Ms. Sanky Bai Sathyan.K, Ms. S Swarna Parvathi / IOSR Journal of Engineering (IOSRJEN)
www.iosrjen.org ISSN : 2250-3021
Vol. 2 Issue 2, Feb.2012, pp.197-202

Security Enhancement of Firewall Policies in Virtual Private Network

Ms. Sanky Bai Sathyan.K
Dept. of Information Technology
Sri Venkateswara College of Engineering
Chennai, India

Ms. S Swarna Parvathi
Dept. of Information Technology
Sri Venkateswara College of Engineering
Chennai, India

***Abstract*- Optimizing firewall policies are crucial for improving network performance. The key technical challenge is that firewall policies cannot be shared across domains because a firewall policy contains confidential information and even potential security holes, which can be exploited by attackers. A virtual private network allows roaming users to access some resources as if that computer were residing on their home organization's network. Although VPN technology is very useful, it imposes security threats on the remote network because its firewall does not know what traffic is flowing inside the VPN tunnel. To address this issue, VGuard was proposed, a framework that allows a policy owner and a request owner to collaboratively determine whether the request satisfies the policy without the policy owner knowing the request and the request owner knowing the policy. However two of the main challenges are the increasing number of classification rules, amount of traffic and network line speed. In order to make the above proposed approach better, this paper presents a traffic-aware top-N firewall approximation algorithm as an enhancement to the previous approach. This algorithm is used for selecting the top-N most frequently matched subset of rules from the original ruleset. The goal is to obtain Top-N rules that cover as much traffic as possible while preserving the dependency relationships that an overall higher packet classification throughput can be achieved.**

Keywords - Approximation Algorithms, Network Security, Privacy, Virtual Private Networks.

I. INTRODUCTION

Firewalls are designed to provide access control. Although there is risk associated with any access, by limiting what access is permitted the risk is limited and understood and can be evaluated against business need to effectively justify the risk. However, poor firewall management defeats this purpose by ineffectively controlling access and limiting visibility into what access is actually permitted; poor management also increases the cost associated with security management. The result of poor management is a firewall policy with unnecessary rules that result in excessive complexity, overly permissive access, unnecessary risk and performance degradation, all of which lead to higher costs that can be avoided. These problems can be addressed with both short-term and long-term activities to clean up the firewall now and prevent this situation from returning. This project discusses the implications of firewall policy complexity, why it remains a problem today and how to resolve it.

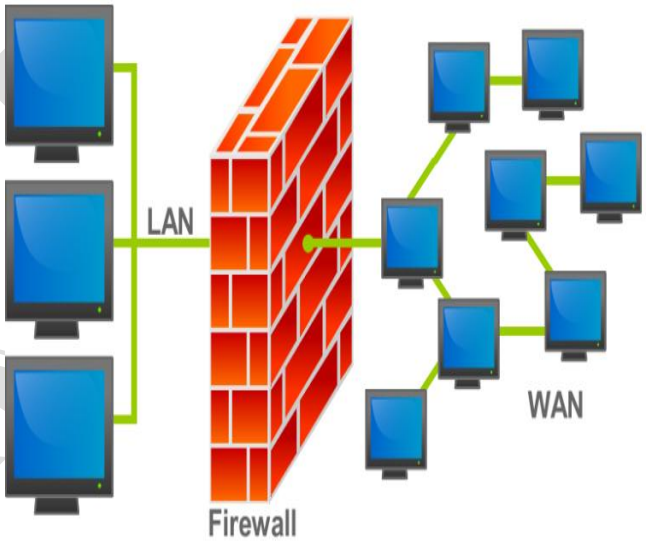

Fig. 1. A typical secure network

As one of the critical network security components, firewall is deployed in virtually all operational networks. A firewall is typically deployed at strategic points of the network such that it will inspect most if not all of the traffic. It is crucial to maintain high classification throughput.

A Virtual Private Network (VPN) is a network that uses primarily Public Telecommunication infrastructure, such as the Internet, to provide remote offices or traveling users access to a central organizational network. VPNs typically require remote users of the network to be authenticated, and often secure data with encryption technologies to prevent disclosure of private information to unauthorized parties. VPNs may serve any network functionality that is found on any network, such as sharing of data and access to network resources, printers, databases, websites, etc. A VPN user

typically experiences the central network in a manner that is identical to being connected directly to the central network. VPN technology via the public Internet has replaced the need to requisition and maintain expensive dedicated leased-line telecommunication circuits once typical in wide-area network installations.

Secure VPNs use cryptographic tunneling protocols to provide confidentiality by blocking intercepts and packet sniffing, allowing sender authentication to block identity spoofing, and provide message integrity by preventing message alteration.

While the VPN tunnel is very useful for the roaming user, it imposes security threats on remote network because its firewall does not know what traffic is flowing inside the VPN tunnel. So that remote network's firewall cannot enforce its policies on the roaming user's computer although that computer is physically on remote network. Thus, the VPN tunnel opens a hole to remote firewall that may allow unwanted traffic to flow in and out. Having such a hole is very dangerous because viruses or worms could flood in through it to the roaming user's computer first and then further spread to other computers on remote network.

The fundamental problem in the above application is: how can we collaboratively enforce firewall policies in a privacy preserving manner for VPN tunnels in an open distributed environment? A satisfactory solution to this problem should meet the following three requirements: (1) The request owner cannot gain any more knowledge on the policy after any number of runs of the protocol than they would by brute force probing of the policy. We refer to this requirement as policy privacy. (2) It should be computationally infeasible for the policy owner to reveal a request. We refer to this requirement as request privacy.

Although this is a fundamentally important problem, it is largely underinvestigated. The state-of-the-art on this problem is the seminal work in [13], where Alex et al. proposed a scheme called VGUARD, is a privacy preserving the framework for collaborative enforcement of firewall policies. In VGuard, it first converts a firewall policy of an ordered list of overlapping rules to an equivalent non ordered set of non overlapping rules, which enables rule shuffling and, consequently, Policy owner cannot identify which original rule matches the given packet. Second, VGuard obfuscates rule decisions, which prevents Policy owner from knowing the decision for the given packet. The main challenges of VGuard are the increasing number of classification rules, the amount of traffic.

The idea is to optimize for rules with higher hit-rates so that an overall higher packet classification throughput can be achieved. A top-N selection is to select the N rules that maximize the total hit-rate achieved by the resulting sub-ruleset A top-N sub-ruleset allows operators to have a small number of top-N rules to be classified at managed switches while minimizing the miss-traffic converge at the controller. Rules that are independent of others (i.e. no conflict), Rules that only depends on other target rules can also be included, as long as their relative order is preserved in the resulting list. For target rules that depend on at least one non-target rule, need to either resolve the conflicts with the dependent rules or include the dependent rules to the top-N list, retaining their relative priorities.

Conflicts can be resolved by splitting the target rule concerned into smaller derived rules that are disjoint with the dependent rules. It is noteworthy that in either way, some target rules, starting from the bottom of the target list, have to be excluded from the sub-ruleset because can only have N rules in the sub-ruleset this unavoidably lowers the overall hit-rate provided by the resulting sub-ruleset

II. RELATED WORK

Secure Function Evaluation (SFE) was first introduced by Yao with the famous "Two-Millionaire Problem" [6]. A secure function evaluation protocol enables two parties, one with input x and the other with y, to collaboratively compute a function f (x, y) without disclosing one party's input to the other. The classical solutions for SFE are Yao's "garbled circuits" protocol [7] and Goldreich's protocol [5]. The method provided by Yao has a computational cost of $O(2^b)$, where b is the number of bits needed to encode x and y. Later, the Secure Function Evaluation problem was generalized to the Secure Multiparty Computation (SMC) problem. Chaum et al. proved that any multiparty protocol problem can be solved if there is an authenticated secrecy channel between every pair of participants [9]. Zero-knowledge protocols [10], [11], [12] also aims to provide the privacy between two parties. A zero knowledge protocol is an interactive method for one party (suppose IBM) to prove to another (suppose MSU) that a statement is true without revealing anything other than the veracity of the statement. Although we could use SFE or SMC solutions to solve this problem as well as the problem of checking whether a value is in a range, the $O(2^b)$ complexity makes such solutions infeasible.

Design and Implementation of Cross-Domain Cooperative Firewall [8] that allows two collaborative networks to enforce each other's firewall rules in an oblivious manner. In CDCF, when a roaming user establishes an encrypted tunnel between his home network and the foreign network, the tunnel endpoint (e.g., a VPN server) can regulate the traffic and enforce the foreign network's firewall rules, without knowing these rules. The key ingredients in CDCF are the distribution of firewall primitives across network domains, and the enabling technique of efficient oblivious membership verification. They have implemented CDCF and integrated it with the Open VPN software, and evaluated its performance using extensive experiments. Their results show that CDCF can protect the foreign network from encrypted tunnel traffic with minimal overhead.

VGuard[13], a framework that allows a policy owner and a request owner to collaboratively determine whether the request satisfies the policy without the policy owner knowing the request and the request owner knowing the policy. We first present an efficient protocol, called Xhash, for oblivious comparison, which allows two parties, where each party has a number, to compare whether they have the same number, without disclosing their numbers to each other. Then, we present the VGuard framework that uses Xhash as the basic building block. The basic idea of

VGuard is to first convert a firewall policy to non-overlapping numerical rules and then use Xhash to check whether a request matches a rule.

In VGuard, different from CDCF, the policy owner does not know which rule matches which request; thus, it makes the selective policy updating attacks infeasible. Furthermore, unlike CDCF, VGuard obfuscates rule de- cisions, which prevents MSU from knowing the decision for the given packet. To make VGaurd efficient, we pro- pose a new oblivious comparison scheme, called Xhash, which uses XOR and secure hash functions. Xhash is three orders of magnitude faster than the commutative encryption scheme used in CDCF. Moreover, VGuard uses decision diagrams to process packets, which is much faster than the linear search used in CDCF. By side by side comparison, our experimental results show that VGuard is is 552 times faster than CDCF on MSU side and 5035 times faster than CDCF on IBM side.

M.G Gouda et al. [4] proposed a new method called structured firewall design To achieve consistency, completeness, and compactness, which consists of two steps. First, one designs a firewall using a firewall decision diagram instead of a sequence of often conflicting rules. Second, a program converts the firewall decision diagram into a compact, yet functionally equivalent, sequence of rules. This method addresses the consistency problem because a firewall decision diagram is conflict-free. It addresses the completeness problem because the syntactic requirements of a firewall decision diagram force the designer to consider all types of traffic. It also addresses the compactness problem because in the second step they use two algorithms (namely FDD reduction and FDD marking) to combine rules together, and one algorithm (namely firewall compaction) to remove redundant rules.

Another direction of research aims at optimizing ruleset for smaller sizes. While these optimization can be very useful for individual firewalls, they do not allow efficient sub- ruleset selection.

Traffic-aware firewall optimization and reordering [2], [1], [3] all face the challenge of dependencies among rules. A common tactic in solving the dependencies is to pre- process the originally ruleset to obtain a totally disjoint ruleset (one that has no rule overlapping with each other in their matching space) and then merge the rules after the proposed optimization. While this approach satisfies R1, the resulting disjoint ruleset will be so big that it will either burden traffic monitoring by requiring hit-counts on a large number of rules (fails R3) or it can only partially satisfy R4 by adapting to historically measured traffic patterns through off- line computation (fails R2).

III. DESIGN

The main objective of this project is improving performance of VGuard by selecting top N rules for matching instead of matching all rules. This is the method to optimize for rules of with higher hit-rates so that an overall higher packet classification throughput can be achieved. Given the complexity of Top-*N* selection problem, this method proposed an algorithm to solve it efficiently. i) The Top-*N* target list is constructed by choosing the *N* rules with highest hit-rates followed by reordering them in descending order of priority. ii) Starting from the highest priority rule, a partition decision algorithm makes a decision for each dependent rule to either modify the rule to resolve the dependency or to include the dependent rule in the Top-*N* list. Depending on the overlapping pattern of two rules, resolving their dependency usually results in additional rules. The different modules of the proposed system are addressed in the following sub topics.

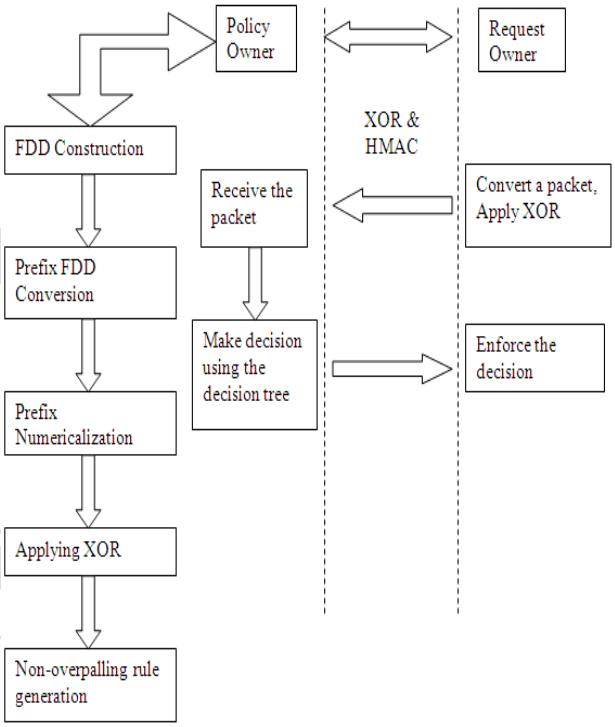

Fig. 2. Block diagram

A. Bootstrapping and filtering protocol

In the bootstrapping protocol, policy owner first converts its firewall policy to a set of non-overlapping prefix rules. Second, policy owner converts each prefix to a number. Third, policy owner applies an XOR operation to every number using its secret key K1. Finally, policy owner sends the anonym zed policy to request owner. Request owner then applies XOR and HMAC operations to every number in the received policy using its secret key K2, obfuscates the decision of each rule, and shuffle the resulting rules. To complete the process, request owner sends the resulting policy back to the policy owner. Converting a firewall policy to a set of non-overlapping prefix rules consists of four steps: FDD construction, range conversion, prefix numericalization, and rule generation.

In the filtering protocol, each time request owner receives a packet that originated from or was sent to its

representative. Request owner first converts the packet to prefixes and further converts each prefix to a number using the same prefix numericalization scheme.

Then the request owner XOR every number in the packet with its key K2, and sends the resulting packet to the third party. The third party further applies XOR and HMAC to the received packet with the secret key K1. Note that the third party and the policy owner shares key K1. Further, the third party sends the resulting packet to the policy owner. Policy owner then searches obfuscated decision for the packet using the received firewall policy from the request owner in the bootstrapping protocol. Finally, policy owner sends the obfuscated decision to request owner and request owner finds the original decision using its decision obfuscation table.

B. Xhash protocol and packet inspection

This module presents the simple and efficient Xhash Protocol to achieve oblivious comparison. The working of the Xhash protocol is as follows: first, policy owner sends N1⊕K1 to request owner then, request owner computes HMAC k (N1⊕ K1⊕ K2) and sends the result to the policy owner. Second, request owner sends N2⊕K2 to the policy owner. Third, policy owner computes HMAC k (N2 ⊕ K2⊕ K1) and compares it with HMAC k (N1⊕ K1⊕ K2),which was received from the request owner. Finally, the condition N1 = N2 holds if and only if HMAC k (N2 ⊕K2 ⊕ K1) = HMACk (N1 ⊕ K1 ⊕ K2).

The above function HMAC is a keyed-Hash Message Authentication Code, such as HMAC-MD5 or HMACSHA1, which satisfies the one-wayness property (i.e., given $HMAC_k(x)$, it is computationally infeasible to compute x and k) and the collision resistance property (i.e., it is computationally infeasible to find two distinct numbers x and y such that $HMAC_k(x) = HMAC_k(y)$. Note that the key k is shared between policy owner and IBM. Although hash collisions for HMAC do exist in theory, the probability of collision is negligibly small in practice. Furthermore, by properly choosing the shared key k, can safely assume that HMAC has no collision.

Packet inspection is proposed with the combination of both Bootstrapping Protocol and the Filtering Protocol. The basic idea is that policy owner and request owner apply Xhash protocol to each character of every string in the signature database and each character of the packet payload, and check whether the resulting packet payload contains the resulting string.

C. Traffic aware Top-N approximation alogorithm

This module deals with the enhanced approach of the proposed system that is the traffic aware Top-N firewall approximation Algorithm. The goal is to obtain Top-N rules that cover as much traffic as possible while preserving the dependency relationships. This algorithm is composed of the following steps. Initially, reorder and choose the top N rules and then resolve the dependency and partition of the Top N list is taken place. Finally, evaluate the proposed approach of VGuard framework based on the Traffic aware top N approximation algorithm with the exiting approach.

IV. IMPLEMENTATION AND RESULTS

The various modules of the system have been implemented. The implementation results are discussed in the following sub topics.

A. Non overlapping rules generation

Figure 3 shows the request owner (Client) window. In the bootstrapping protocol, the policy owner (Server) first converts its firewall policy to a set of non-overlapping prefix rules. Second, policy owner converts each prefix to a number. Third, policy owner applies an XOR operation to every number using its secret key K1. Finally, policy owner sends the anonym zed policy to request owner.

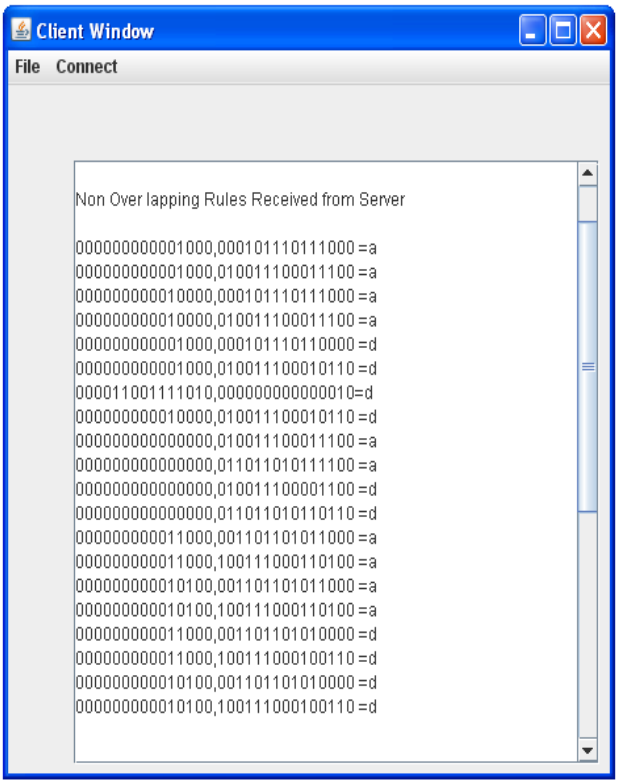

Fig. 3. Non overlapping rules generation

B. Decision obfuscates rules generation

Figure 4 shows the Policy owner (Server) window. The request owner applies XOR and HMAC operations to every number in the received policy using its secret key K2, obfuscates the decision of each rule, and shuffles the resulting rules. To complete the process, request owner sends the resulting policy back to the policy owner.

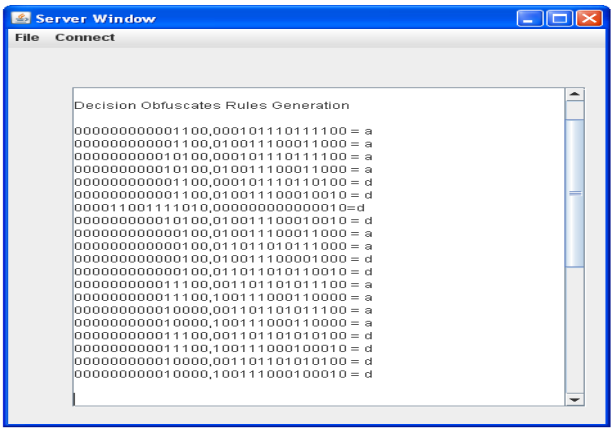

Fig. 4. Decision obfuscates rules generation

C. Filtering Protocol at Client

Figure 5 shows the filtering process at Client side. In the filtering protocol, each time request owner receives a packet that originated from or was sent to its representative. Request owner first converts the packet to prefixes and further converts each prefix to a number using the same prefix numericalization scheme. Then the request owner XOR every number in the packet with its key K2, and sends the resulting packet to the third party.

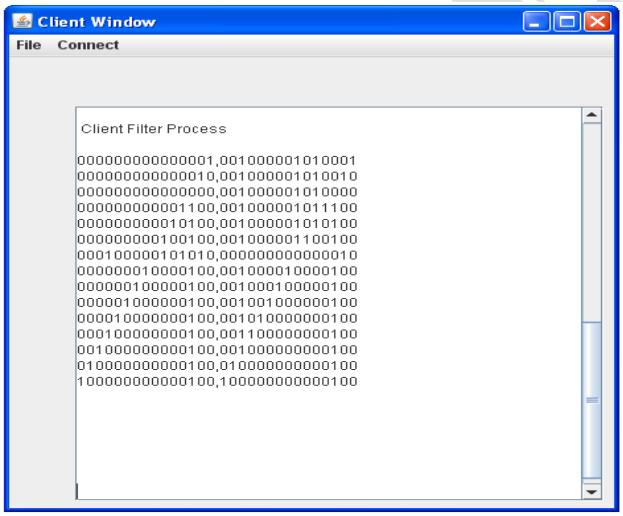

Fig. 5. Filtering process at Client window

D. Filtering protocol at Third party

Figure 6 shows the filtering process at third party window. It will apply XOR and HMAC to the received packet with the secret key K1. Note that the third party and the policy owner shares key K1. Further, the third party sends the resulting packet to the policy owner.

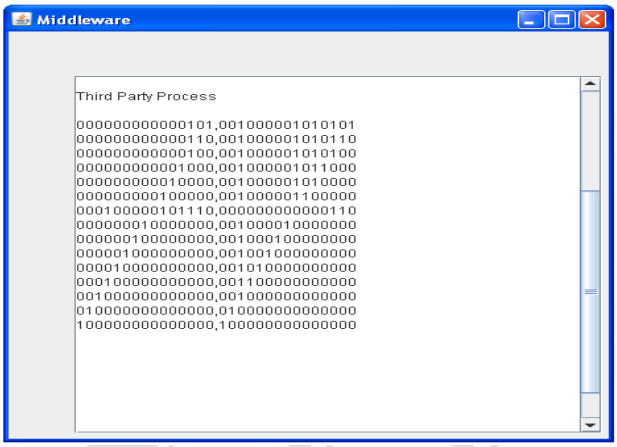

Fig. 6. Filtering process at Third party window

E. Filtering protocol at Server

Figure 7 shows the filtering process at Server side. Once the Policy owner receives the packet from the third party, then server searches obfuscated decision for the packet using the received firewall policy from the request owner in the bootstrapping protocol. Finally, policy owner sends the obfuscated decision to request owner and request owner finds the original decision using its decision obfuscation table.

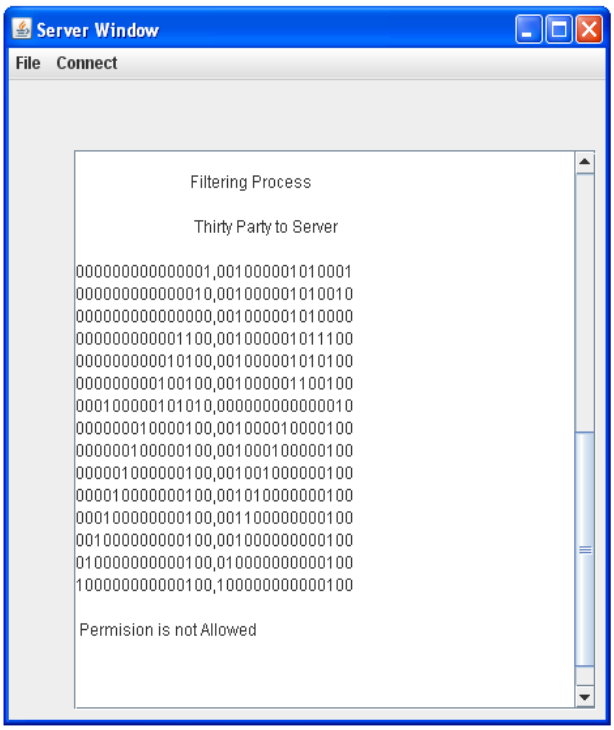

Fig. 7. Filtering process at Server window

F. Hit-rates

This difference between the existing and proposed system decreases as the size of Top-N grows. Especially, when the size of Top-N list and the ruleset is equal, both system give an identical Top-N list, since in such case, the job for them is just to reorder the ruleset in priority descendent order.

G. Running time

This will record the running time of both existing and proposed system on different size of ruleset, ranging from 100 to 1000. The experiment on the running time reflects performance of two systems. And will show that proposed system always take lesser time than existing system.

V.CONCLUSION AND FUTURE WORK

This paper suggested four basic requirements to a top-N sub-ruleset selection problem. An optimization framework has been proposed and the challenges of the problem are identified. This algorithm is proposed to choose rules among those with the highest hit-rates, their associated dependencies and derived rules that have their dependencies resolved. The algorithm does not require conflict-free ruleset to be pre-computed. The required hit-rate statistics are readily available with little overhead from most managed switches, routers and firewalls. The simulations show that the top-N approximation algorithm achieves cumulative hit-rate is reasonably close to the optimal. The running time is in the order of seconds and thus is able to respond to dynamic changes in traffic pattern.

The future work intends to improve the security among the third party and the policy owner. For that Trap door commitment strategy is used to find output with honesty and without any assumption. By using this method the security will be increased.

REFERENCES

[1] S. Acharya, M. Abliz, B. Mills, T. F. Znati, J. Wang, Z. Ge, and A. Greenberg, "OPTWALL: A hierarchical traffic-aware firewall," in *NDSS '05: Proc. of the 12th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, 2005.

[2] H. Hamed and E. Al-Shaer, "Dynamic rule-ordering optimization for high-speed firewall filtering," in *ASIACCS '06: Proc. of the 2006 ACM Symp. on Inform., comput. and commun. security*.New York, NY, USA: ACM, 2006, pp. 332–342.

[3] S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greenberg, "Traffic- aware firewall optimization strategies," in *ICC '06. IEEE International Conference on Communications, 2006.*, vol. 5, June 2006.

[4] Mohamed G. Gouda and Alex X. Liu. Structured firewall design. *Computer Networks Journal (Elsevier)*, 51(4):1106–1120, 2007.

[5] Silvio Micali Oded Goldreich and Avi Wigderson. How to playany mental game. In *Proc. ACM Conf. on Theory of computing*, 1987.

[6] Andrew C. Yao. Protocols for secure computations. In*Proc. IEEESymposium on the Foundations of Compute Science(FOCS)*, pages160–164, 1982.

[7] Andrew C. Yao. How to generate and exchange secrets. In *Proc.IEEE Symposium on Fundations of Computer Science (FOCS)*, pages 162–167, 1986.

[8] Jerry Cheng, Hao Yang, Starsky H.Y. Wong, and Songwu Lu. Design and implementation of cross-domain cooperative firewall.In *Proc. IEEE Int. Conf. on Network Protocols (ICNP)*, 2007.

[9] David Chaum, Claude Crepeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proc. ACM Symposium onTheory of Computing*, pages 11–19, 1988.

[10] Fabrice Boudot. Efficient proofs that a commited number lies in an interval. In *Proc. Advances in Cryptology (EUROCRYPT), volume1807 of Lecture Notes in Computer Science*, May 2000.

[11] Ronald Cramer, Matthew K. Franklin, Berry Schoenmarks, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Proc. Advances in Cryptology (EUROCRYPT), volume 1070of Lecture Notes in computer Sceince*, 1996.

[12] Wenbo Mao. Guaranteed correct sharing of integer factorizationwith off-line shareholders. In *Proc. Public Key Cryptography (PKC),volume 1431 of Lecture Notes in Computer Science*, February 1998.

[13] Alex X. Liu and Fei Chen." Privacy Preserving Collaborative Enforcement of Firewall Policies in Virtual Private Networks*".:Proc.of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*. Michigan State Univ., East Lansing, MI, USA, :IEEE,2011,pp. 887-895