to improve the performance of packet filters. In this paper, we propose a simple algorithm that exploits traffic behavior by utilizing incoming traffic statistics to dynamically modify rule ordering in access control lists. Hence repeated packets or multiple packets from the same source require lesser number of comparisons before a rule is matched. When testing was performed for the proposed work using both a simulated firewall and simulated traffic the performance of the firewall showed considerable improvement.

Keywords: Firewall, Packet Filter, Access Control List, Rule Ordering, Traffic Characteristics.

1. INTRODUCTION

The need for firewall arises due to the inefficiencies of encryption algorithms when it comes to protecting the trusted internal network from malicious packets. This is due to the fact that packets can be forwarded into the network whether or not they are encrypted. A firewall can be implemented as a separate device, a software or combination of both [1]. The firewall secures the trusted network by controlling access to its resources. It scrutinizes incoming and outgoing packets and compares their structure against a set of predefined rules called the access control list. The packet may then be dropped or permitted based on which rule it maps onto. Every packet that attempts to enter or leave the network has to pass through the firewall. It can be said that the firewall acts as a gateway of the network [2 and 3].

We can consider the packet to be a structure with a set number of attributes such as source port, destination port, source IP address and destination IP address. The firewall's configuration will determine the decision to be taken for each individual packet [2]. The firewall decision takes the form of two possible actions- permit or deny i.e. the packets are either routed into the network or filtered at the firewall's interface itself.

2. SUMMARY OF COMMON FIREWALL TECHNOLOGIES

The firewall's most common form is the stateless packet filter. The stateless packet filter considers each incoming packet as an individual entity and decides whether or not to forward the packet based on its characteristics/ attributes only. It does not take into account any data about traffic history as it does not store connection state data [1]. Sometimes, the packet firewall is integrated into the router itself [4]. Stateless packet filter are susceptible to some forms of attack. For instance, they cannot detect spoofed packets [3].

A stateful packet filter keeps track of network connections. When an incoming packet is received on its interface, the stateful packet filter scans the packet to determine whether it is a part of an existing connection state or is a request for a new connection [1]. A connection request will usually take the form of a SYN packet- the first step in the three way handshake process. If neither of the two criteria is met, the packet is dropped. Stateful packet filter works on the assumption that packets from the same source need not be examined repeatedly as long as they belong to an existing connection. The stateful packet filter is considerably more efficient than its stateless counterpart from viewpoint of performance as not all incoming packets need to be compared against rules defined in the access control list. It provides a stronger level of security and is easy to configure [5]. The implementation complexity is however greater when compared to a stateless packet filter [4].

Firewalls are also implemented as application level gateways (ALGs) which as the name suggests function at the application layer. These third generation firewall architectures are also called proxy servers. It acts as an intermediary between the client and the server. Hence the server views the ALG as the client and the client views ALG as the server. ALGs are capable of detecting malicious code and viruses as they scrutinize the application layer format in the packet. They provide a higher level of security, logging services and end to end encryption. The implementation complexity is however greater which leads to a considerably slower performance [1, 5, 6, 7]

Lastly, firewalls are also employed as circuit level gateways which operate at the transport layer. They examine the contents of both the layer 3 and layer 4 headers to determine whether or not to permit the packets. When combined with a regular packet filter, it is termed as a dynamic packet filter. It observes and validates the formation of a TCP connection by observing the three way handshake process [6].

Firewalls still have a few disadvantages despite rapid technological growth. It cannot prevent some forms of attack such as those perpetrated by those within the network itself [7]. The firewall also becomes ineffective if an unauthorized user has already gained access to the network's resources. Hence there is a unyielding requirement to implement additional security measures such as encryption [6]. Another disadvantage of using a firewall as the entry and exit point of the network is the fact that it can potentially cause a bottleneck effect and also lead to a single point of failure. As the network increases in size or if there is an increase in traffic volume, the load on the firewall also increases [8].

Some of the commonly available firewalls in the market include the Checkpoint SPLAT, Cisco's adaptive security appliance and the OpenBSD packet filter all of which do an excellent job as the watchman of the network. Performance analysis of these three firewalls in a lab environment has indicated that the Cisco ASA exhibits a better performance for TCP, UDP and HTTP throughput. The BSD permitted more number of concurrent connections and connections per second. The increasing number of regulations has led to a pressing requirement to improve firewall technologies [9].

3. FUNCTIONING OF A STATELESS PACKET FILTER

As stated previously, a stateless packet filter works by examining each incoming packet and comparing its structure against a set of predefined rules called an access control list (ACL). The rule in the ACL corresponding to the incoming packet will determine whether or not the packet will be forwarded into the network. There are two types of ACL's that can be used to define the policy of the firewall-

a) Standard access control lists
b) Extended access control lists

First we define a simplified structure for the packet to illustrate the working of ACLs. The simplified structure contains the following attributes- source IP address, destination IP address, source port number and destination port number. The actual packet will contain several other fields such as time to live and header length but such attributes are ignored here as they do not play any significant role in the functioning of a stateless packet filter. The following is an illustration of the packet's structure-

```
Struct packet {
Source IP address
Destination IP address
Source port
Destination port
}
```

The standard access list uses only one of the attributes in the above illustration- source IP address- to decide whether or not to forward the packet into the network. The source IP address of the incoming packet is compared with the rules in the standard access list sequentially until a match occurs or no more rules are left to compare with. In the former case, the corresponding action is taken on the packet- it is either permitted or dropped. In the latter case however, the implicit deny takes over [10].

The extended access list uses all the attributes defined in the packets structure to arrive at a decision. The working of the extended ACL is mostly identical to that of the standard ACL. The only difference between standard ACLs and extended ACLs is that extended ACLs compare several attributes while the standard ACL compares only the source IP address [10]. For the purposes of this paper, we consider a standard access control list.

The command format for defining a standard access list or access control list based on a Cisco router is as follows:

access-list [access-list number 1-99] [permits or deny] [source IP address] [wild card mask]

After the access control list has been defined, it must be applied to an interface on the firewall or the router (which can play the role of a packet filter) in one of two possible directions- inbound or outbound before it can be considered functional. Usually only one access list can be applied per interface per direction [10]. When the ACL is applied in the outbound direction, the packet is forwarded to the interface where the ACL has been defined and only then compared against the rules defined. When the ACL has been applied in the inbound direction, then the packet is first compared against the defined rules before being routed through or dropped. The following is an example of a smaller than regular ACL that has been defined as per the aforementioned format.

1. access-list 25 permit 184.29.6.54 0.0.0.0
2. access-list 25 deny 184.29.0.0 0.0.255.255
3. access-list 25 permit 161.34.6.0 0.0.0.255
4. access-list 25 deny 161.34.0.0 0.0.7.255
5. access-list 25 permit 26.212.36.4 0.0.0.0
6. access-list 25 permit 17.39.112.0 0.0.0.255

Access list 25 defined above works as follows,

a. Packets from the host address 184.29.6.54 are permitted but packets from the rest of the 184.29.0.0 /16 network are denied.
b. Packets from the 161.34.6.0 /24 network are permitted but packets from the remaining addresses in the 161.34.0.0 /21 network are denied.
c. Packets from the host address *26.212.36.4* are permitted.
d. Packets from the 17.39.112.0 /24 network are permitted.
e. An implicit deny is enforced by default at the end of the access list.

Firewall rule ordering is an important area of research and has been the subject of a few noteworthy papers recently. Optimizing rule order leads to a better performance, which is what has been attempted in this paper. However, rule ordering in access control lists cannot be modified indiscriminately. If rule order in an ACL is not altered correctly, it will lead to incorrect functioning of the firewall. This is because packet structure is compared against ACL rules sequentially. The following can be considered the necessary and sufficient condition for successfully reordering rules in an ACL- *The rules in the access control list of a firewall F is considered to be reordered correctly, if the functioning of the firewall F remains the same before and after reordering is performed.*

This functioning of a firewall is altered when rule order is changed due to the presence of related rules. Related rules mean that a packet could match to more than one of the defined rules. In the case of the access list 25 a packet from 184.29.6.254 could match to both rules 1 and 2 in the access list. The order of two related rules is important only if the action of the rules differs. In the case of a packet, from 184.29.6.53, the actions of rule 1 and 2 (related rules) differ. The term dependent is used to refer two rules that have a relationship where it is necessary to maintain the order to comply with the security policy and avoid conflict [8].

The ACL example given above has been redefined after it is reordered by interchanging rules 1, 2 and rules 3, 4. For all purposes, the intent behind defining the access list remains the same. The modified access control list is shown below:

1. access-list 35 deny 184.29.0.0 0.0.255.255
2. access-list 35 permit 184.29.6.54 0.0.0.0
3. access-list 35 deny 161.34.0.0 0.0. 7.255
4. access-list 35 permit 161.34.6.0 0.0.0.255
5. access-list 35 permit 26.212.36.4 0.0.0.0
6. access-list 35 permit 17.39.112.0 0.0.0.255

The reordered access control list functions as follows,

a. All packets from the 184.29.0.0 /16 network are denied.
b. All packets from the 161.34.0.0 /21 network are denied.
c. Packets from the host address *26.212.36.4* are permitted.
d. Packets from the 17.39.112.0 /24 network are permitted.
e. An implicit deny is enforced by default at the end of the access list.

The intent behind defining the reordered ACL is still the same as the one behind defining the original ACL. But, access list 35 does not permit packets from the host 184.29.6.54 while packets from the same host are permitted by access list 25. This is because, when access list 25 is applied, the packet's attributes are first compared against the rule *access-list 25 permit 184.29.6.54 0.0.0.0* which permits the packet into the network. However, when access list 35 is applied, the packet's attributes are first compared against the rule *access-list 35 deny 184.29.0.0 0.0.255.255* which denies all packets from the 184.29.0.0 /16 network which by extension includes the host 184.29.6.53.

The very same logic is also applicable to packets arriving from the network 161.34.6.0 /24. When access list 25 is applied packets from this network are permitted to enter the trusted network because the attributes of any packet from this network are first compared with the rule *access-list 25 permit 161.34.6.0 0.0.0.255* before the rule *access-list 25 deny 161.34.0.0 0.0.7.255* which therefore permits the packet into the network. However when access list 35 is applied, the packet is first compared with the rule *access-list 35 deny 161.34.0.0 0.0. 7.255* before the rule *access-list 35 permit 161.34.6.0 0.0.0.255,* hence causing packets from the 161.24.6.0 /24 network to be dropped.

4. RELATED WORK

The motivation for optimizing firewall performance comes from the fact that the rule sets in firewalls can become considerably large when there is a combination of complex user requirements and diverse networked applications. The packet matching process is much more complex than a routing table lookup process as the rules perform searches over many fields in the packet and may also record state information. A large rule set can hence have a detrimental effect on the performance of the firewall [8].

A good amount of research related to firewall performance optimization has been undertaken recently. The stateless packet filter compares the attributes of the packet against the rules in the access list sequentially. This is inefficient as the worst case time complexity will be proportional to the number of rules in the ACL. This makes the implementation less scalable. Many of the proposed methods include specialized data structures and even hardware based solutions. Hardware based methods use content addressable memories (CAM) to exploit parallelism in the hardware to match multiple rules in parallel. But, this method is limited to smaller firewall policies due to the power, size and cost limitations of CAM [11].

In [12] a "Firewall compressor" algorithm is proposed to optimize performance. This algorithm works to minimize the overall size of the firewall policy by reducing the number of rules in the rule set. This minimization is achieved by analyzing the rules in terms of the search space they cover after which new rules are framed to cover the same search space. This usually results in many of the original rules being combined to produce fewer rules.

According to [8], the firewall optimization problem is to reorder the rules in such a way that the more frequently used rules are near the top of the rule set which therefore leads to an improvement in performance. Hence, rules are associated with a weight that equals the number of matches of these rules for a representative flow of traffic. Rule dependencies are also factored in when reordering the rules. The algorithm initially operates on unoptimized rule set which contains rules associated with a weight equal to proportion of packet matches. This initial list is used to create a heap which contains rules sorted in the order of rule weight while disregarding rule dependencies. The algorithm then creates another list which is initially empty and fills it with rules as the algorithm executes. The algorithm executes as long as there are items in the heap that need to be processed and when it finishes executing, this list contains the optimized ruleset.

[13] proposes a method to improve firewall packet filtering time by optimizing the order of security policy filtering fields for early packet rejection. The filtering fields are optimized based on traffic statistics. This method provides protection against denial of service (DOS) attacks that target the

default rule. Early packet acceptance is achieved by using a splay tree data structure which adjusts dynamically based on traffic flows which leads to a reduced value of matching time as repeated packets require lesser number of memory accesses. The proposed algorithm consists of a set of statistical splaying filters that use binary search on prefix length and is called: Statistical splaying filters with binary search on prefix length. This technique uses three levels of filtering to reject unwanted traffic at the earliest,

1) *Statistical policy filtering level* for early packet rejection. At this level, for a given window of traffic policy fields are arranged in descending order starting with the filed with highest rejection rate.
2) *Field filtering level* for early packet rejection and acceptance. In this level, each filtering field consists of a collection of hash-tables and a splay tree.
3) *Cascaded filtering level* for early packet rejection. In this level, list of matched field rules is intersected with previously intersected matched rule list. If there are no common rules between the lists, the packet will be rejected as early as possible.

The three filtering levels are combined together to enhance packet processing time.

[14] presents a method based on histograms of packet filtering to predict packet filtering patterns in terms or rule and rule fields order. The mechanism is even more significant when the firewall is loaded with burst traffic. A method is proposed to optimize the early acceptance path as well as early rejection path using histograms of both packet matching rule and packet not matching rule fields. The algorithm calculates the histograms in terms of packet matching and non-packet matching probabilities on a real time segment basis.

In [15], a method that segments traffic space is proposed. The traffic space is first segmented and the matching rate for each rule is calculated. The statistics, mean and variance are then deduced for a predefined window of segments. The means and variances are used to update the positions of the filtering rules in the security policy. The first calculated value is the matching rate which is the percentage of packets that matched a particular rule R_i. Multiple windows of segments are used to take into account the effect of past network statistics. Based on the window size and the number of packets, the match ratio is then calculated after which the rules are dynamically ordered based on a matching rate coefficient.

In [11], a method to perform early rejection of unwanted flows without impacting other traffic flows is proposed. This method uses adaptive statistical search trees to utilize important traffic features and minimize the average packet matching time. The statistical properties of traffic passing through the firewall are considered and used for building a search tree that gives near optimum search time. The constructed trees for each field are combined to create an optimal statistical matching tree of all rules in the policy. An adaptive alphabetic tree is used to dynamically insert the most frequently used field values at the shortest path in the search tree leading to significant matching reduction for the most popular traffic. The alphabetic tree is reconstructed periodically to match the most recent traffic features.

[16] proposes a method of using internet traffic characteristics to optimize firewall filtering policies. This technique utilizes some calculated statistics to adjust to the present traffic conditions by dynamically optimizing the ordering of the rules in the firewall. However, this work does not use statistics related to previous traffic flows.

In [17], two techniques are proposed. The first one is termed Segment-based tree search (STS). This technique uses bounded Huffman trees and segmented traffic to improve the performance by using statistics learnt from the segments. But, this scheme has the disadvantage of having a large overhead associated with it for maintaining the tree. The second technique called Segments-based list search attempts eliminates this overhead by keeping the segments in a MRU (most recently used) order. This technique can be used when packet traffic is steady.

[18] proposes a method to eliminate redundancies in an access control list as redundancies can lead to degradation in performance. This paper considers two types of redundant rules- forward and backward. This paper considers the access control list as a linked list data structure and implements a mechanism to eliminate nodes that correspond to redundant rules. The proposed mechanism is simulated and compared with the traditional static method and the results indicate that considerable performance improvement can be achieved.

Hereon, we use the term access lists to refer to access control lists unless mentioned otherwise.

5. IMPLEMENTATION

During the implementation, the access list was implemented using a singly linked list data structure for the purpose of ease of implementation even though theoretically, a splay tree or a height balanced tree data structure would give better results [19]. Every node in the linked list corresponds to one rule in the access list. When an incoming packet enters the interface where the access list is applied, its structure is compared against the rules in the access list by comparing the attributes of the packet against the corresponding attributes of each rule in the firewall sequentially until there is a match or the end of the list is reached. The aim here is to reduce the average number of comparisons required before a rules is matched by reordering rules based on traffic characteristics.

This algorithm works on an existing access list by keeping track of the incoming packets and reordering the rules based on recent traffic history. The algorithm can be implemented internally in the firewall to execute continuously as long as incoming packets are being received on the firewall's interface. The working of this algorithm is transparent from the view point of firewall administrators.

5.1 Proposed Algorithm

We define the following data types, functions and structures for our algorithm.

List- A standard access list defined as a linked list with m nodes each representing one of the rules in the list.
Packet- A structure which represents a packet
Rule- A node in the linked list **List** which corresponds to one of the rules in the access list
Integer count- Counter value indicating the number of packets received since the last time the access list was reordered. Count reaches a maximum threshold value n after being incremented each time the algorithm is executed. This maximum value represents the size of the window of packets whose characteristics are used to dynamically reorder the access list.
Reorder (**List** L) – Function that dynamically reorders the access list **List** L based on recent traffic characteristics.
Update () –Function that keeps track of recent traffic characteristics.
Compare (**Packet** P, **Rule** R) - Function that compares the attributes of the incoming **Packet** P with the attributes of **Rule** R.
Perform-Action (**Rule** R) - Function that performs the action corresponding to **Rule** R. It then sets FLAG=1;

Initially **List** L has m rules and the count value is initialized to 0 and FLAG=0. The structure of the **Packet** and structure of a **Rule** in the **List** L are illustrated below.

Struct **Packet** {
Source port number;
Source IP address;
Destination port number;
Destination IP address;
}

```
Struct Rule {
Source IP address range;
Wildcard Mask;
Action;
}

Algorithm dynamic-reorder (Packet Pi) {
For (every Rule R in List L) {
        Compare (Pi, R);
        If Rule R matches with Packet Pi {
                Perform-Action (R);
                Break;
        } //end of if
        Else continue;
} //end of Loop
If FLAG is equal to 0 Perform-Action (Deny);
Count++;
Update ();
If count equals threshold value {
        Reorder (L);
        Count=0;
} //end of If
FLAG=0;
} //end of Algorithm dynamic-reorder
```

The algorithm dynamic-reorder takes as input the **Packet** P_i. The attributes of the packet are then compared with the attributes of every **Rule** R in **List** L. If there is a match, then the corresponding action is performed with the **Perform-Action** function. If none of the rules in the **List** L match the incoming **Packet** P_i, then the packet is dropped by implicit deny. The counter Count is then incremented and the **Update** function is executed. The Update function performs the necessary operations and keeps track of recent traffic characteristics. Once Count reaches or exceeds a particular threshold value the order of rules in **List** L is changed by reordering the Rule nodes to reflect traffic characteristics of last window of packets. After the **Reorder** function finishes execution, the rules at the beginning of the access list will correspond to packets that were more commonly seen. The following access list is defined to provide a practical illustration of the algorithm.

Rule Number	Source IP	Wildcard Mask	Action
1	15.16.142.3	0.0.0.0	Deny
2	19.24.0.0	0.0.255.255	Deny
3	26.0.0.0	0.255.255.255	Permit
4	53.130.46.0	0.0.0.127	Deny
5	74.46..154.96	0.0.0.31	Permit
6	157.64.0.0	0.0.63.255	Deny
7	167.23.45.48	0.0.0.15	Permit
8	194.201.169.171	0.0.0.0	Deny
9	68.120.0.0	0.0.255.255	Permit
10	180.45.64.0	0.0.31.255	Deny

FIGURE 1: Sample Access List- Initial.

FIGURE 2 represents this access list as a singly linked list. The numbering of the nodes corresponds to the rules in the access list.

FIGURE 2: Access list as a linked list.

The reasoning behind the algorithm is that, if there are n rules in the access list, the distribution of incoming packets will not be even i.e. it is incorrect to assume that percentage of incoming packets corresponding to each rule will be 100/n. In real world scenarios, the distribution of incoming packets will be uneven and in some cases extremely biased towards a few particular rules. We define the following incoming packet distribution for the above access list.

Rule Number	Percentage of Incoming Packets
1	5
2	6
3	7
4	4
5	12
6	11
7	13
8	20
9	10
10	12

FIGURE 3: Incoming Packet distribution for sample access list.

If the size of the window was assumed to n and x packets were to be generated according to the distribution in FIGURE 3 where x >> n, then for each incoming **Packet** P_i, the **dynamic-reorder** algorithm compares the packet's attributes against the attributes of each rule in FIGURE 1 and FIGURE 2 (both represent same list) until a match occurs. The corresponding action is performed by the **perform-action** function after which the update function updates the internal database based on incoming packet's characteristics. Finally, once the number of incoming packets exceeds the window size, the Reorder function executes and dynamically changes the ordering of the nodes in the list and by extension the ordering of the rules. The access list in FIGURE 1 will be modified as shown in FIGURE 4 after **dynamic-reorder** finishes executing. The rule numbering has been retained from FIGURE 1 for ease of understanding. This should in theory lead to a reduced value for average number of comparisons before rule match.

Rule Number	Source IP	Wildcard Mask	Action
8	194.201.169.171	0.0.0.0	Deny
7	167.23.45.48	0.0.0.15	Permit
10	180.45.64.0	0.0.31.255	Deny
5	74.46..154.96	0.0.0.31	Permit
6	157.64.0.0	0.0.63.255	Deny
9	68.120.0.0	0.0.255.255	Permit
3	26.0.0.0	0.255.255.255	Permit
2	19.24.0.0	0.0.255.255	Deny
1	15.16.142.3	0.0.0.0	Deny
4	53.130.46.0	0.0.0.127	Deny

FIGURE 4: Sample Access list after being modified by **dynamic-reorder** algorithm.

5.2 Testing Setup and Process

An access list with ten rules was defined for the implementation phase. All these rules were defined to be independent of each other i.e. the correct functioning of the firewall is not affected by the ordering of the rules. The access list is shown in FIGURE 4. Both C and Java programming were used for implementing this algorithm. We simulated the generation of 200000 packets. These 200000 packets were divided into 4 groups of 50000 packets with each group corresponding to one of the following scenarios expressed in FIGURE 4. Scenario 1 is applicable for the first 50000 packets, scenario 2 for the next 50000 packets and so on.

Rule Number	Source IP	Wildcard Mask	Action
1	21.234.65.1	0.0.0.0	Permit
2	33.65.78.80	0.0.0.15	Permit
3	96.42.32.0	0.0.31.255	Deny
4	74.161.201.128	0.0.0.31	Permit
5	112.63.192.0	0.0.63.255	Deny
6	86.24.0.0	0.0.255.255	Permit
7	175.0.0.0	0.255.255.255	Deny
8	201.14.0.0	0.0.255.255	Deny
9	66.78.196.5	0.0.0.0	Permit
10	146.57.44.128	0.0.0.127	Permit

FIGURE 5: Access list defined for testing.

Scenario 1		Scenario 2		Scenario 3		Scenario 4	
Rule Number	Packet Distribution	Rule Number	Packet Distribution	Rule Number	Packet Distribution	Rule Number	Packet Distribution
1	5%	1	8%	1	12%	1	6%
2	12%	2	5%	2	6%	2	7%
3	4%	3	15%	3	7%	3	12%
4	9%	4	9%	4	8%	4	9%
5	6%	5	5%	5	9%	5	3%
6	14%	6	6%	6	10%	6	11%
7	11%	7	8%	7	10%	7	12%
8	8%	8	12%	8	20%	8	14%
9	16%	9	20%	9	12%	9	9%
10	15%	10	12%	10	6%	10	17%

FIGURE 6: Packet distribution for the four scenarios. .

The average number of comparisons required per packet before a rule was matched was calculated for the above four scenarios both with and without using the **dynamic-reorder** algorithm. For the **dynamic-reorder** algorithm, the windows size was considered to be 5000 packets.

For both the cases, the average number of comparisons per packet was first calculated and then averaged out over 10000 iterations of the simulation. The comparison is performed between two simulated firewalls- one that does not implement any optimization technique and one that employs the **dynamic-reorder** algorithm. The following formula was defined to calculate the average number of comparisons.

Average number of comparisons per packet= Total number of comparisons / Total number of packets.

The average number of comparisons is represented symbolically as C_{avg}

To reduce implementation complexities and difficulties, the following assumptions were made,

1) There is no rule dependency in the access list
2) None of the incoming packets go unmatched i.e. each incoming packet is matched successfully to at least one rule.

6. RESULTS

The following results were observed after the implementation,

a) When the C_{avg} value was computed without implementing the **dynamic-reorder** algorithm it was found to be 6.095 after all 200000 packets were generated.
b) When the C_{avg} value was computed after implementing the **dynamic-reorder** algorithm it was found to be 4.49 after all 200000 packets were generated.

The ordering of rules after each set of 50000 packets is shown in FIGURE 6.

After 50000 packets

Rule Number	Source IP	Wildcard Mask	Action
1	66.78.196.5	0.0.0.0	Permit
2	146.57.44.128	0.0.0.127	Permit
3	86.24.0.0	0.0.255.255	Permit
4	33.65.78.80	0.0.0.15	Permit
5	175.0.0.0	0.255.255.255	Deny
6	74.161.201.128	0.0.0.31	Permit
7	201.14.0.0	0.0.255.255	Deny
8	112.63.192.0	0.0.63.255	Deny
9	21.234.65.1	0.0.0.0	Permit
10	96.42.32.0	0.0.31.255	Deny

After 100000 packets

Rule Number	Source IP	Wildcard Mask	Action
1	66.78.196.5	0.0.0.0	Permit
2	96.42.32.0	0.0.31.255	Deny
3	201.14.0.0	0.0.255.255	Deny
4	146.57.44.128	0.0.0.127	Permit
5	74.161.201.128	0.0.0.31	Permit
6	21.234.65.1	0.0.0.0	Permit
7	175.0.0.0	0.255.255.255	Deny
8	86.24.0.0	0.0.255.255	Permit
9	33.65.78.80	0.0.0.15	Permit
10	112.63.192.0	0.0.63.255	Deny

After 150000 packets

Rule Number	Source IP	Wildcard Mask	Action
1	201.14.0.0	0.0.255.255	Deny
2	21.234.65.1	0.0.0.0	Permit
3	66.78.196.5	0.0.0.0	Permit
4	86.24.0.0	0.0.255.255	Permit
5	175.0.0.0	0.255.255.255	Deny
6	112.63.192.0	0.0.63.255	Deny
7	74.161.201.128	0.0.0.31	Permit
8	96.42.32.0	0.0.31.255	Deny
9	33.65.78.80	0.0.0.15	Permit
10	146.57.44.128	0.0.0.127	Permit

After 200000 packets

Rule Number	Source IP	Wildcard Mask	Action
1	146.57.44.128	0.0.0.127	Permit
2	201.14.0.0	0.0.255.255	Deny
3	96.42.32.0	0.0.31.255	Deny
4	175.0.0.0	0.255.255.255	Deny
5	86.24.0.0	0.0.255.255	Permit
6	74.161.201.128	0.0.0.31	Permit
7	66.78.196.5	0.0.0.0	Permit
8	33.65.78.80	0.0.0.15	Permit
9	21.234.65.1	0.0.0.0	Permit
10	112.63.192.0	0.0.63.255	Deny

FIGURE 7: Order of rules in the access list after each window of packets were generated.

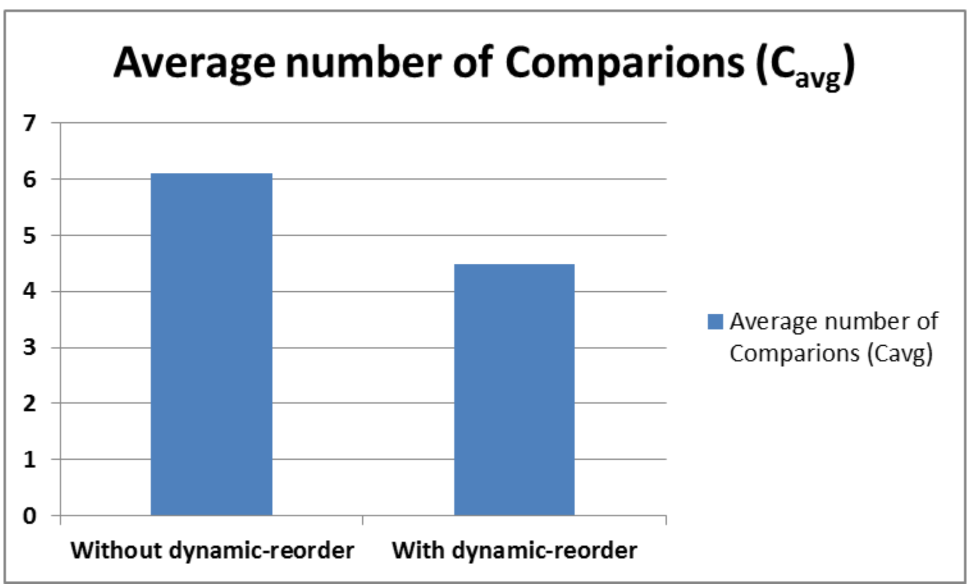

FIGURE 8: Comparison of C_{avg} before and after implementing dynamic-reorder algorithm.

7. ANALYSIS AND DISCUSSION

The importance of improving firewall performance cannot be understated. The firewall is the guard of the network and protects it from intruders. However, its presence also causes some inconvenience to hosts within the trusted network. This is caused because of the negative impact of the firewall on the overall performance of the network. The presence of the firewall implies that all packets entering and leaving the network has to pass through it. Each packet is queued up in the firewall's buffer until it has been matched to a rule in the access control list thus slowing down the rate of transmission. This effectively means that the bandwidth of the network is not being utilized to its fullest.

The process of matching the packet with a rule in the access list is in effect an overhead that needs to be reduced. While research in this area may not lead to substantial reductions in the C_{avg} value, even a small reduction can have a great impact on the performance of the network. This gain in performance can be realized if we bring into perspective that most corporate networks have millions of packets traversing through their infrastructure at any point of time. Thus, even a minor improvement in this criterion can lead to a better overall performance of the network. For example, if we assume that there are a million packets in the network and each comparison takes one hundredths of a second, then a reduction in the C_{avg} by a value of one, leads to an overall decrease of the number of comparisons by one million and the total time taken reduces by about five and a half hours. That is a substantial improvement which can be perceptible to users of the network.

After the implementation was completed successfully, the immediate observations clearly indicated that the C_{avg} value when **dynamic-reorder** algorithm was not implemented was about 35% higher than when **dynamic-reorder** was implemented. The C_{avg} value when dynamic-reorder is applied is about 73% of the value when it is not applied. Hence, a considerable improvement in performance is obtained when the dynamic-reorder algorithm is implemented compared to the case where only a static approach is used.

8. FUTURE WORK

There is a lot of scope for undertaking further research in the field of network security especially in firewalls and VPNs. There is a pressing requirement to improve the performance of the firewall. There are several aspects that can be given serious consideration for research such as removing rule redundancy, reducing impact of burst traffic and reducing the number of rules in firewall policies by combining two or more of them. Also, in this paper the results were obtained through continually running simulations in a lab environment. It would be of interest to us to test our algorithm in a production network and verify its impact on the performance. Such research will be the subject of our future work.

9. CONCLUSION

Firewalls play an increasingly important role in modern day networks across the world. Hence there is a strong motivation to improve firewall performance by optimizing rule order to reduce the average number of comparisons required before a rule is matched successfully and by extension, reduce the time required. This motivation arises due to the fact that traffic distribution is not uniform. Static ordering of firewall rules does not take packet traffic characteristics into account. A good amount of research has been done in this area recently and this paper attempts to add to this. In this paper, we proposed a method to dynamically reorder rule in an access list to improve firewall performance. The results based on the simulation of our algorithm clearly indicated that considerable performance improvement could be obtained by implementing the proposed **dynamic-reorder** algorithm.

10. REFERENCES

[1] 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.

[2] A. Liu, M. Gowda. “Complete Redundancy Detection in Firewalls.” Lecture Notes in Computer Science, Vol. 3654, pp 193-206, 2005.

[3] K. Scarfone and P. Hoffman. (2009) “Guidelines on Firewalls and Firewall Policy.” U.S.A.: National Institute of Standards and Technology.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

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

[11] 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.

[12] A.X. Liu, E. Torng, and C. R. Meiners. “Firewall compressor: An algorithm for minimizing firewall policies.” In Proceedings of the 27th Conference on Computer Communications, 2008, pp. 176–180.

[13] 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.

[14] 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.

[15] Z. Trabelsi, H. El Sayed & 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.

[16] H. Hamed, A. El-Atawy & E. Al-Shaer. “On Dynamic Optimization of Packet Matching in High-Speed Firewalls.” IEEE Journal on Selected Areas in Communications, vol. 24, issue 10, pp. 1817-1830, 2006.

[17] El-Atawy A, Samak T, Al-Shaer.E & Hong Li. “Using online traffic statistical matching for optimizing packet filtering performance.” In Proceedings of the 26th IEEE International Conference on Computer Communications, 2007, pp 866-874.

[18] A. Vasu, A. Ganesh, P. Ayyappan and A. Sudarsan. "Improving Firewall Performance by Eliminating Redundancies in Access Control Lists." International Journal of Computer Networks, vol. 6, issue 5, pp. 92-107, 2014.

[19] 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.