Traffic-Aware Firewall Optimization Strategies

Subrata Acharya[†], Jia Wang[§], Zihui Ge[§], Taieb F. Znati[†,‖] and Albert Greenberg[§]
[†]Department of Computer Science,
[‖]Telecommunications Program
University of Pittsburgh, Pittsburgh, PA 15260
[§]AT&T Labs Research, Florham Park, NJ 07932
(sacharya, znati)@cs.pitt.edu
(jiawang, gezihui, albert)@research.att.com

***Abstract*— The overall performance of a firewall is crucial in enforcing and administrating security, especially when the network is under attack. The continuous growth of the Internet, coupled with the increasing sophistication of the attacks, is placing stringent demands on firewall performance. In this paper, we describe a traffic-aware optimization framework to improve the operational cost of firewalls. Based on this framework, we design a set of tools that inspect and analyze both multi-dimensional firewall rules and traffic logs and construct the optimal equivalent firewall rules based on the observed traffic characteristics. To the best of our knowledge, this work is the first to use traffic characteristics in firewall optimization. Furthermore, we develop a novel adaptation mechanism that dynamically detects anomalous traffic behavior and adaptively alters the firewall rules to avoid serious performance degradation due to the traffic anomaly. To evaluate the performance of our approaches, we collected a large set of firewall rules and traffic logs at tens of enterprise networks managed by a Tier-1 service provider. Our evaluation results find these approaches very effective. In particular, we achieve more than 10 fold performance improvement by using the proposed traffic-aware firewall optimization.**

I. Introduction

A firewall is a combination of hardware and software used to implement a security policy governing the flow of network traffic between two or more networks. In its simplest form, a firewall acts as a security barrier to control traffic and manage connections between internal and external network hosts. The actual means by which this is accomplished varies widely, and ranges from packet filtering and proxy service to stateful inspection methods. A more sophisticated firewall may hide the topology of the network it is employed to protect, as well as other information, including names and addresses of hosts within the network. The ability of a firewall to centrally administer network security can also be extended to log incoming and outgoing traffic to allow accountability of user actions and to trigger alerts when unauthorized activities occur.

Firewalls have proven to be useful in dealing with a large number of threats that originate from outside a network. They are becoming ubiquitous and indispensable to the operation of the network. The continuous growth of the Internet, coupled with the increasing sophistication of attacks, however, is placing further demands and complexity on firewalls design and management.

Increased firewall complexity, undoubtedly, brings with it increased vulnerability and reduced availability of individual network services and applications. Analysis of real configuration data has shown that corporate firewalls are often enforcing rule sets that violate established security guidelines. Furthermore, the need to deal with large set of diverse security policies and rules imposes additional burden on firewalls, thereby rendering the performance of the firewall highly critical to enforcing the network security policy. In this context, the protection that a firewall provides becomes as good as, not only the policies it is configured to implement, but equally importantly the speed at which it enforces these policies. Under attack or heavy load, firewalls can easily become a bottleneck. As the network size, bandwidth, and processing power of networked hosts continue to increase, there is a high demand for "optimizing" firewall operations for improved performance.

Multi-dimensional firewall optimization is proven to be NP-hard [1], [2]. This has led the research community to focus on developing various "optimization" heuristics to make firewalls more efficient and dependable. Despite significant progress in the design of firewalls, the techniques for firewall optimization remain static, and as such, fail to adapt to the continuously varying dynamics of the network. This is mostly due to their inability to take into account the traffic characteristics logged by the firewall, such as source and destination, service requests and the resulting action taken by the firewall in response to these requests. Moreover, current firewall designs do not support adaptive anomaly detection and countermeasure mechanisms to deal with short and long term attacks. Consequently, they run the risk to become unstable under attack.

The objective of this paper is to address the above shortcomings and develop a sound and effective "toolset" to accelerate firewall operations and "adapt" its performance to the dynamically changing network traffic characteristics. Achieving this goal, however is challenging, as the number of policies and security rules a firewall has to enforce for enterprise networks is large. In addition, there is a need for maintaining high policy integration. This is further compounded by the limited resources of firewalls relative to the increased ability of the network to process and forward traffic at extremely high speed.

In this paper, we focus on *list-based* firewalls, as they are the most widely used firewalls. The major contribution of this paper is the design of a Traffic-aware Firewall Optimizer to study, analyze, and optimize multi-dimensional list-based firewalls. To the best of our knowledge, this is the first work which considers traffic characteristics in optimizing firewall performance. A unique feature of this toolset is its adaptive anomaly detection and countermeasure mechanism, used to dynamically alter the firewall rule set to improve performance. The experimental study shows that the firewall optimizer is successful in dynamically exploiting the traffic characteristics to significantly enhance the performance of the firewall. Results show that operational cost of firewalls is reduced to less

than 7% of the initial unoptimized rule set.

The rest of the paper is organized as follows: In Section II, we briefly review related work. In Section III, we describe a framework for firewall optimization and argue about the importance of using traffic characteristics in optimizing firewall operations. In Section IV, we discuss the various optimization strategies. We present a discussion of the experimental results in Section V. Finally, we provide the conclusion in Section VI.

II. Related Work

Due to the enormous impact of firewalls on network security, there has been a significant amount of research work on how to optimize firewalls. Much of this work, however, has been in the area of firewall rule and policy modeling and optimization. Few attempts have been made to achieve multi-dimensional firewall optimization. In [3], a tool to model firewall policies and detect conflicts is described. In this work, the authors focus mainly on single prefix rules. Similarly, in [4] a constraint logic programming (CLP) framework to analyze rule sets is discussed. These research work offer a good insight in how to model and analyze rule sets. Neither of these works, however, consider optimizing a multi-dimensional rule set.

The approach proposed in [5] optimizes the firewall rule set using Directed Acyclic Graphs (DAGs) to describe rule dependencies. However, it does not provide a methodology to build the DAG. Furthermore, for complex graphs this scheme is ineffective. The approach proposed in this paper removes all the dependencies and hence it becomes possible to achieve optimum rule ordering. In [6], a framework to analyze and optimize rule sets is described. However, the authors do not provide specific details on how optimization can be achieved within the proposed framework. Furthermore, this work does not consider the traffic characteristics in its optimization approach. The proposed accelerating firewall toolset differs from the above approaches, in its unique approach to consider firewall traffic characteristics in optimizing firewall rule sets.

III. List-Based Firewalls

This study focuses on list-based firewalls, a widely used class of firewalls for large networks. In this section, we first discuss briefly the basic structure of a list-based firewall. We then define rule redundancy and dependency relationships in a list-based firewall.

A. List-based firewalls

A firewall security is typically defined by a set of rules. A rule is a multi-dimensional structure, where each dimension is either a set of network fields or an action field. A network field can be *source address*, a *destination address*, a *service type*, a *protocol number*, and a *port number*. An action field can be either *accept* or *deny*, or *others* (e.g. redirect to a server that perform further processes etc). Formally, a rule R can be represented as: $R = [\Phi^1, \Phi^2, \cdots, \Phi^k; \Sigma]$, where Φ^j, represents network fields and Σ is an action field. In an Internet environment, a typical rule can be represented as follows:

$$< src = \{s_1, s_2, \cdots, s_n\};\ dst = \{d_1, d_2, \cdots, d_m\};$$
$$srv = \{\sigma_1, \sigma_2, \cdots, \sigma_l\};\ action = \{drop\} >,$$

where s_i represents a source IP address, d_i a destination IP address, and σ_i a service type.

In list-based firewalls, rules describing the network security policies from a "priority" list. The priority of a rule, also referred to as its *rank*, is based on its position within the list. Earlier occurring rules have higher rank than later ones.

List-based firewalls work by logically examining the rules in sequential order. For each packet, the first matching rule determines the action taken by the firewall. This is referred to as the *first hit principle*[1].

B. Rule redundancy and dependency

Rule redundancy in list-based firewalls can be of two types, namely internal or external. For a given rule, internal redundancy occurs when at least one of its fields contains duplicate entries. Internal redundancy can also occur if there are suboptimal representations of entries within a field. For example, if one of the fields of the rule represents a network address, the appearance of the address values *192.168.1.0/24* and *192.168.0.0/24* within this same field constitutes an internal redundancy. This apparent redundancy can be removed by replacing the above address values with *192.168.0.0/23*.

In list-based firewalls, external redundancy between two rules occurs when one of the rules is a superset of the other one and appears earlier in the firewall rule set. This makes the second rule redundant with respect to the first one, as all traffic for which the second rule applies is filtered by the first rule. Formally, rule R_2 is said to be externally redundant with respect to rule R_1, if and only if: (i) R_1 is a superset of R_2, and R_2's rank $> R_1$'s rank. Externally redundant rules can be removed without violating the semantic integrity of the security policy.

Two rules are dependent if they mutually exhibit a precedence relationship. Formally, rules R_1 and R_2 are dependent if the following conditions are satisfied: (i) R_1 and R_2 are not disjoint, (ii) R_2's rank $> R_1$'s rank, and (iii) R_1's action field $!= R_2$'s action filed. As a consequence, if two rules R_1 and R_2 are dependent, then R_2 cannot be moved before R_1, without violating the semantic integrity of the rule set.

Two rules are said to be disjoint if they differ at least in one of their fields. Formally, rule $R_1 = [\Phi_1^1, \Phi_1^2, \cdots, \Phi_1^k; \Sigma_1]$ and $R_2 = [\Phi_2^1, \Phi_2^2, \cdots, \Phi_2^k; \Sigma_2]$ are disjoint if and only if there exists at least one i such as $\Phi_1^i \bigcap \Phi_2^i = \emptyset$.

IV. Firewall Optimization Framework

As stated above, firewall policies of an actively managed enterprise network may often change in response to new services, new threats or underlying network changes. The intrinsic complexity of the firewall policies makes it difficult to track down these changes. As a consequence, inefficiency, such as redundancies between rules and suboptimal representations of rule sets and fields within a rule, arise. The Traffic-aware Firewall Optimizer (TFO) takes a novel approach and uses traffic characteristics to address these problems. The overall architecture of the TFO is depicted in Figure 1. The figure also illustrates the optimization process used to optimize a raw initial firewall rule set.

The process starts with the *Pre-Optimization* phase. The main objective of this phase is to remove all redundancies in the rule set. At the end of this phase, all internal and external redundancies in the rule set are removed. Unless there

[1]It is to be noted that not all firewalls work with the first hit principle, also there are list-based firewalls which do not use first hit principle.

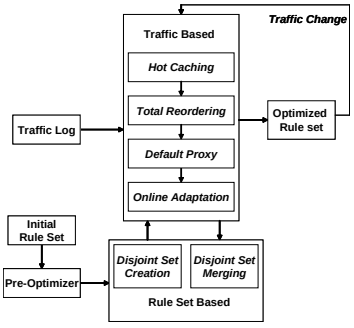

Fig. 1: Traffic-Aware Firewall Optimizer

Rule	Src	Dst	Srv	Action
R_1	s_1, s_2, s_3	d_1, d_2, d_3	σ_1	drop
R_2	s_2, s_3, s_4	d_2, d_3, d_4	σ_1	accept
R_3	s_5	d_4	σ_1	accept

TABLE I: Pre-optimized rule set: $\mathcal{S}_\mathcal{I}$

Rule	Src	Dst	Srv	Action
R_1	s_1, s_2, s_3	d_1, d_2, d_3	σ_1	drop
R_2^1	s_4	d_2, d_3, d_4	σ_1	accept
R_2^2	s_2, s_3	d_4	σ_1	accept
R_3	s_5	d_4	σ_1	accept

TABLE II: Disjoint rule set: $\mathcal{S}_\mathcal{D}$

is a change in the current firewall policy, the pre-optimization phase is performed only once.

The core component of the optimization process uses a *rule set based* optimizer and a *traffic based* optimizer. Both optimizers cooperate to adaptively optimize the rule set in response to dynamically changing traffic characteristics. This cooperation is achieved through a dynamic feedback mechanism. The rule set based optimizer takes as input the pre-optimized rule set and produces a rule set based optimized set of rules. This set is then fed to the traffic based optimizer. Using the current traffic log, the traffic based optimizer produces an optimum rule set which reflects the current characteristics of the traffic without violating the semantic integrity of the initial rule set. The traffic-aware optimized rule set is used by the firewall to enforce the security policy. This continues until changes in the traffic characteristics take place. In response to these changes, the adaptive optimization process is re-invoked using the current rule set and a new traffic-aware optimized rule set is produced. This process continues iteratively, until the enterprise network security administrator changes the rule set. When this occurs, the new rule set is pre-optimized before the rule-based and traffic-based optimizers are invoked. In rest of the section we discuss the basic mechanisms used by each of these optimizers and their collaborative interactions.

A. Rule Set Based Optimization

The rule set based optimizer operates exclusively on the rule set, with no additional consideration of other factors impacting network or traffic behavior. The optimizer continuously seeks to create new definitions in order to make rules in the current rule set disjoint. This, in turn, provides the traffic based optimizer with full flexibility to re-order rules based on traffic characteristics.

The rule based optimizer is composed of two basic components, namely the *Disjoint Set Creator (DSC)* and the *Disjoint Set Merger (DSM)*. These two components are typically executed sequentially. Initially DSC detects and removes dependencies from the current rule set. Then it creates new rule definitions in order to make the entire rule set disjoint. It is to be noted that this phase may lead to an increase in the rule set size. This is due to the fact that more rules may be needed to define each set of dependent rules. It is typical that there is only a small portion of rules that dependent on other rules[2].

The main task of DSM is to merge the rules of the disjoint rule set produced by DSC in order to optimize the rule set representation. The merging process iteratively selects one rule and tries to merge it with other rules. Merging occurs between rules with same action field, to preserve semantic integrity. Merging between two rules occurs when at most one field has different values in the rules. Upon completion of this optimization step, the rule set size is reduced to its most concise representation.

Notice that it is possible to reduce the rule set based optimization strategy to rule merging only, without the creation of disjoint rules. Such an approach still results in improved rule set representation, while minimizing the processing overhead. Combining disjoint set creation and merging, however, enables the optimizer to effectively capture the dynamics of the traffic characteristics, thereby resulting in an optimized rule set representation.

To illustrate the process of creation of disjoint rules out of an initial pre-optimized set of rules, and merge the resulting disjoint rules into a concise rule set representation, consider the example of a pre-optimized rule set, $\mathcal{S}_\mathcal{I}$, as shown in Table I.

Notice that R_2 is dependent on R_1, since the source and destination fields of R_2 intersect with the corresponding fields of R_1, while the action fields of the two rules are different. These rules can be made disjoint, without violating semantic integrity. This is achieved by keeping R_1 unchanged and forking R_2 into two new rules, R_2^1 and R_2^2, resulting in the disjoint rule set, $\mathcal{S}_\mathcal{D}$, as shown in Table II.

As observed from the above example, creating a new disjoint rule set increases the size of the original rule set. The new set size can be further optimized by merging rule R_2^2 and R_3 into R_4, to produce the final rule set, $\mathcal{S}_\mathcal{F}$, as shown in Table III.

Rule	Src	Dst	Srv	Action
R_1	s_1, s_2, s_3	d_1, d_2, d_3	σ_1	drop
R_2^1	s_4	d_2, d_3, d_4	σ_1	accept
R_4	s_2, s_3, s_5	d_4	σ_1	accept

TABLE III: Final rule set: $\mathcal{S}_\mathcal{F}$

Our DSC scheme looks similar to [7]'s decorrelation algorithm, but it uses a multi-dimensional tree instead of single dimensional tree to improve the runtime. Furthermore, we gain much better optimization using DSM, that the above work does not consider.

B. Traffic Based Optimization

The traffic based optimizer operates on the rule set produced by the rule set based optimizer. The optimizer uses current

[2]In the analyzed firewall dataset, this ratio is around $(1/15)^{th}$ of the total number of rules.

traffic characteristics to determine the order in which rules in the rule set are to be invoked to optimize the operational cost of the firewall. To achieve this goal we use four schemes, namely *hot caching*, *total re-ordering*, *default proxy*, and *online adaptation*.

The *hot caching* revolves around the concept of a *hot rule set*. A rule is said to be *hot* if it experiences a large number of traffic hits. The basic idea of this approach is to identify a small set of hot rules, relative to the original rule set, and cache these rules at the top of the rule set. Such a strategy results in dealing with a large amount of traffic hits, very early in the inspection process, thereby reducing the overall firewall operational cost.

Contrary to the first scheme which focuses only on a small set of rules, the *total re-ordering* scheme takes a more aggressive approach and performs a total re-ordering of the rule set based on the current traffic characteristics. This re-ordering is achieve based on a priority assignment which takes into consideration, not only the frequency at which the rule is invoked, but equally importantly the rule size. More specifically, the priority of rule, R_i, can be expressed as: $Pr(R_i) = \frac{hit\_count(R_i)}{size(R_i)}$. Notice that ordering firewall rules based on the above priority assignment achieves the lowest expected cost. We provide a proof of this in [8].

The *default proxy* is the third scheme and is based on the observation that, during traffic inspection, the default deny action is heavily invoked, in comparison to actions resulting from other rules. In a list-based firewall, the default deny action is "enforced" when a packet fails to match any of the rules within a rule set. A relatively high hit ratio of the default deny action is, therefore, bound to increase considerably the overall operational cost of the firewall. The main reason for this increase is that, before a default deny action is enforced and the packet is dropped, all rules in a rule set have to be examined. This is mainly caused by the absence of any representation of the default deny action in the rule set. This, in turn, suggests that the addition of drop rules may alleviate the problem. Adding drop rules, however, brings about several issues to be addressed, including how may rules must be created, what values should be be associated with these new drop rules and what should be their priorities.

The *default proxy* scheme addresses these issues by creating a set of drop rules. The field values of these rules are derived from the corresponding fields of the packets dropped by the default deny action. Initially, the fields of a drop rule are set to *any*, except for the action field which is set to *drop*. The drop rule can be represented as:

$$<\Phi^1 : any;\ \Phi^2 : any;\ \cdots, \Phi_n : any;\ action = drop>$$

As packets are dropped by default deny rule, the values of the drop rule are set to the values of corresponding fields of the dropped packets. This corresponds to the hit rate of the drop rule. The priority each newly created drop rule is computed based on its hit rate and its size in a similar manner as in total re-ordering.

The *online adaptation* scheme encompasses two basic mechanisms: *profile based re-ordering* and *anomaly detection and countermeasure*. Profile based re-ordering uses traffic characteristics to build a long-term rule hit profile, offline. The approach used to build this profile exploits traffic variability[9]. The resulting rule hit profile is then used to detect long and short term anomalies and adapt the rule set accordingly.

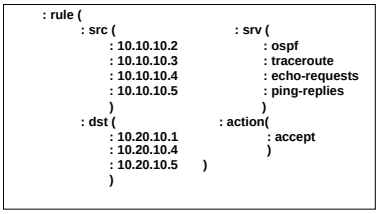

Fig. 2: Rule Structure

The basic idea of *anomaly detection and countermeasure* is to compare the short term traffic pattern with a long term traffic profile. The later is used to optimize the firewall rules. If a significant discrepancy exists between the short term traffic pattern and long term profile, and this discrepancy can result in bad predicted performance, the rules are adjusted as a countermeasure against anomalies. Adjusting the rules entails rule re-ordering and adding explicit reject rules.

Note that anomalies can be either transient or long-lived. If the anomaly analysis reveals a potential performance hazard, a temporary re-ordering of rules is performed. If a given anomaly occurs consistently then it is absorbed into the long term profile. The same anomaly detection and countermeasure procedure is also applied to the default deny rule. Depending on any potential performance hazard created by a default deny rule, a temporary default deny rule is added to the short term profile. If the pattern is repetitive then the new default deny rule is added to the rule set based on its priority and hence absorbed into the long term profile.

V. Experimental Study

The first part of this section briefly describes the data used in this performance study. The second part provides an analysis of the dataset prior to optimization.[3] The last part describes the metric used to assess the performance of the proposed optimizer, and discusses the results of the evaluation study.

A. Data set description

The data set used in the experimental study is obtained from a list-based firewall managed by a large Tier-1 ISP for its partner networks. The Tier-1 ISP provides secure access to and from about 50 business partners. The data set consists of firewall rule sets and traffic logs.

Each rule set consists of several thousand rules. Each rule is a multi-dimensional structure of tuples. A rule set contains on average one million tuples. Figure 2 shows an instance of a rule structure. The dimensions of the rule include the source address, src, the destination address, dst, the service types, srv, and the action. Each dimension contains multiple values. An instance of a tuple of this rule is $< src : 10.10.10.2;\ dst : 10.20.10.1;\ srv : ospf; action : accept >$. Figure 3 depicts an entry of the firewall traffic log. The firewall logs one entry per session.

B. Data set analysis

The first study of the dataset measures the accept and drop hit rates. It is widely believed that drop rules, especially the default deny reject rule, are the ones which contribute the most

[3]Since the exact numbers of flow rate or hit count are considered proprietary information, we omit showing the real values on the y-axis of the figures

num;date;time;orig;type;action;alert;i/f_name;i/f_dir;
product;src;dst;s_port;service;proto;……..

1;27Jul2005;23:59:04;10.10.10.1;log;accept;;qfe1;
inbound;X;10.30.10.1;10.20.10.1;53480;161;udp;;;

Fig. 3: Traffic Log Instance

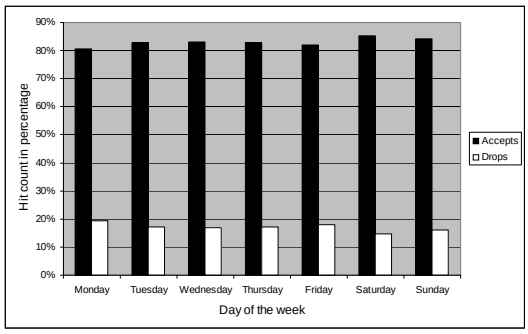

Fig. 4: Accepts vs. Drop Statistics

to the operational cost of the firewall. Contrary to this belief, the results depicted in Figure 4 show that the accept hit rate is on average 3.5 times higher than the drop hit rate. These results suggest that both accept and drop rules should be considered in the firewall optimization.

The second study analyzes the distribution of rule hits based on traffic characteristics. Figures 5 and 6 show the distribution of the top ten hit rules over a weekly and daily period, respectively. The results also show that heavy hit rules are not appropriately ranked in the rule set. This suggests that, in order to reduce the operational cost of the firewall, heavy hit rules, which have lower ranks, should be assigned higher ranks. Upon reordering, rule 3 and 37 must be ranked 1 and 2, as they have the highest hits.

To understand the severity of the default deny hits, an analysis of the hit distribution over hours for a typical day of a given rule set is conducted. Results in Figure 7 show that the default deny rule contributes heavily to the operational cost of

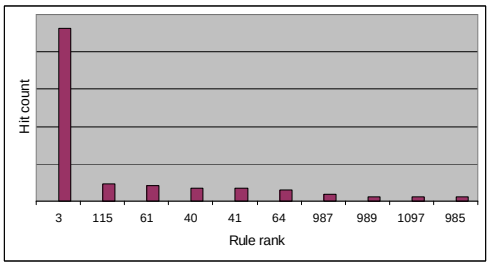

Fig. 5: Rule Hit Distribution: Over One Week

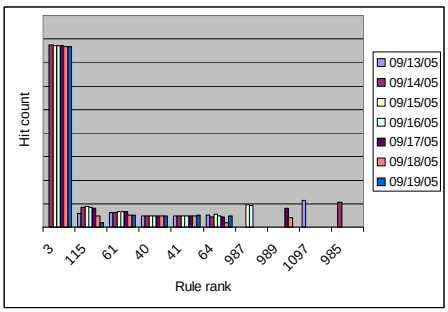

Fig. 6: Rule Hit Distribution: Over Days

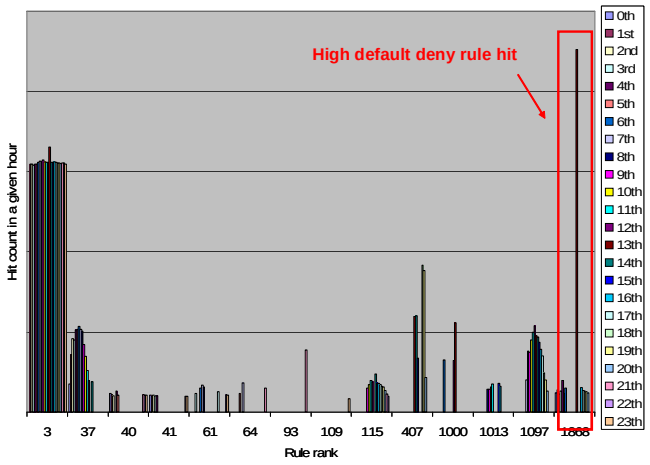

Fig. 7: Default Deny Rule Hits

the firewall. Moreover, a closer look at 3 month firewall data reveals that on average about 65% of the tuples are consistently repeated in the default deny hit tuples. This shows that there is a large number of consistent default deny hit tuples, which could be placed earlier in the rule set with the potential to enable considerable cost improvement.

The outcome of the above studies clearly shows that considering traffic characteristics in firewall optimization is crucial to achieve significant improvement in performance.

C. Evaluation study

In this section, we first describe the performance metrics. We then present the performance results of the evaluation study.

1) Performance evaluation metrics: The main factor that affects the performance of a firewall is the processing overhead due to packet inspection. In order to capture the overhead cost incurred by a firewall to process a rule and enforce the security policy, two metrics are defined. The first metric, denoted as *rule_size()*, measures the size of a given rule in terms of the number of bits necessary to determine unambiguously a match between the rule definition and the corresponding fields of a packet under inspection. The assumption underlying the *rule_size()* metric stems from the fact that the complexity of a matching operation is proportional to the size of the rule. Formally, given a rule r, *rule_size(r)* can be defined as:

$$rule\_size(r) = \begin{cases} \sum_{\mathcal{S}_p,\mathcal{D}_p}\{\alpha_1 \times \|s_p\| + \alpha_2 \times \|d_p\|\} \\ +\beta \times N_s \times \{\|Pr_r\| + \|Po_r\|\} \end{cases} \quad (1)$$

where, α_1, α_2, and β are weight parameters, $\mathcal{S}_p$ and $\mathcal{D}_p$ are respectively the set of source and destination prefixes which occur within the definition of the rule, s_p and d_p are the bit representation of the source and destination prefixes, respectively, N_s is the number of services defined within the rule, and Pr_r and Po_r are the bit representation of the protocol and port identifiers, respectively.

The second metric used in our experimentation is the cost of operating on a given rule set. This cost depends on the rule's rank and size, and on how often the rule is invoked by the firewall. Formally, given a set of rules $r_1, r_2, \ldots, r_k$, the cost of a given rule, r_i, $cost(r_i)$, is defined as follows:

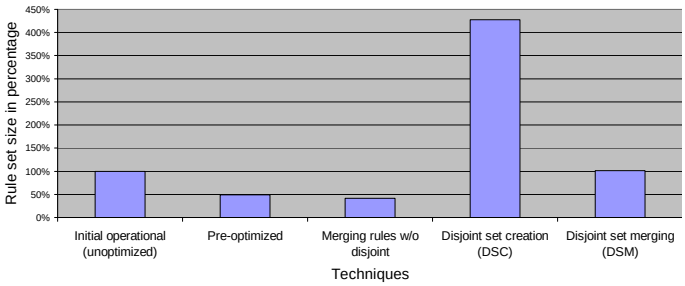

Fig. 8: Rule Set Based Optimization: Size-based

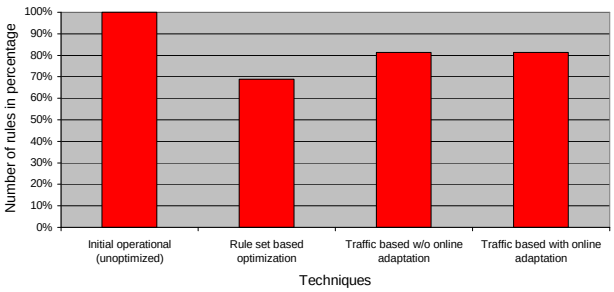

Fig. 9: Traffic Based Optimization: Size-based

$$cost(r_i) = hit\_count(r_i) \times \sum_{\forall r_k \in \mathcal{P}r_i} rule\_size(r_k) \quad (2)$$

where, $\mathcal{P}r_i$ is the set of r_i's predecessors in the list-based set of rules.

Using the above metrics, the aim of optimization is to reduce the rule set size and consequently the processing time of the rule set. This in turn reduces the overall firewall operational cost.

2) Performance evaluation results: In order to evaluate the impact of the various optimization strategies on the firewall performance, an experimental simulation-based study is conducted. The simulation was run on SunOS 5.8 over a Sun-Fire-15000. Results show that the optimization strategies lead to considerable firewall performance improvement.

3) Rule set based optimization: The results depicted in Figure 8 show that the final rule set size after optimization is similar to the size of the initial rule set, but most importantly, the rules in the resulting rule set are all disjoint. This provides the system administrator full flexibility to re-order the rules based on traffic characteristics, as appropriate.

4) Traffic based optimization: In this experiment, traffic based optimization are applied to the firewall rule set. Results in Figure 9 demonstrate a significant decrease in the number of rules. More specifically, the results show that nearly 20% of initial operational rules are eliminated.

The final experiment is aimed at evaluating the impact of the various optimization strategies on the firewall operational cost. The results depicted in Figure 10 indicate that the optimization strategies, applied to the pre-processed dataset, result in reducing the initial operational cost to around 6.3%.

The results clearly indicate that the proposed traffic-aware optimization strategies have great potential to significantly improve the performance of firewalls and reduce their operational cost. A more extensive analysis of the schemes and experimental results can be found in [8].

VI. Conclusion and Future Work

This paper studies the problem of firewall optimization in detail. It is the first effort in using firewall traffic log information to design and optimize firewall rules sets. Both rule set based and traffic based optimizations are integrated in our firewall accelerating tool. The paper also introduces a novel adaptive anomaly detection/countermeasure mechanism to deal with short term and long term anomalies. We have started our efforts to validate the size and cost metrics and the optimization results. Furthermore, we are working on an efficient

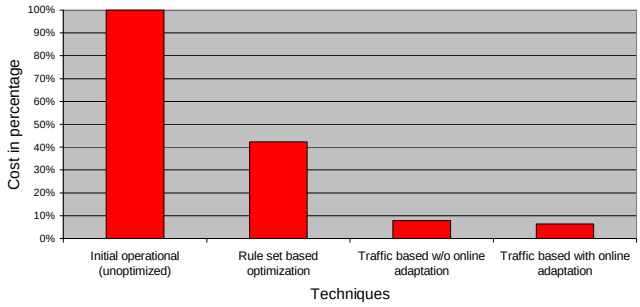

Fig. 10: Traffic Based Optimization: Cost-based

implementation for the algorithms to reduce the processing overheads of optimizations in the existing prototype. As future work we intend to use other ISP datasets and firewalls to study, optimize and validate our approaches. We would also be extending our optimization ideas on other types of firewalls (not only list based ones). We believe this paper is the first step in the design of a complete accelerating toolkit for firewall optimization.

References

[1] T. V. Lakshman and D. Stidialis, "High speed policy-based packet forwarding using efficient multi-dimensional range matching," in *In Proceedings of SIGCOMM*. ACM Press, 1998.

[2] V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," in *In Proceedings of SIGCOMM*. ACM Press, 1999.

[3] E. Al-Shaer and H. Hamed, "Modeling and management of firewall policies," *IEEE Trans. Network and Service Management*, vol. 1, no. 1, Apr 2004.

[4] P. Eronen and J. Zitting, "An expert system for analyzing firewall rules," in *Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec 2001)*, Copenhagen, Denmark, Nov. 2001, pp. 100–107.

[5] E. W. Fulp, "Optimization of network firewalls policies using directed acyclic graphs," in *Proceedings of the IEEE Internet Management Conference*, 2005.

[6] J. Qian, S. Hinrichs, and K. Nahrstedt, "ACLA: A framework for access control list (acl) analysis and optimization," in *Communications and Multimedia Security*, 2001.

[7] M. Condell and L. Sanchez, "On the Deterministic Enforcement of Unordered Security Policies ," BBN Technical Memorandum No. 1346, April 26, 2004.

[8] S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greenberg, "A Traffic-Aware Framework and Optimization Strategies for Large Scale Enterprise Networks," *Technical Report*, pp. 1–20, September 2005.

[9] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning," in *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*. New York, NY, USA: ACM Press, 2002, pp. 91–92.