Investigation and Analysis of Efficient Firewall Packet Filtering and Matching Algorithms

Dr. Prof.P.K.Deshmukh pkdeshmukh9@gmail.com
Dr. Prof. A.B.Bagwan, Aliakbar.bagwan@gmail.com
Ms.P.Kinage pragatikinage@yahoo.com
Ms. S.A.Jadhav Jadhav.shriys36@gmial.com

I. Abstract

For the network devices like firewall or IPSec, packet filtering is plays very critical role in high speed networks. Thus it is important that firewall policies should be optimized in order to provide the efficient security for high speed networks. There are many techniques presented by researchers for exploiting the characteristics of the filtering policies, however they do not consider the traffic behavior in optimizing their search data structures. In this paper we are discussing the recent new optimized packet filter and packet matching techniques for both stateless and statfull firewall. Algorithm first is presented with an objective of reduction in packet matching cost in all cases where as algorithm second is presented with an objective of less cost and less packet matching time.

We are discussing the algorithm first which is basically presented in order to produce efficient performance in terms of lower cost of packet matching of the firewall. The performance of the algorithm is related to complexity of the firewall rule set and is compared to an alternative algorithm demonstrating that the algorithm here has improved the packet matching cost in all cases. Thus in short we present an algorithm which orders the rules in a firewall rule set to best suit the trends in the network traffic (as given by a recent network trace file) and therefore reduce the potential number of packet-rule matches. Whereas in second investigated algorithm we consider a classical algorithm that we adapted to the firewall domain. We call the resulting algorithm "Geometric Efficient Matching" (GEM). The GEM algorithm enjoys a logarithmic matching time performance.

Index Terms—GEM, network security, firewall, firewall queries, firewall testing, firewall correctness.

II. Introduction

Firewalls and IPSec gateways have become major components in the current high speed Internet infrastructure to filter out undesired traffic and protect the integrity and confidentiality of critical traffic. In these devises, the filtering decision is based on a *security policy* defined according to predefined high level security requirements and is composed of a set of ordered filtering rules against which the net work traffic is sequentially matched in order to determine the appropriate filtering action. Therefore, packet filtering is a critical component that determines the performance of these networking devices. With the dramatic advances in the current network speeds, firewall packet filtering must be constantly optimized to cope with the network traffic demands and attacks. This requires reducing the packet matching time needed to \allow" or \deny" packets in order to minimize the end-to-end delay. This problem is even more critical for application-level filtering, where a wide variety of packet filtering fields is used. Thus, efficient yet easy to implement filtering policy optimization is highly crucial to enable high speed packet filtering for effective deployment of traffic filtering technologies in the Internet.

In this paper we have to present the solution to this well studied problem of firewall packet matching. Apart from this, there are some other issues which make such problem bit complex. 1. Unlike firewalls, routers use "longest prefix match" semantics. 2. Firewall matching problem is 4D or 5D, whereas router matching is usually 1D or 2D: A router typically matches only on IP addresses, and does not look deeper, into the TCP or UDP packet headers. 3. Major firewall vendors support rules that utilize IP address ranges, in addition to subnets or CIDR blocks: this is the case for Check Point and Juniper—the main exception is Cisco that only supports individual IP addresses or subnets. Thus in these cases it becomes crucial that firewall needs their own efficient packet filtering and matching algorithm with objective of cost and time effective processing.

General efficient packet matching techniques exist; a review of some is presented by Gupta and McKeown [1]. However, these techniques do not make use of information about the relative frequency of the packets that are to be matched. Hence it will just increase the overall cost of matching and computation time.

In this paper, we are investigating the two new methods for packet matching which are efficient as compared to existing algorithms. Below Section III is presenting the literature overview over firewall types and rules optimization techniques. Section IV discussing both algorithms investigated, and section V discuss the work done over this algorithms.

III. Literature Review

A. *Firewall Types*

This research is focusing on the processing mode type of firewalls as they relate to Network Layer packet filtering. We however give a summary of firewall classifications here following.

1. *Packet Filter Firewall*: These are based on first generation firewall technology. They analyze network traffic at the transport layer. They examine each IP network packet to see if it matches one of the rules defined for allowing or denying data flows.

The decision is based on the information they get from the packet's transport layer headers and the direction the packet is going into. They are therefore configured to check:

Transport layer type (TCP, ICMP, UDP)
Source port
Destination IP address
Source IP address
Network interface the packet arrives on
Destination port

Packet filters do the above by applying a rule set residing in the TCP/IP kernel that defines what action goes with which rule. They come under three sub categories: 1.1 Static Filtering, 1.2 Dynamic Filtering, and 1.3 Stateful Filtering.

2. Circuit Level Firewalls: These are based on second generation firewall technology. They work based on the fact that a packet is either a data packet or a connection request belonging to a connection or circuit between two peer transport layers. These firewalls work by:
Checking that each connection setup follows a handshake system for the transport layer protocol being used.
Storing a session identifier for the connection
Connection state: handshake, established, or closing
Only forwarding packets after the handshake is complete
Maintaining a table of valid connections and removing it once the connection is terminated
Closing the virtual circuit after transmission

3. Application Layer Firewall: Also called third generation firewall. These firewalls evaluate packets for valid data at the application layer before allowing a connection.
Examines data in network packets at the application layer
Maintains connection state and sequencing information
Can validate passwords and service requests
Most of them include proxy services for specific services such as HTTP or FTP which provide more checks and generate audit records about the traffic they transfer.

4. Dynamic Firewalls [Stateful Firewall]: A fourth generation firewall type allowing modification of the rule base. A virtual connection is established and the packet is allowed to travel the firewall server. These provide support for UDP packets by associating them with a virtual connection. The connection information is kept for a short period and the connection is terminated if no response packet is received within that short time. They are good for not allowing unwanted UDP packets into a network because the response packet must contain a destination address that matches the original source address.

5. Hybrid Firewall: Because of the need to do more than packet inspection, firewalls are being implemented as hybrid systems. These are mostly implemented by adding packet filtering to an application gateway. Cisco PIX firewalls are an example of such hybrid firewalls.

B. Firewall Rule Set and Optimzation Techniques

A rule set is a group of rules programmed to allow or deny packets. The decision to allow or deny is based on the values contained in the packet. The firewall rule set processes both the packets arriving from the public Internet, as well as the packets originating from the internal network. Every service based on TC/IPA i.e.: telnet, www, mail, etc. is predefined by its protocol i.e. SSH, HTTP, SMTP etc. and privileged (listening) port. Packets destined for a specific service, originate from the source address using an unprivileged (high order) port and target the specific service port on the destination address. All the above parameters (i.e. ports and addresses) are used as traditional selection criteria to create rules which will pass or block services.
Optimization Techniques:
1. Rule set clean up: This approach analyzes the rule set to identify inconsistencies and redundancy in the rule set and removes them. Rules that make no unique contribution to the firewall behavior are removed. These are either Redundant or Shadowed rules. Redundant rules never match packets because there are more preceding rules matched first. Unused rules that have the log option but have no logs showing that they have matched packets. They are therefore candidates for removal when optimizing though their removal may affect firewall behavior later. Remove rules for unused Network groups E.g. if the organization does not have a mail server, SMTP rules can be taken out of the configuration.
2. Rule set re-ordering: This technique relies on the network statistics logged on the firewall or written to file in a network database. It lists most used rules in decreasing order of usage by hit count and percentage hit count. These rules can then be moved towards the beginning of the rule set to improve performance. Optimize the rule set by ordering rules based on the rule usage data and rule order dependencies that does not alter the firewall behavior. This moves the most used rules toward the beginning of the rule set until they are very close to the source of an order dependency. The complexity with this method is the dependencies problem.
3. Rule Grouping: It is evident that a major part of the network traffic matches a small subset of the firewall rules. This therefore calls for selecting these rules and calling them by groups depending on their usage. This scheme divides the filtering policy into two layers of rules, (a) most active rules-those performing the most packet matching and (b) inactive rules-perform much less matching. Rules are checked and if two or more rules are found to have the same matching action, they are merged. This reduces the rule set size and consequently the search time for the filtering

algorithm because fewer rules are inspected when deny or allow decision is to be made by the firewall.

4. Rule Frequency re-ordering: The number of times a rule is triggered is recorded and used to determine matching patterns and arrangement of the whole rule set. Ehab Al-Shaer et.al. propose an adaptive way of dynamically optimizing firewall rule sets using actively calculated statistics.

5. Rule Editing: Firewalls have thousands of rules and hundreds of IP addresses to take care of. The typical approach is to scan through all these rules in a linear method until a match is found for the packet being inspected.

6. Go To Function: Modern Firewalls come with a feature allowing skipping from one rule or rule set to another rule in a rule set. The go to function is used to switch the search and match flow from the default one (next rule in the list) to the one specified in the go to command.

IV. INVESTIGATED ALGORITHMS

1. Stateless Firewall Optimization Algorithm

As in the problem statement, the optimization problem of firewall is nothing but the placing rules in way that most commonly used rules are behind the top of the rule set and hence it results into the less searching time. Thus to provide the optimization to this problem, generally the rules those are associated with weight is equals to total number of maches of this particular rules in traffic flow. A naive approach would be to rank order the rules according to the weightings; but this would not take into account rule dependencies and would probably change the security policy. This is prompt for new more efficient and complex approach for rule optimization.

Thus the new algorithm is presented in [1], which is recent research in this area for stateless firewall. Here below algorithm 1 is added from [1] as investigation algorithm for this research paper.

Algorithm 1 OptimiseAllRules(start_rules)

```
1: $H \leftarrow$ BuildHeap($H$, start_rules)
2: rule_list $\leftarrow$ CreateList(rule_list, nil)
3: **while** $H$ _= ∅ **do**
4:   $Rb \leftarrow$ HeapGet($H$)
5:   $S \leftarrow \{$all rules preceding $Rb\}$
6:   **for all** $Rc \in \{S$ then $Rb\}$ **do**
7:     **if** $Rc$ /∈ rule_list **then**
8:       current $\leftarrow$ ListTail(rule_list)
9:       best_cost $\leftarrow$ Cost(rule_list)
10:      best_pos $\leftarrow$ current
11:      ListRemove(rule_list, $Rc$)
12:      **while** current _= ListHead(rule_list) **do**
13:        current $\leftarrow$ ListPrevious(current)
14:        $Rt \leftarrow$ ListGet(current)
15:        **if** $Rt$ is not preceding $Rc$ **then**
16:          ListInsertAfter(current, $Rc$)
17:          **if** Cost(rule_list) < best_cost **then**
18:            best_cost $\leftarrow$ Cost(rule_list)
19:            best_pos $\leftarrow$ current
20:          **end if**
21:          ListRemove(rule_list, $Rc$)
22:        **else**
23:          **break**
24:        **end if**
25:      **end while**
26:      ListInsertAfter(best_pos, $Rc$)
27:      HeapRemove($H$, $Rc$)
28:    **end if**
29:  **end for**
30: **end while**
31: **return** rule_list
```

This algorithm is showing the more improved results as compared to previous algorithm for all the performance metrics [1]. This algorithm is evaluated using the SCADIA Dataset which is collected from various intonations internet flows.

2. Stateful Firewall Optimization Algorithm

Recently many firewalls are generally statefull, which having the two advatanges such as 1. administrator does not need to write explicit rules for returntraffic—and such return-traffic rules are inherently insecure since they rely on source-port filtering. And hence such firewalls are more secure as compared to simple stateless packet filters. 2. One more advantage is that this kind of firewalls are more faster and simpler than the rule match algorithms. Hence the statefulness is provided great perforamnce advantages [2].

There are basically two different search mechanisms from which the firewall statfulness is implemented like the first one is the *slow algorithm* which implements the “first match” semantics and compares a packet to all the rules and the second method is the *fast state lookup* mechanism which checking that whether this particular packet is belongs to an existing open flow [2].

As per stated in investigated paper [2], for the TCP connections, stateful firewalls delivering the more performance in which the fast lookup method is handling many of packets. But for the connectionless ICMP or UDP, only the “slow” algorithm is active and hence this makes real bottleneck. Hence as per stated in [2], “slow” algorithm doesn’t need to be slow, and its showing that the GEM algorithm has a matching speed that is comparable to that of the state lookups: In isolation, the algorithm required under 1 usec per packet, and our Linux GEMiptables implementation sustained a matching rate of over 30,000 packets-per-second (pps), with 10,000 rules, without losing packets, on a standard PC workstation [2].

Thus according to the stated problem, it becomes essential to present the complex algorithm for stateful firewalls which is most efficient and practical algorithm for firewall packets. In this paper we have to investigate the efficient packet matching algorithm for firewall networks that adopted in [2]. They consider a classical algorithm that we adapted to the firewall domain. We call the resulting algorithm "Geometric Efficient Matching" (GEM) [2]. The GEM algorithm enjoys a logarithmic matching time performance. However, the algorithm's theoretical worst-case space complexity is O(n4) for a rule-base with n rules. Because of this perceived high space complexity, GEM-like algorithms were rejected as impractical by earlier works. Contrary to this conclusion, this paper shows that GEM is actually an excellent choice. Based on statistics from real firewall rule-bases, we created a Perimeter rules model that generates random, but no uniform, rule-bases. We evaluated GEM via extensive simulation using the Perimeter rules model.

Proposed GEM:

The firewall packet matching problem finds the first rule that matches a given packet on one or more fields from its header. Every rule consists of set of ranges [li; ri] for i = 1; : : : ; d, where each range corresponds to the i-th field in a packet header. The field values are in 0 · li; ri · Ui, where Ui = 232 ¡ 1 for IP addresses, Ui = 65535 for port numbers, and Ui = 255 for ICMP message type or code. Table 1 lists the header fields we use (the port fields can double as the message type and code for ICMP packets) [3].

The Data Structure:

The GEM search data structure consists of two parts. The first part is an array of pointers, one for each protocol number, along with a cell for the '¤'—all protocols and small header for every cell in array. Header contains information about the order of data structure levels and pointer to the first level and the number of simple ranges in that level. The second part represents the levels of data structure. Every level is a set of nodes, where each node is an array. Each array cell specifies a simple range, and contains a pointer to the next level node. In the last level the simple range information contains the number of the winner rule instead of the pointer to the next level.

The Search Algorithm:

The packet header contains the protocol number, source and destination address and port numbers fields. First, we check the protocol field and go to the protocol array of the search data structure, to select the corresponding protocol database header. From this point, we apply a binary search with the corresponding field value on every level, in order to find the matching simple range and continue to the next level. The last level will supply us with the desired result—the matching rule number [2].

Search time:

In each level we execute a binary search on an array of at most 2n entries, where n is the maximal number of active rules. We process two searches: one with the packet's protocol and one in the '¤' data structure. Thus, for d levels, the search time is O(d log n). For a constant d = 4, we get an O(log n) search time. Note that the '¤'search data structure only has 2 levels (for IP addresses), thus the search time is dominated by the time to search the 4 levels of the TCP search data structure [3].

The Build Algorithm:

The build algorithm is executed once for each protocol. The input to the build algorithm consists of the rule-base, plus the field order to use. The order dictates the contents of each data structure level, and also, the order in which the header fields will be tested by the search algorithm [2]. Thus using this build algorithm we evaluated the performance metrics such as build time and space complexity.

V. EVALUTION

Both the above algorithms are evaluated using the different online datasets in order to shows their effectiveness. Their particular results are presented in our investigation papers. In this our investigation work, prepared the following architectural overview of this two approaches one for stateless firewall filters and other for stateful firewall filters and hence both are efficient in their own cases still to the date [Figure 1 and Figure 2].

However, in the same papers we found that this approaches still having some disadvantages which we can put for further research optimization work for both stateful and stateless firewalls.

In case of algorithm 1 for stateless firewall, some studies show that this algorithm is about 11 times slower than the Hamed and Al-Shaer algorithm. With the optimization of a static rule set this performance disadvantage is of little consequence as the optimization of a typical rule set using Algorithm 1 is performed in the order of 10 seconds and this task would only need to be performed infrequently when new rules are inserted or when a significant change in traffic performance is noted [1].

Same in case of proposed new GEM algorithm, when we are trying to apply it on the higher levels of our data structure, it resulted into that this greatly increases the preprocessing time, and only gives minor improvements to the space complexity. It was basically proposed to overcome this problem but still its showing the litter bad performances in such cases.

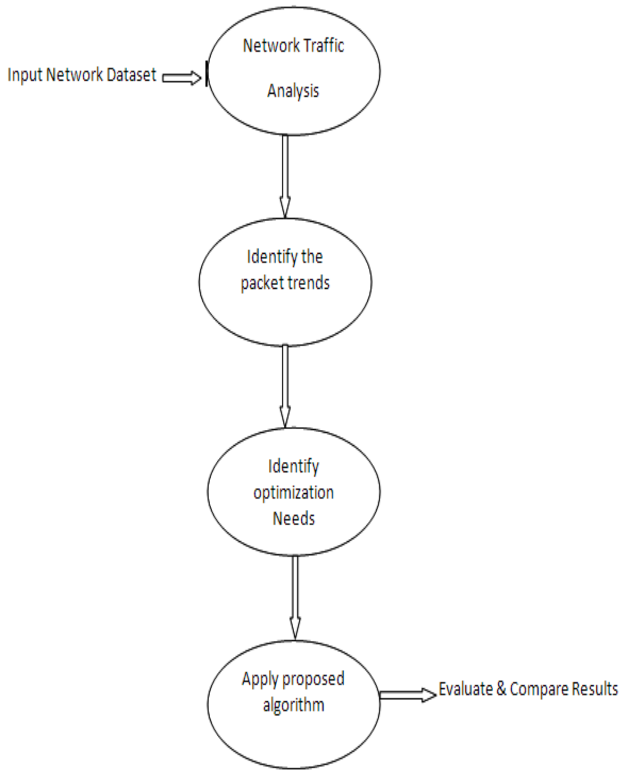
Input Network Dataset
Network Traffic Analysis
Identify the packet trends
Identify optimization Needs
Apply proposed algorithm
Evaluate & Compare Results

Figure 1: Stateless Algorithm 1 Implementation Design

VI. Conclusion and Future Work

Thus in this research paper we investigated the two packet matching and filtering methods for stateless and statful firewalls. Here we discussed those approaches along with their evaluation results. In case of stateless approach, one is the motivation to optimize the rule order of firewalls such that the performance of the firewall is improved by reducing the potential number of packet-rule matches. Hence we investigate this new algorithm 1 here. The algorithm presented here is shown to have improved performance compared to an earlier reported algorithm in all cases, at the cost of higher runtime complexity. The increased runtime complexity is unlikely to be significant for the offline optimization of a firewall, which is the main target of this algorithm.

In the other case means stateful firewall optimization GEM is presented and investigated. We have seen that the GEM algorithm is an efficient and practical algorithm for firewall packet matching. We implemented it successfully in the Linux kernel, and tested its packet-matching speeds on live traffic with realistic large rule-bases. GEM's matching speed is far better than the naive linear search, and it is able to increase the throughput of iptables by an order of magnitude. On rule-bases generated according to realistic statistics, GEM's space complexity is well within the capabilities of modern hardware. Thus, we believe that GEM may be a good candidate for use in firewall matching engines [2].

Apart from this investigation studies, still there are downfalls for this algorithms which we need to present in further work and proposed new approaches to overcome this drawbacks.

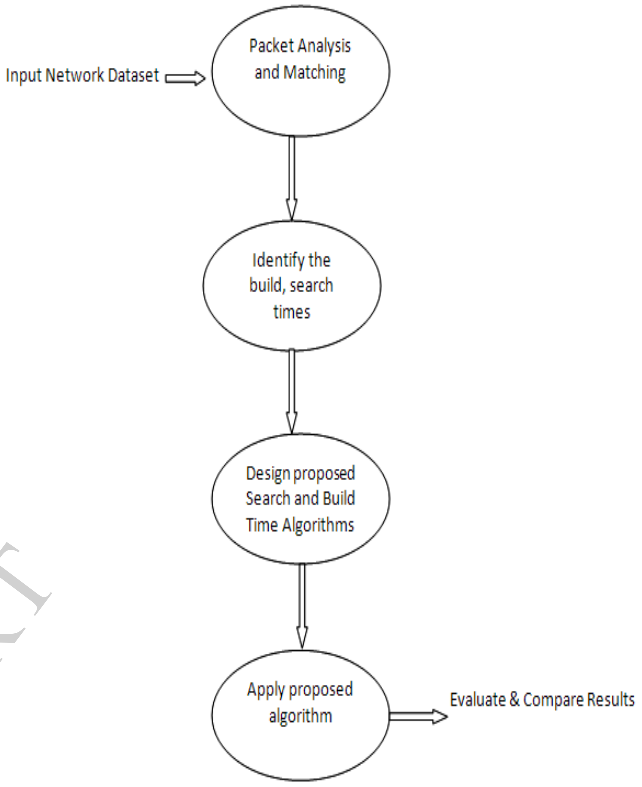
Input Network Dataset
Packet Analysis and Matching
Identify the build, search times
Design proposed Search and Build Time Algorithms
Apply proposed algorithm
Evaluate & Compare Results

Figure 2: Stateful Algorithm Design

VII. Reference

[1] "Optimising Rule Order for a Packet Filtering Firewall", Ian Mothersole and Martin J. Reed, IEEE, 2011.

[2] "The Geometric Efficient Matching Algorithm for Firewalls", Dmitry Rovniagin and Avishai Wool, Senior Member, IEEE, 2011.
[3] "Study of GEM", http://dc310.4shared.com/doc/fbIZsCX8/preview.html

[4] M.M. Buddhikot, S. Suri, and M. Waldvogel, "Space Decomposition Techniques for Fast Layer-4 Switching," Proc. Conf. Protocols for High Speed Networks IV, pp. 25-41, Aug. 1999.

[5] W.R. Cheswick, S.M. Bellovin, and A. Rubin, Firewalls and Internet Security: Repelling the Wily Hacker, second ed. Addison-Wesley, 2003.

[6] M. Christiansen and E. Fleury, Using Interval Decision Diagrams for Packet Filtering, http://www.cs.auc.dk/fleury/publications. html, 2002.

[7] E. Cohen and C. Lund, “Packet Classification in Large ISPs: Design and Evaluation of Decision Tree Classifiers,” Proc. ACM SIGMETRICS, pp. 73-84, 2005.

[8] S. Crosby and D. Wallach, “Denial of Service via Algorithmic Complexity Attacks,” Proc. 12th USENIX Security Symp., pp. 29-44, Aug. 2003

[9] C. Shannon, E. Aben, k. claffy, and D. Andersen, “The CAIDA anonymized 2008 internet traces.” [Online]. Available: http://www.caida.org/data/passive/passive 2008 dataset.xml

[10] E. Cohen and C. Lund, “Packet classification in large ISPs: Design and evaluation of decision tree classifiers,” in *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS ’05. New York, NY, USA: ACM, June 2005, pp. 73–84.

[11] E. L. Lawler, “Sequencing jobs to minimize total weighted completion time subject to precedence constraints,” in *Algorithmic Aspects of Combinatorics*, ser. Annals of Discrete Mathematics, B. Alspach, P. Hell, and D. J. Miller, Eds. Elsevier, 1978, vol. 2, pp. 75–90.

[12] E. W. Fulp, “Optimization of network firewall policies using directed acyclic graphs,” in *Proceedings of the IEEE Internet Management Conference*, 2005.