Design and Implementation of Packet Filter Firewall using Binary Decision Diagram

Gopal Paul[†], Amaresh Pothnal[#], C. R. Mandal[†], Bhargab B. Bhattacharya[§]
[†]Dept. of Computer Science and Engineering, IIT Kharagpur, INDIA
[#]Oracle Corporation, Bangalore, Karnataka, INDIA
[§]ACM UNIT, Indian Statistical Institute, Kolkata, INDIA
gpaulcal@yahoo.com, amaresh.pothnal@gmail.com, chitta@iitkgp.ac.in, bhargab@isical.ac.in

Abstract **– Packet filtering is the one of the major contemporary firewall design techniques. An important design goal is to arrive at the decision at the packet only. Implementation of such packet filter using Binary Decision Diagram (BDD) gives more advantages in terms of memory usage and look up time. In the case of the list-based packet filter firewall where rules are checked one by one for each incoming packet, the time taken to decide on a packet is proportional to the number of rules. The performance is improved with rule promotion but that itself a slow procedure. In this work we present a BDD-based approach which gives much better result in terms of number of comparisons or accesses the rule list make. Results on 1 million packets show that for most-accept packets, on an average, 75% reduction happens in such comparisons when BDD-based approach is used over list-based with promotion approach. For most-reject packets this reduction is nearly 34%.**

I. INTRODUCTION

A. Firewall

With increasing reliance on the Internet, security remains a top concern due to the increase in number of potential sources of attack networks require protection from unintentional incidents and malicious acts. Any new local area network is connected to the Internet hosts present in that network become vulnerable to the attack. Hence, to overcome this security problem, these systems is configured carefully so that the design of the network systems becomes robust. Most of the complexities in using firewalls today lie in managing a large number of firewalls and ensure whether they enforce a consistent policy across an organization's network [6].

A.1 Types of Firewalls

There are mainly three types of firewall are commonly used. In this section we explain each of them in brief.

A.1.1 Application-level Firewall

Application-level firewalls or proxies work at the application layer and are protocol specific [7]. Each protocol requires a separate proxy. HTTP proxies can, for example, filter out certain HTTP commands such as 'PUT'. These can be used for logging user activities. Such firewalls are CPU intensive but provide good security [1].

A.1.2 Circuit-level Firewall

Circuit level firewall works in transport layer of the protocol stack and provides more general type of security. This acts as proxy for TCP/IP application by monitoring the incoming and outgoing packets. Disadvantage of this approach is that many packets can go with undetected due to more general things in consideration of filtering the packets.

A.1.3 Packet-filter Firewall

Packet filter firewall is the common core component of the any network monitoring tool, which processes every packet header and passes those packets according to filter rules present in the access list [2]. It works in the network layer of the protocol stack and is the simplest type firewall compare to other types of firewall exist [3, 4]. In this type of firewall there is no consideration of applications, therefore focus is on the individual packets. Each incoming and outgoing packet contains the IP header, which has the information about source and destination address. IP packet internally contains the transport layer protocol information like source, destination port and protocol types. TCP and UDP are the transport layer protocols, in which TCP is connection oriented protocol and UDP is the connectionless protocol. Packet filter firewall is known as less secured one when compared with its counterparts because of its operation in low level devices. However the literature shows that even though such firewall is less secure, they are more efficient and easily accessible. Complexity of the packet filter firewall is easier to handle. Packet filtering technologies are the most widely used technologies due to the following reasons.

i) Packet filter technologies are fast compared to the other technologies.
ii) These are transparent to users and applications [5].
iii) Development cost is very less.

In this work, we focus on the packet filter firewall technologies. This is a five dimensional firewall system which considers the protocol, source address, source port, destination address and destination port [3, 4]. Implementation of the packet filter firewall can be done using different types of methodologies. The work in [8] describes the Trie-based development. Network policy is represented in the Trie data structure. Rules are represented as an ordered set of tuples maintaining precedence

 978-1-4244-8943-5/11/$26.00 ©2011 IEEE

Authorized licensed use limited to: Università di Bari Aldo Moro. Downloaded on October 28,2025 at 00:17:12 UTC from IEEE Xplore. Restrictions apply.

relationships among rules and ensuring policy integrity. It explains the construction of the Trie using sample rules. The advantage of using the Trie-based data structure is that it stores similar rules together, which allows the elimination of multiple rules as packet is processed. List based implementation is the simplest among the methods used to develop the packet filter firewalls [4]. For every incoming or outgoing packet firewall examines the rules in the access list in sequential order. Whenever the packet matches any rule then the corresponding accept/reject decision is taken. The list can be dynamically upgraded by promoting the most active rules on top of the list and thus the decision time of the incoming packet gets reduced. This work gives comparison results with the list-based packet filter implementation. But the disadvantage of this approach is that it is a static kind of packet filter firewall implementation. It doesn't consider the traffic characteristics. Another disadvantage using Trie-based approach is that if the access list contains the rules which are extremely different from one another then tree becomes larger and look up takes more traversals [8]. Therefore, Trie-based approach is good if the access list contains the rules which are interpreted and the access list is small.

Hashing can also be used in the packet filter firewall development [9]. The rules which are defined for the packet filtering contain tuples and for that packet classification algorithm called tuple space search has been introduced. In the real world, access lists typically use only a small number of distinct field lengths by mapping filters to tuples. Each tuple in a particular rule has a known set of bits in each field. By concatenating these bits in order hash key can be generated. This key can be used to map filters of that tuple in to hash table. They have used five-dimensional tuple space as we mentioned earlier. They have also given some concepts on tuple pruning algorithm which can be used to reduce the search time. This method is useful if the access list length is small and the rules are relative to each other. Otherwise, during the construction of hash table many collisions will occur by increasing the tuple search operation.

II. BASIC ISSUES AND APPROACHES

A. Binary Decision Diagrams

A Binary Decision Diagram (BDD) is a data structure that is used to represent a Boolean function [10]. Operations on BDDs are performed on the compressed representation of these functions. BDD is a rooted directed acyclic graph and consists of one or many decision nodes and two terminal nodes and edges which connect these nodes. A Solid edge is known to be high (1) and a dashed edge is known to be low (0). The following Fig. 1 shows the example BDD. The basic BDD can be classified into two types.

Ordered BDD (OBDD): if each variable of the BDD appears at most once in each complete path and if the variables appear in the same order in all other complete paths [11, 12].

Reduced OBDD (ROBDD): it is an ordered BDD without redundant nodes [11].

Out of these variants of BDDs ROBDD are most used types of BDDs. These ROBDDs can be obtained using CUDD package [13] by giving BDD as input. Example of such ROBDD obtained from Fig. 1 is shown in Fig 2. We can observe that number of nodes in the ROBDD is reduced by two as compared with original BDD. User defined variable ordering can also be used to construct the BDDs. TABLE I is the example of such variable ordering used in BDD construction.

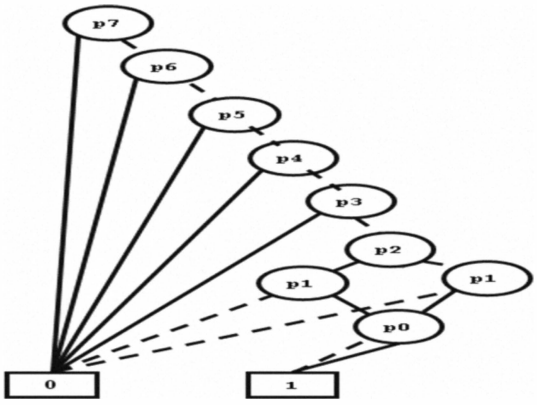

Figure 1: Example of a BDD

TABLE I: Boolean variables required for the representation of access list

Dimension Field	Boolean Variables	Total Number
Protocol type	p7, p6, ..., p1, p0	8
Source IP Address	sa31, sa30, ..., sa1, sa0	32
Destination IP Address	da31, da30, ..., da1, da0	32
Source Port	sp15, sp14, ..., sp1, sp0	16
Destination Port	dp15, dp14, ..., dp1, dp0	16
Total		**104**

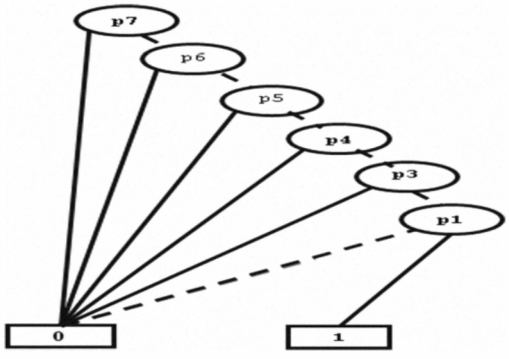

Figure 2: ROBDD representation after Fig. 1

Authorized licensed use limited to: Università di Bari Aldo Moro. Downloaded on October 28,2025 at 00:17:12 UTC from IEEE Xplore. Restrictions apply.

B. Use of BDD in Packet Filter Firewall

The basic theory of BDD-based packet filter firewall is given in the report [14]. Each packet contains header information like protocol, source address, source port, destination address and destination port. All these are represented using numbers and can be converted in to the equivalent binary format. Suppose, the protocol is TCP and its number is 6, which can be converted to binary as 00000110. To store these 8 bits 8 variables are needed and they can be named as p7, p6, p5, p4, p3, p2, p1, p0. Traversal on the BDD will be done from the top to bottom. For example, number 6, 128, 64 contains the following 8 bits.

p7	p6	p5	p4	p3	p2	p1	p0
0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0

First line in the above table contains bits for number 6. The above BDD (Fig. 2) represents both the numbers 6 and 3. Therefore, number 6 will traverse the BDD by comparing each bit from top to bottom and reaches leaf node 1 and which indicates the number 6 is accepted by the BDD. Bits present in the second line are for number 128. Now traversal starts from the top node p7 and corresponding bit in the table for number 128 is 1. At node p7, if the comparing bit 0 it traverses to the p6, but now the bit is 1, so that it reaches leaf node 0, indicating that 128 is rejected by the BDD. Number 128 also gets rejected by the BDD due to the bit mismatch at node p6. The table also mentions that the number variables are required to store the access list in the BDD format. For example purpose we give the sample access list in the following table.

TABLE II: Sample access list containing 5 rules

Rule#	Proto type	src_addr	src_mask	src_port	dest_addr	dest_mask	dest_port	Action
1	TCP	0.0.0.0	255.255.255.255	1023	146.141.0.0	0.0.255.255	25	Permit
2	IP	146.141.0.0	0.0.255.255	80	146.141.0.0	0.0.255.255	120	Permit
3	TCP	0.0.0.0	255.255.255.255	1023	146.141.0.0	0.0.255.255	80	Permit
4	IP	0.0.0.0	255.255.255.255	25	146.141.0.0	0.0.255.255	80	Permit
3	TCP	0.0.0.0	255.255.255.255	1023	146.141.0.0	0.0.255.255	1023	Permit

C. Advantages of BDD-based Approach

There are many advantages of BDDs in firewall technologies listed below.

i) *No redundant computation*: For deciding the action of the packet the look up algorithm checks from the root node to the leaf node without any repetition. Therefore we cannot find the redundant computation using BDD approach. But in case of other approaches there are many re-computations of the same variables.

ii) *Multi-Dimensional filtering*: BDD represents the multidimensional filtering with five dimensions like protocol, source address, source port, destination address and destination port.

iii) *Bit level information representation*: BDD holds the all dimension values in binary format.

III. DESIGN AND DEVELOPMENT DETAILS

In this section we are going to discuss about the implementation details and what are the modules implemented for the successful completion of the packet filter firewall project.

A. Development of a Module to Generate .blif File

In Table II a sample access list is shown. But in real life access list may contain hundreds or thousands of rules. We have implemented a module which automatically takes the access list as input and gives the corresponding .blif file as output. We can give this .blif file as input to the CUDD package [13] to obtain the optimized form of BDD as output files. The algorithm developed for the .blif file generation is as follows.

//Input: Rule List
//Output: blif file containing the binary form of the rule list.
While (Rule is exist in the rule list)
1. *take each rule from the rule list.*
2. *extract the protocol number, source address and port, and destination address and port.*
3. *convert each part in to its binary format in required number of bits*
4. *Store the each part of the converted binary form in to the blif file.*

B. Development of a Packet Filter

In this module packet filter firewall takes the *dot file* generated from CUDD [13] as the input file and decides the action of the incoming and outgoing packets.

***B.1 Algorithm*: BDD Lookup algorithm**

//Input : *CUDD order, dot file.*
//Output : *ACCEPT or REJECT the packet.*
Check the head of the BDD given by the CUDD
If (solid) then
    final_op = 1
else
    final_op = 0
lookup_ptr=first node of the BDD
while(lookup_ptr == 1 OR lookup_ptr == 0)
    if (header bits[lookup_ptr] == 1)
        lookup_ptr = high (lookup_ptr)
    else
        lookup_ptr = low (lookup_ptr)
    if (lookup_ptr == 1) then

Authorized licensed use limited to: Università di Bari Aldo Moro. Downloaded on October 28,2025 at 00:17:12 UTC from IEEE Xplore. Restrictions apply.

ACCEPT the packet

else

REJECT the packet

This algorithm searches the BDD generated by the CUDD according to the rule set for which the BDD is given. Thereafter it takes the each bit of the header and compares with the BDD. Once all the bits in the header are checked by traversing the BDD from root to leaf node the decision of packet's acceptability is considered. Normally the packet filter information is stored in a 2-dimensional vector, which contains the information all BDD nodes and their high and low edge values. TABLE III illustrates this. The word ***dotted*** indicates the output of the sub-graph below that node should be complimented. The word ***dashed*** means that the input to the corresponding node is 0. The lines which are not either ***dashed*** or ***dotted*** are ***solid*** lines, which denote the input for that node is 1.

TABLE III: Two-dimensional vector contains dot file information

row/ column	0	1	2	3	4	5
0	F0	->	d4	[style	=	solid]
1	d4	->	d3	height	1	
2	d4	->	af	[style	=	dashed]
3	d3	->	ad	height	2	
4	d3	->	d2	[style	=	dashed]
5	d2	->	d1	height	3	
6	d2	->	ad	[style	=	dashed]
7	d1	->	ad	height	4	
8	d1	->	d0	[style	=	dashed]
9	d0	->	ad	height	5	
10	d0	->	cf	[style	=	dashed]

Initially packet filter firewall takes the incoming packet from the network and the parses its header to get the header information. Each component is identified as *protocol type, source address, source port, destination address, destination port* and stores them in a bit array. Consider an incoming packet header as shown below.

<tcp><10.14.1.2><1023><146.141.0.0><25>

This packet header is divided and converted as below.

header_bits [104] = "000000110.....0010101"

As observed by the structure of BDD, for acceptance of a packet the firewall should check all variables present in the BDD. For rejecting a packet it can compare up to the node where the comparison reaches the leaf nodes. If the final result is ***1*** then ***accept*** the packet and if ***0*** then ***reject*** the packet.

C. List-based Packet Filter with Promotion Algorithm

List-based firewalls are widely used class of firewalls for large networks [4]. As we have explained in the above list-based firewalls are also multidimensional packet filter firewalls. The dimensions are same as protocol, source address, source port, destination address and destination port. Therefore, access list contains the different rules to be used to filter the packets. Each rule in the access list contains the action information about whether to accept or reject the packet which matches the corresponding rule. List-based algorithm has some major disadvantages. One of them is that rules are placed without any prior information like traffic characteristics. This may lead to the increase in decision making time when the packet comes to the firewall. For example, suppose given access list contains the 1000 rules. If 900th rule is the mostly occurring rule in accepting or rejecting a packet then this rule is hit by the maximum number of incoming packets. So, if this rule is placed at the beginning of the access list then significant gain can be obtained in terms of faster decision. Therefore, as in [4], we have incorporated the promotion algorithm in the list- based algorithm to obtain this gain. The algorithm C.1 for this is given below.

C.1 Algorithm for list-based packet filter with promotion

1. *Extract the incoming packet header information.*
2. *while access list is not empty final result = 1 do*
3. *if action = PERMIT then*
4. *ACCEPT the packet*
5. *rule hit count[rule no]++*
6. *else if action = DENY then*
7. *REJECT the packet*
8. *rule hit count[rule no]++*
9. *else*
10. *rule action is not defined*
11. *end if*
12. *end while*
13. *//for the default deny case*
14. *if access list is empty final result = 0 then*
15. *REJECT the packet*
16. *end if*
17. *//In the next iteration alter access list according to the rule hit count.*

Algorithm C.1 finds which rule is hit in the current iteration and it increments the number of counts of the corresponding rule. Likewise, we need to observe the large chunk of traffic data coming daily or weekly and find the correct order of the access list based on the promotion made on the rules. After all the packets are passed through the access list the algorithm sorts the list based on the descending order of the hit count of each rule. Thus list promotion helps the firewall to decide early on packet's accept/reject decision as these are mostly hit by the most active rules present on top of the new list.

IV. RESULTS AND ANALYSIS

We have carried out many experiments on the access lists generated by our own and also from the real world access list

Authorized licensed use limited to: Università di Bari Aldo Moro. Downloaded on October 28,2025 at 00:17:12 UTC from IEEE Xplore. Restrictions apply.

from our local computer centre. We have used Protocol-Source-Destination (PSD) order for the experiments and found significant outcome in the performance of packet filter firewall. Initially access list length is set to five to differentiate between a fixed order and the best variable order generated by CUDD. For example, TABLE IV and TABLE V illustrate two sets of access lists and the corresponding comparative result is shown in Fig. 3.

TABLE IV: Sample Access List containing 5 rules

# Rule	Proto type	src_addr	src_mask	src_port	dest_addr	dest_mask	dest_port	Action
1	TCP	0.0.0.0	255.255.255.255	1023	146.141.0.0	0.0.255.255	25	Permit
2	IP	146.141.0.0	0.0.255.255	80	146.141.0.0	0.0.255.255	120	Deny
3	TCP	0.0.0.0	255.255.255.255	1023	146.141.0.0	0.0.255.255	80	Permit
4	IP	0.0.0.0	255.255.255.255	25	146.141.0.0	0.0.255.255	80	Permit
5	TCP	0.0.0.0	255.255.255.255	1023	146.141.0.0	0.0.255.255	1023	Deny

TABLE V: No. of nodes varies using both user order and CUDD order

Access List No.	List length	No of nodes using user order	No of nodes using CUDD order
1	1	92	92
1	2	188	124
1	3	200	128
1	4	206	134

*Column B: fixed order and *Column C: with CUDD generated order

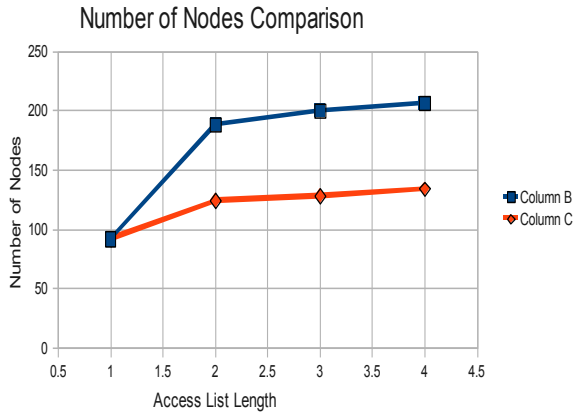

Figure 3: Number of nodes in BDD using with order and without variable ordering

From Fig. 3 it is realized that the number of nodes using the CUDD generated order is much lower than the user given fixed order. Therefore, storage needed for the corresponding packer filter firewall can be reduced significantly.

A. Results for Real World Access List

The above results shown are for the rules which are generated using the concepts of access list. For real world access list we have considered the access control list from our local computer centre, which has around 267 rules. We have taken initial 40 rules as first access list. We have used this first access list for the both the firewalls (BDD-based and list-based) as input. After that, in each stage, we have added 40 more rules to get another set of results. For each access list we have checked the number of comparisons required for the reject or acceptance of the given set of input packets, which are generated randomly. In BDD-based packet filter the number of comparisons is the number of nodes visited by the firewall to decide on the acceptance or rejection of the packets. For acceptance of the packet it will take more comparisons than the reject as it goes to the bottom level of BDD for confirmation. In list-based packet filter number of comparisons is the number of rules it has visited and in each rule how many packets it has checked. We have generated two different types of protocol; one is "most-accept packets" and another is "most-reject packets". In the case of "most-reject packets" protocol BDD-based packet filter will traverse lesser nodes in the BDD to decide the fate of the packet. But in the case of list-based packet filter the whole set of rules in the current access list will be traversed. Second type of input data we used is the "most-accept packets" protocol. In this case BDD-based packet filer the algorithm traverses the BDD from the top to bottom to decide the packet's acceptance. But in case of list-based approach the incoming packet may match with any rule in the access list. Hence, the total number of comparisons required will be depending on the position of the matching rule in the access list. Interestingly, in the second case also our design gives better result compared to list-based packet filter firewall.

TABLE VI shows the average number of comparisons taken by both the firewalls to decide it acceptance or rejection on the given one million random packets. In the case of the "most-reject packets" the performance of the BDD-based packet filter is higher as the number of comparisons is significantly less. For one million incoming packets with access list sets from 40 to 267 the average reduction in number of comparisons for BDD-based packet filter firewall is 75% when compared with list-based packet filter firewall and "most-reject packets" protocol is chosen. For "most-accept packets" protocol the average reduction is nearly 34%, but for 80-267 access list sets this reduction is even better, nearly 50%. Hence, for both the protocols the performance of BDD-based packet filter firewall is significantly better than the list-based (with rule promotion) packet filter firewall system. Fig. 4 and Fig. 5 illustrate these comparative results in graphical format.

V. CONCLUSION

Aim of this work is to represent the access list in the form of BDD and implementation of such BDD-based packet filter

Authorized licensed use limited to: Università di Bari Aldo Moro. Downloaded on October 28,2025 at 00:17:12 UTC from IEEE Xplore. Restrictions apply.

firewall. List-based packet filter was proven to be the best with rules promotion method, but considerably lacks its efficiency when compared to BDD-based approach. Our BDD-based design for packet filter firewall takes less space for storage and less look-up time for accept or reject the incoming packets.

TABLE VI: Comparative results between BDD-based and List-based (with promotion) approaches

Most-Reject Packets	**Number of comparisons with 1 million packets**		
Access List Length	**BDD based**	**LIST-based with promotion**	**% Reduction**
40	32054234	55668183	63.46
80	32026791	120134708	73.34
120	32537347	171710572	81.07
160	42643811	198408729	78.50
200	48068402	215368648	77.68
267	49690340	220103588	77.42
			Average Reduction = 75.24
Most-Accept Packets	**Number of comparisons with 1 million packets**		
Access List Length	**BDD based**	**LIST-based with promotion**	**% Reduction**
40	60386771	40360524	-49.61
80	58431012	68786826	15.05
120	57783157	100364081	42.47
160	58144886	129886903	55.23
200	55868916	164023455	65.93
267	55555756	208888173	73.40
			Average Reduction = 33.74
			Average Reduction (> 40) = 50.41

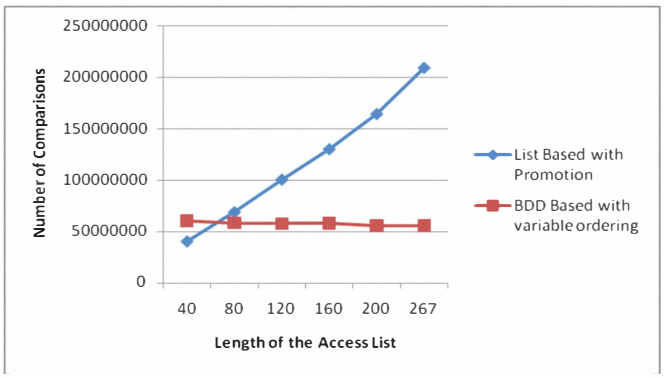

Figure 4: Graph comparison for "most accept" packets

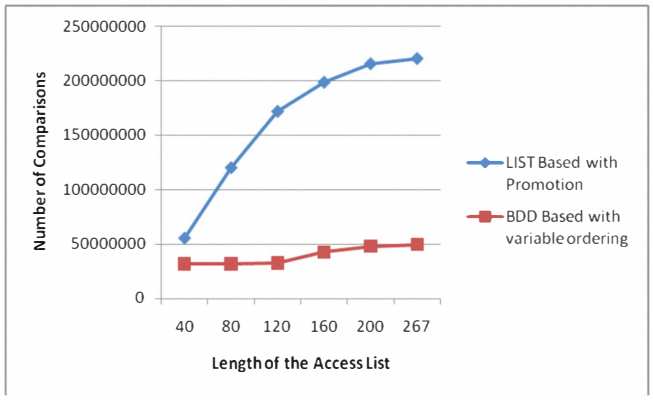

Figure 5: Graph comparison for "most reject" packets

REFERENCES

[1] K. Ingham and S. Forrest, "A History and Survey of Network Firewalls", *Technical Report 2002-37*, University of New Mexico Computer Science Department, 2002.

[2] C. Shen, T. Chung, Y. Chang and Y. Chen, "PFC: A New High Performance Packet Filter Architecture", *Journal of Internet Technology*, Vol.8, No.1, Page (s): 67-74, 2007.

[3] S. Acharya, J. Wang, Z. Ge, T. Znati and A. Greenberg, "Simulation study of Firewalls to Aid Improved Performance", *Proceedings of ANSS*, Page (s): 18-26, 2006.

[4] S. Acharya, J. Wang, Z. Ge, T. Znati and A. Greenberg, "Traffic-Aware Firewall Optimization Strategies", *Proceedings of ICC*, Page (s): 2225-30, 2006.

[5] H. Julkunen and C. Chow, "Enhance Network Security with Dynamic Packet Filter", *Proceedings of ICCN*, Page(s): 268-275, 1998.

[6] W. R. Cheswick and S. M. Bellovin, "Firewalls and Internet Security: Repelling the Hacker", *Addison-Wesley*, Reading, MA, 1994.

[7] M. Christiansen and E. Fleury, "An MTIDD Based Firewall", *Telecommunication Systems 27:2-4*, Page(s): 297-319, 2004.

[8] E. W. Fulp and S. J. Tarsa, "Trie-Based Policy Representations for Network Firewalls", *Proceedings of ISCC*, Page(s): 434-441, 2005.

[9] V. Srinivasan, S. Suri and G. Verghese, "Packet Classification using Tuple Space Search", *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM, Page(s) 135-146, 1999.

[10] S. B. Akers, "Binary decision diagrams", *Transaction on Computers*, Vol. C-27(6), Page(s): 509–516, 1978.

[11] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation", *Transaction on Computers*, Vol. C-35(8), Page(s): 677-691, 1986.

[12] R. E. Bryant, "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams", *ACM Computing Surveys*, Vol. 24, No.3, Page(s): 293-318, 1992.

[13] F. Somenzi, "CUDD: CU Decision Diagram Package"; *http://bessie.colorado.edu/~fabio/CUDD*.

[14] S. Hazelhurst, A. Fatti and A. Henwood, "Binary Decision Diagram Representations of Firewall and Router Access Lists", *ftp://ftp.cs.wits.ac.za/pub/research/reports/TR-Wi*, 1998.

Authorized licensed use limited to: Università di Bari Aldo Moro. Downloaded on October 28,2025 at 00:17:12 UTC from IEEE Xplore. Restrictions apply.