A Data Driven FireWall for Faster Packet Filtering

Mohammad M. Masud
m.masud@uaeu.ac.ae

Umniya Mustafa
Umniya.M@uaeu.ac.ae

Zouheir Trabelsi
Trabelsi@uaeu.ac.ae

College of Information Technology, United Arab Emirates University
PO Box 15551, Al Ain, United Arab Emirates

***Abstract*—Packet filtering performance of basic firewalls largely affects the throughput of a network protected by the firewall. The packet filtering firewalls filter packets based on a set of filtering rules. The traditional approach for packet filtering works by checking a packet against the filtering rules by scanning from the first rule in the set and continuing to scan rules until a match is found. If no match is found, then a default rule is applied. This approach is inefficient if the number of rules is too large and majority of the packets match with rules located towards the end of the rule set. In this paper, we propose a data mining based technique for packet filtering. We consider each rule in the rule set a class. A classifier is first trained with labeled training data. Each such labeled data point contains a packet header info and the corresponding class label (i.e., rule number with which the packet matches). Then the classifier is used to classify new incoming packets. The predicted class (i.e., rule number) is checked against the packet to see if this packet really matches the predicted rule. If yes, the corresponding action (i.e., accept or deny) of the rule is taken. Otherwise (if prediction of the classifier is wrong), we go back to the traditional way of matching rules. The advantage of this data mining firewall is that it offers a much faster rule matching. We have proven both analytically and empirically that even with millions of real network traffic packets and hundreds of rules, the classifier can achieve very high accuracy, thereby making firewall six times or more faster in making filtering decision.**

I. Introduction

A firewall acts as a network's "first line of defense" or security guard at the entrance of a local network. It is mainly used by an organization to protect itself from unauthorized access from and to the Internet while allowing legitimate communication to pass through.In other words, it handles incoming and outgoing network traffic by allowing traffic to either enter the network or block it from going through. This protects the network from potential threats or unwanted activities that might be caused by both the insiders who intent to communicate the Internet and outsiders who intent to communicate the local network. The permission of allowing or denying is based on the configuration of a set of rules called "filtering rules". These filtering rules are configured to permit or deny traffic according to the packet's characteristics, including but not limited to the source and destination IP addresses, port numbers, services, and protocol types. It uses one or more set of rules to inspect the packets as they come in or go out from the network. These rules inspect one or more characteristics of the packet. Whenever the firewall triggers a rule that matches a packet, based on the configuration of the firewall, it either discards or permits it from accessing into the network. This technique is called 'Packet Filtering'. In the firewall, the number of filtering rules can be very large.. When a packet is to be filtered by a firewall, this single packet is compared with all the filtering rules one by one in order until a match is found. This leads to longer processing time for the firewall because it processes each packet and decides whether to discard or allow it. The situation even becomes worse if a large amount of traffic is sent at the same time (e.g. in case of a denial of service attack). In this case, the firewall becomes overloaded and might either drop some of the packets when they should not be dropped, might allow some packets when they should not be allowed, or the server might crash. These three cases have been identified as very critical problems in networking security, and they have been vital research topics in the last few decades. This problem addresses the need for an effective way to optimize the firewall processing time. In this paper, we propose a solution using Data Mining Classification techniques to improve the firewall processing time and thus, improve its performance. This Section is followed by a review of the previous works that are related to this topic. Section III then describes our implementation in details. Then in Section IV, we evaluate our approach with several experiments. Finally, Section V concludes the paper with directions to future works.

II. Related Work

Most of the early research works on firewall performance focus on the improvement of packet searching times using various mechanisms including hardware-based solutions [1], specialized data structures [2], [3], and heuristics [4]. Research works in [5]–[8] focus on the statistical filtering schemes to improve the average packet processing time. The structure of searching by taking into account the packet flow dynamics is introduced in [8], [9]. The optimization of firewall filtering policies utilizing the characteristics of packet flow over Internet is presented in [10]. Segments-Based Tree Search (STS) scheme [7] uses bounded depth Huffman trees to enhance the searching based on the statistics collected from segments. However, this scheme may need large overheads for maintaining the tree periodically. To reduce the overheads, Segments-based List Search (SLS) [7] was proposed using (MRU) order instead of using trees.

The idea of firewall optimization through early packet rejection was introduced in [6], [11]–[13]. In [11], early packet

978-1-4799-3764-6/14/$31.00 © 2014 IEEE

rejection is done through rule-fields ordering. In [12], early packet rejection is done through multilevel filtering process including field and intersection filtering modules. In [6], an approach named FVSC is proposed to optimize the rejection path. This technique uses set cover approximation algorithm to construct early rejection rules from the original security policy common field values. PBER technique in [13] is considered as a generalization of FVSC [6] in the sense that FVSC [6] focuses only on rejection path while PBER [13] finds short cuts for both accepted and rejected packets.

The works relevant to this paper are the ones dealing with rules reordering. They fall into two categories: 1) Rules reordering including dependency as in [6], [10] and 2) Rules reordering for disjoint rules as in [14], [15]. To the best of our knowledge, all research work done in the field of firewall optimization through rule reordering emphasize on the importance of rule-fields reordering in early packet rejection. [16] applies decision tree classifier for packet classification. However, our approach addresses the problem differently, where each class label is a rule, rather than accept or deny.

III. Data Driven Firewall

The proposed data mining based firewall will be referred to as "data driven firewall" (DDF). The high level overview of DDF is illustrated with figure 1

There are two main components of the firewall, namely, the offline and the online component. The offline component collects network packets and use them to train a classifier (see Section III-A). The online component does the actual packet filtering online (see Section III-B). Here the classifier trained with the offline component is used to classify each incoming packet. The predicted class corresponds to rule, which belongs to the set of filtering rules. If the classifier predicts the correct rule, i.e., the predicted rule *matches* the packet, then the corresponding rule action is taken. Otherwise, the prediction is wrong, and in that case the traditional firewall is used to find the matching rule for the packet and the corresponding rule action is taken. By "rule action" we mean the action (i.e., accept or deny) corresponding to the rule.

Before going into the details, we will introduce several terms that will be frequently used in this paper.

Definition 1 (Packet, E_j): A network *packet* E_j is a data structure consisting of several fields, such as Protocol, Source_IP, Destination_IP, Source_Port, Destination_Port, and so on. In this paper, we consider only the packet header information and ignore the payload content.

Definition 2 (Rule, R_i): A rule R_i consists of two parts, namely, a condition and an action, as follows R_i: $C_i \Rightarrow A_i$, where C_i is the condition and A_i is the action. The condition of a rule consists of a number of field tests (e.g. Source_IP = 10.*.*.* AND Protocol = TCP AND Destination_Port = 80 AND ...), and the action is either *accept* or *deny*.

Definition 3 (Rule Match, $\mathcal{M}(E_j, R_i)$): A packet E_j matches a Rule R_i if the field values of E_j satisfy all the field tests in the condition C_i. We will denote this case (i.e., E_j matches with R_i) with the symbol $\mathcal{M}(E_j, R_i)$. Likewise,

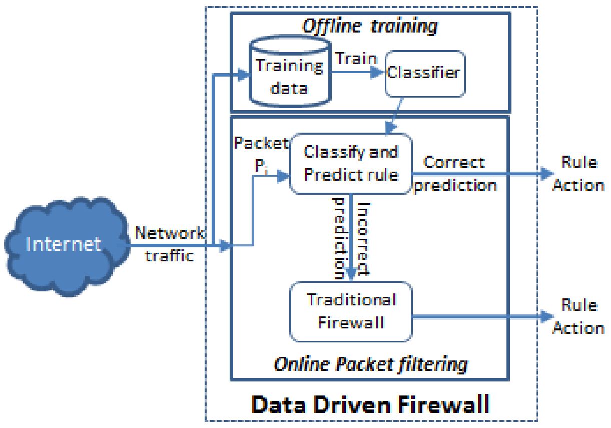

Fig. 1. Overview of the data driven firewall

$\neg\mathcal{M}(E_j, R_i)$ will be used to denote the case where E_j doesn't match R_i.

Definition 4 (Rule Set, $\mathcal{R}$): The rule set $\mathcal{R} = \{R_1, R_2, ..., R_N\}$ is the set of N filtering rules in the firewall, where each R_i is a rule in the set. Usually, the last rule R_N is the default rule, having empty condition (i.e., matches with any packet).

Definition 5 (Standard firewall (STF)): The standard firewall is a firewall that for each incoming packet E_j, sequentially searches through the rule set $\mathcal{R}$, starting from the first rule R_1. If E_j doesn't match R_i (i.e., $\neg\mathcal{M}(E_j, R_i)$, then the search continues with the next rule R_{i+1}. Otherwise, if $\mathcal{M}(E_j, R_i)$, then the action A_i is taken.

A. Training the DDF

The training data $\mathcal{D} = \{d_1, ..., d_M\}$ consists of a set of M training instances d_k, $k \in \{1, ..., M\}$, where d_k is the tuple: (E_{j_k}, i), such that $\mathcal{M}(E_{j_k}, R_i)$. In other words, each training instance consists of a packet E_{j_k} and the *class label* of the packet, i. The class label, i, of a packet is the index of the first rule (R_i) in the rule set such that matches the packet. For example, let the packet E_j matches the 10-th rule (i.e., R_{10}) in the rule set. Then the class label of E_j is 10. The class label of a packet can be found by running the STF for the packet.

Once we have a training data, we can use them to train a classifier L of our choice. We have tried with many different classifiers, but the best ones in terms of classification time and accuracy of prediction were "decision tree" and "ripper" classifiers (Section IV). Note that the whole process of collecting the training data and training the classifier is done offline.

B. Filtering with DDF

A packet E_j is provided to the classifier L, as input. Note that we only provide the packet, not the class label. Recall that the class label of a packet is the index of the rule with which the packet matches. The task of the classifier is the predict the class label of E_j. Let $L(E_j)$ be the prediction (output or predicted class label) of the classifier for the packet E_j. Let $L(E_j) = i$, i.e., the classifier predicts i as the class label. A classifier can have wrong prediction, and therefore, we must check the validity of this prediction. So, now we test if $\mathcal{M}(E_j, R_i)$. If yes, then the prediction $L(E_j)$ is correct, and the corresponding action A_i is taken. Otherwise, (i.e., if $\neg\mathcal{M}(E_j, R_i)$) the prediction $L(E_j)$ is wrong, and the packet must go through the STF to fetch the matching rule and accordingly, the corresponding action.

C. Performance improvement

Let $T_S(E_j)$ be the time needed to filter the packet E_j with the STF, and $T_D(E_j)$ be the time needed to filter E_j with the DDF. Also, let $I_L(E_j)$ be the indicator function such that

$$I_L(E_j) = \begin{cases} 0, & \text{if prediction } L(E_j) \text{ is correct} \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

Let $T_L(E_j)$ be the time needed by the classifier to predict the class label of E_j (i.e., the classification time). Therefore, we can write:

$$T_D(E_j) = T_L(E_j) + I_L(E_j).T_D(E_j) \tag{2}$$

In other words, equation 2 says that the time to filter a packet by the DDF is equal to the classification time if the prediction is correct, and the classification time plus STF filtering time if the prediction is wrong. Therefore, the time needed to filter a batch of B packets $\mathcal{E} = \{E_1, ..., E_B\}$ is given by:

$$T_S(\mathcal{E}) = \sum_{j=1}^{B} T_S(E_j) \tag{3}$$

$$T_D(\mathcal{E}) = \sum_{j=1}^{B} T_L(E_j) + (1-P)\sum_{j=1}^{B} T_S(E_j) \tag{4}$$

where P is the percentage of packets correctly classified (i.e., predicted correct) by the classifier. In other words, P would be the accuracy of prediction for the set of packets $\mathcal{E}$. Using equations 3 and 4 we can infer that there would be a gain in filtering time if:

$$\begin{aligned} & T_D(\mathcal{E}) < T_S(\mathcal{E}) \\ \Rightarrow & \sum_{j=1}^{B} T_L(E_j) + (1-P)\sum_{j=1}^{B} T_S(E_j) < \sum_{j=1}^{B} T_S(E_j) \qquad (5) \\ & \Rightarrow \sum_{j=1}^{B} T_L(E_j) < P\sum_{j=1}^{B} T_S(E_j) \\ & \Rightarrow T_L(\mathcal{E}) + < PT_S(\mathcal{E}) \\ & P > \frac{T_L(\mathcal{E})}{T_S(\mathcal{E})} \qquad (6) \end{aligned}$$

Therefore, there would be a gain in running time if the prediction accuracy of the classifier is greater than the ratio of total classification time (T_L)to the total STF filtering time. For many classification techniques, T_L would be much less than T_S. For example, for the decision tree classifier, the classification time is less than half the filtering time of STF (which is empirically justified by our experiments). So, in this case, there would be a gain even if the prediction accuracy is 50% (i.e., half of the packets are incorrectly classified). In real world scenarios, the accuracy of a classifier is much higher than 50% provided that it is trained with enough training data. We also derive an interesting relationship between the classification accuracy and running time of DDF from equation 5. That is, the filtering time of DDF decreases with increasing accuracy of the classifier. This has been confirmed with the empirical evaluations with the DDF (Section IV).

IV. Experiments

In this section we describe the datasets, experimental environment, and discuss and analyze the results.

A. Data sets and experimental setup

We have used real network traffic from the CAIDA anonymized internet traces 2012 dataset (http://www.caida.org/data/passive/passive_2012_dataset.xml). We extracted only the packet headers and used the following fields: Protocol, Packt_len, IP_src, IP_dst, PORT_src, PORT_dst, TCP_flag0, ..., TCP_flag7, ICMP_type, ICMP_code.

Competing approaches:

DDF: The data driven firewall that we propose in this paper. Here the classifier is decision tree.

STF: The standard firewall, which is used as the baseline.

Parameter settings: We filter exactly the same set of packets for both the DDF and STF, which consists of 10 million packets. For training the classifier, we use 10,000 packets. But these training packets are not used in test set (i.e., for filtering). Also, exactly the same set of firewall rules are used to test the both firewalls, which consists of 500 rules. These rules have been generated by hand, which follows from standard security policies observed in our institution.

Hardware and software: The experiments were done on a standalone workstation having Intel Core i5 2.4GHz processor with 8GB RAM and 750GB Hard Drive. The OS was Windows 7. All programs (STF and DDF) has been developed with Java in NetBeans environment and Weka machine learning API (http://www.cs.waikato.ac.nz/ml/weka/) have been used for the classifier.

B. Evaluation

We evaluate the system based on two criteria: i) the total processing time, and ii) the total number of *hits*. A comparison of a field value of a packet with the corresponding field test of a rule is counted as a hit.

Figure 2(a) shows the processing time comparison between STF and DDF. The X-axis of this graph corresponds to number of packets processed (in millions) and the Y-axis corresponds the total processing time in milliseconds. For example, at X=10 Million, the Y values of STF and SSF are 821098 and 123181, respectively, meaning, STF takes 821098 millioseconds to filter 10 Million packets, whereas DDF takes only 123181 milliseconds, meaning, DDF takes

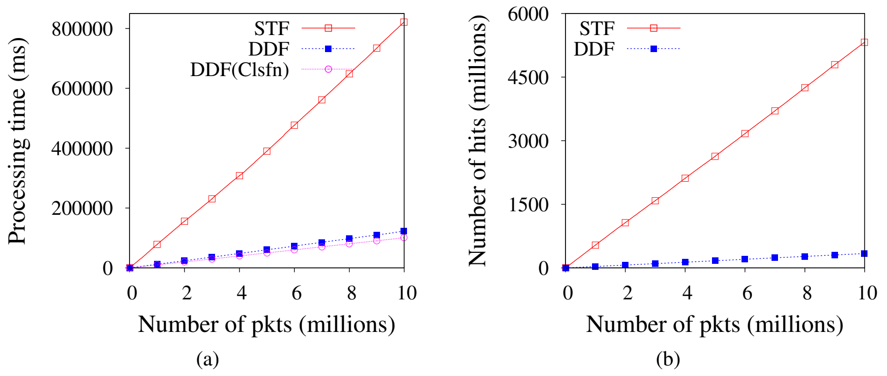

Fig. 2. Number of packets vs (a) cumulative processing time and (b) number of hits

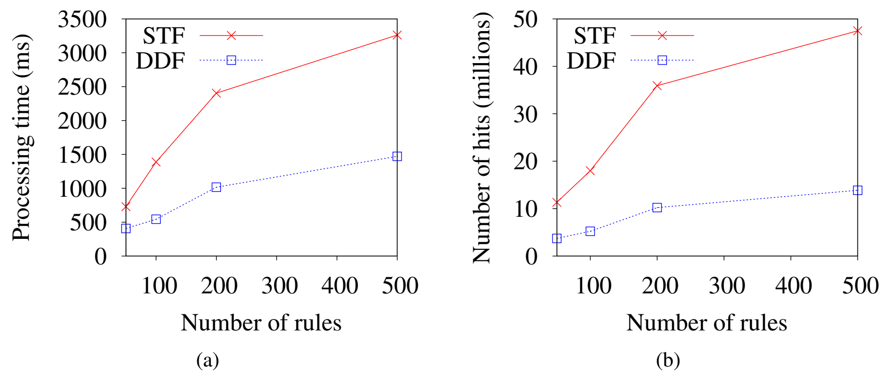

Fig. 3. Number of rules vs (a) cumulative processing time and (b) number of hits

about one seventh of the time taken by STF. In other words, the throughput of DDF is more than six times as that of STF. Figure 2(a) also shows the classification time (cumulative) taken by the classifier of DDF, which is shown with the *DDF(Clsfn)* curve. It is evident from the chart the the total filtering time required by DDF is just a little more than the classification time, which means most of the class predictions have been correct. In fact, the class prediction accuracy in this case was 98% or more. However, using equation 6, Section III-C, we can infer that even if the prediction accuracy were as low as 13% ($>$ 1/8), the total running time of DDF would have been less than that of STF, because here, the classification time of DDF is clearly one eighth of the filtering time of STF (821 seconds and 101 seconds, for STF and classification by DDF, respectively).

Figure 2(b) plots the number of hits (in millions) against the number of packets processed. We see also see a similar trend where number of hits increase linearly. However, for DDF, the number of hits is less than one seventh of that of STF. This improvement occurs because the STF searches for rules sequentially through the list of rules incurring large number of hits as well as running time. On the contrary, DDF finds the appropriate rule with the help of the classifier without browsing through the list, and therefore, incurs much less hits and running time.

Figure 3(a) shows how the processing time varies with the number of rules in the firewall. We have run both the firewalls with 50, 100, 200, and 500 rules. As expected, with the increasing number of filtering rules, processing times also increases. But the rate of this increment is higher for STF than DDF. This is because with the increase in the number of rules, the time to find a matching rule also increases in STF as it has to browse through a longer list. However, for DDF, the searching time does not increase as much because of the (mostly) correct predictions made by the classifier. The slight increase that DDF observes is due to the extra time needed to classify an instance (since now the tree is more complex) and extra time needed when the classifier makes wrong prediction. Figure 3(b) reports how the number of hits varies with the number of rules in both firewalls, which also follow similar trend as of the processing time.

Effect of accuracy and number of training data on processing time of DDF: The size of training data has a direct impact on the accuracy of the classifier. Generally, the prediction accuracy of a good classifier should increase with increasing size of training data. This is observed in our experiments, as shown in Figure 4(a). Here the X-axis represents the number of training data (in hundreds) and the Y-axis represents the prediction accuracy of the classifier on the entire test data. There are two curves, one for 100 rules and the other for 500 rules. The curve for 100 rules shows the accuracy of the classifier when we have 100 rules in the firewall, and

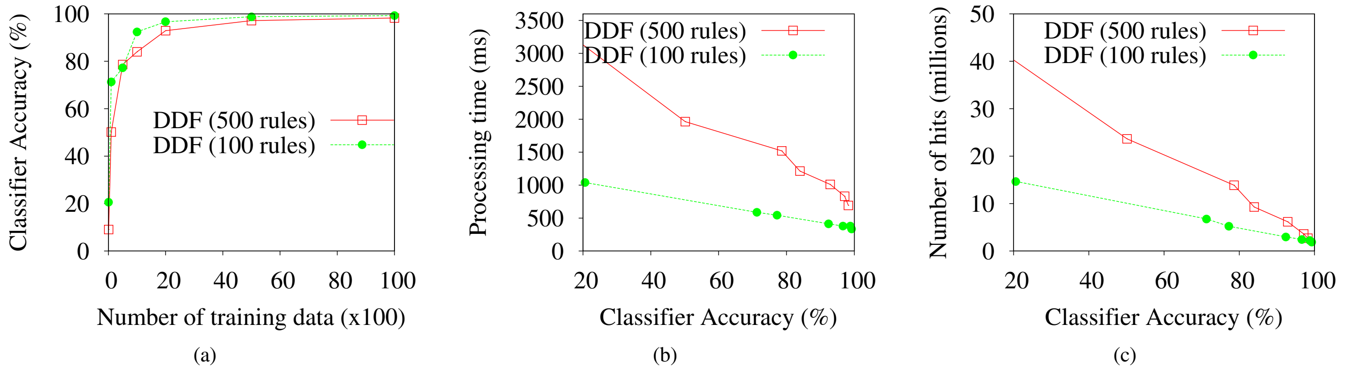

Fig. 4. (a) Size of training data vs classifier accuracy, (b) accuracy vs processing time and (c) number of hits for DDF

the classifier is trained and tested accordingly. The same description goes for the 500 curve. Both of these curves observe the expected behavior, i.e., increase of prediction accuracy with increasing training data.

Recall that from our theoretical analysis on the impact of classifier accuracy on filtering time (Section III-C), we concluded that filtering time of DDF decreases with increasing accuracy. The empirical observations confirm this theory, as reported in Figures 4(b),(c). The figure 4(b) shows how the processing time varies with the prediction accuracy of the classifier and figure 4(b) shows the same for number of hits. In both these figures we report the performance of DDF for 100 rules and 500 rules. In general, the processing time and number of hits reduce as the prediction accuracy increases. This is because with higher prediction accuracy, the correct rule is predicted more often, requiring less browsing through the rule list to find the matching rule. We get another interesting observation from these figures. We observe that for larger number of rules, the processing time, as well as the number of hits of DDF is higher. For example, when accuracy is 92%, the processing times of DDF with 100 rules and 500 rules are 414 ms and 1011 ms, respectively. For the same accuracy, the number of hits are 2.9 millions and 6.2 millions, respectively. Therefore, the larger the number of rules, the higher the number of hits (and running time) even for the same accuracy. This happens because for example when the accuracy is 92%, only 8% packets are incorrectly classified, meaning, the DDF has to browse through the list of rules to find a match for these 8% packets. However, this browsing takes longer time and incurs more hits when the list is larger (e.g. 100 vs 500 rules).

V. Conclusion

We have introduced a novel data driven approach for faster packet filtering. In this approach a classifier is first trained to predict the matching rule for any given packet and employed in the real network. This results in at least six times more speed in packet filtering compared to a standard firewall that browses through the set of rules to find the matching rule for a packet. We have proven the effectiveness of our approach both theoretically and empirically with real network traffic. We have also analyzed different parameters of the system, such as number of rules and the size of training data.

In the future, we would like to enhance the data driven firewall by experimenting with different classifiers, larger real data, consider different attack scenarios to evaluate the robustness of our approach.

References

[1] F. Baboescu and G. Varghese, "Scalable packet classification." in *ACM SIGCOMM'01*, 2001.

[2] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings." *Interconnects*, vol. 8, 1999.

[3] T. Y. C. Woo., "A modular approach to packet classification: Algorithms and results." in *IEEE INFOCOM'00.*, March 2000, pp. 1213–1222.

[4] P. Gupta and N. McKeown, "Algorithms for packet classification." *IEEE Network*, vol. 15, no. 2, pp. 24–32, 2001.

[5] H. Hamed, A. El-Atawy, and E. Al-Shaer., "On dynamic optimization of packet matching in high-speed firewalls." *IEEE Journal on Selected Areas In Communications*, vol. 24, no. 10, Oct 2006.

[6] I. Mothersole and M. Reed., "Optimizing rule order for a packet filtering firewall." in *SAR-SSI*, 2011.

[7] A. El-Atawy, T. Samak, E. Al-Shaer, and H.Li., "Using online traffic statistical matching for optimizing packet filtering performance." in *IEEE INFOCOM'07*, 2007, pp. 866–874.

[8] L. Kencl and C. Schwarzer., "Traffic-adaptive packet filtering of denial of service attacks." in *WOWMOM'06:: The 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks.*, Washington, DC, USA 2006, pp. 485–489.

[9] S. Acharya, M. Abliz, B. Mills, and T. Znati., "Optwall: a hierarchical traffic-aware firewall." in *Proc. of 14th Annual Network and Distributed System Security Symposium (NDSS).*, San Diego, US, Feb 2007.

[10] H. Hamed and E. Al-shear., "Dynamic rule-ordering optimization for high-speed firewall filtering." in *ASIACCS' 06.*, Taipei, Taiwan, March 2006.

[11] Z. Trabelsi, L. Zhang, and S. Zeidan., "Packet flow histograms to improve firewall efficiency." in *ICICS'11*, Dec 2011.

[12] Z. Trabelsi and S. Zeidan., "Multilevel early packet filtering technique based on traffic statistics and splay trees for firewall performance improvement." in *ICC'12*, June 2012.

[13] E. Al-Shaer, A. El-Atawy, and T. Tran., "Adaptive early packet filtering for defending firewalls against dos attack." in *IEEE INFOCOM'09.*, 2009.

[14] W. Wang, H. Chen, J. Chen, and B. Liu., "Firewall rule ordering based on statistical model." in *International Conference on Computer Enginnering and Technology*, 2009.

[15] W. Wang, R. Ji, W. Chen, B. Chen, and Z. Li., "Firewall rules sorting baseb on markov model." in *International Symposium on Data Privacy and E-Comerce.*, 2007.

[16] E. Cohen and C. Lund., "Packet classification in large isps: design and evaluation of decision tree classifiers." in *SIGMETRICS '05, Proceedings of the 2005 ACM SIGMETRIC international conferenceon Measurement and modeling of computer systems*. ACM Press, New York, NY, USA. March 2005, pp. 73–84.