Increasing SIP Firewall Performance by Ruleset Size Limitation

Sven Ehlert
Fraunhofer FOKUS, Berlin, Germany
sven.ehlert@fokus.fraunhofer.de

Ge Zhang
Karlstads Universitet, Sweden
ge.zhang@kau.se

Thomas Magedanz
Fraunhofer FOKUS, Berlin, Germany
thomas.magedanz@fokus.fraunhofer.de

***Abstract*— To protect SIP communication networks from attacks, especially flooding attacks like Denial-of-Service or message spam, Intrusion Detection Systems (IDS) are deployed at the ingress point of the network to filter potential malicious traffic. A key issue of IDS performance is the operation of its firewall to block malicious user requests. Depending on the complexity of the firewall ruleset, filtering performance of the IDS can decrease considerably during high-load flooding situations. In this paper we propose a scheme to increase IDS firewall performance by merging several similar rules into more general ones and ignoring lesser relevant rules to limit the number of firewall rules. We formalise a mathematical model to compute new firewall rules and show exemplary with traffic from SIP VoIP communication networks how the calculation can be performed. If applied to a VoIP IDS, the scheme can increase firewall thoughput considerably, while retaining most of its effectiveness.**

I. Introduction

The Session Initiation Protocol (SIP) [1] is gaining ever more popularity, as it is the key protocol for most VoIP services in the Internet and next generation networks like the Internet Multimedia Subsystem (IMS) [2]. As such, it becomes ever more attractive for attackers [3] [4], who can launch e.g., Denial-of-Service attacks to disturb the service, annoy users by initiating multiple spam sessions or steal credentials to misuse the service (call theft and fraud).

Protecting the service thus becomes of utmost importance for the provider. Session Border Controllers (SBC) have been established as all-in-one advanced protection solutions for SIP-based networks. Besides SBCs, dedicated security solutions for SIP networks exit [5] [6] [7]. The key component of such an advanced security solution is a *SIP-aware firewall*, that can take SIP features into account. By examining the message payload i.e., SIP header fields, fine-grained security policies can be defined. However, due to the additional complexity of such a SIP firewall, its performance (i.e., throughput capabilities) can decrease, depending on the size of firewall ruleset and the amount of traffic to be processed.

In this paper we address such performance penalties of SIP-aware firewalls. Our goal is to increase message processing capability of the firewall by limiting the ruleset size to a fixed value. An *optimiser* component is employed to reduce the original firewall ruleset (R_i) of possibly unlimited size to a new ruleset (R_o) of fixed size, which will actually be applied at the firewall. The optimisation process is performed by *merging* several similar rules into one new rule, and by *dropping* rules with lesser security impact. As this process causes an inaccuracy in the new ruleset R_o, a metric is defined to minimise it.

In the remaining sections we address the problem space and scope, and introduce our ruleset optimiser solution. The algorithm is explained with examples and operation is visualised with real-life VoIP traffic traces. We present other relevant work in this direction and outline some open issues.

II. Background Information

A. Voice over IP with the Session Initiation Protocol

SIP [1] is a text-based protocol designed to establish or terminate a session between two or more partners. The message format is similar to the HTTP protocol, with message headers and corresponding values e.g., `From: user@sip.org` to denote the sender of a message. Several message types are defined (e.g. REGISTER, INVITE, ACK, BYE, ...) and encoded in the first line of each message (Request-URI). Message headers include e.g., From, To, Call-ID, or User-Agent.

A SIP VoIP network consist of several entities, including *User Agents* (UA) that generate or terminate SIP requests, *Registrars*, where users log in and announce their availability in the SIP network and *Proxies* that forward requests in the appropriate SIP networks. Several proxies can be deployed in a SIP infrastructure e.g., outbound proxies that regulate routing outgoing traffic from one network to a foreign network and incoming proxies that handle all incoming SIP requests possibly enforcing additional security checks.

B. Firewalls

A firewall is a network security component deployed between networks of different trust levels. Located at the network ingress and possibly at the egress point, it examines all in- and outgoing traffic which is then forwarded or dropped according to security policies. Stallings [8] defines three common firewall types: 1) A *packet filter* inspects packets which represent the basic unit of data transfer on the network level. Packet filters operate solely at the IP level. 2) An *Application-Layer Gateway* (ALG) works as a relay on the application level. Additional to a packet filter it has application-layer knowledge of certain protocols (common examples are DNS, HTTP, or FTP) and applies its security policy rules also on the application level. A SIP-aware firewall is thus an ALG with SIP knowledge. Note, that an SIP ALG needs to implement

978-1-4244-2644-7/08/$25.00 ©2008 IEEE

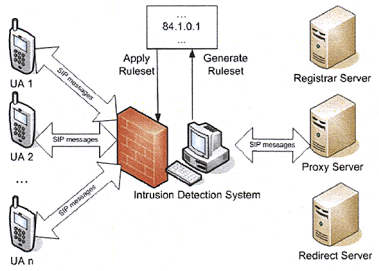

Fig. 1. IDS protected SIP VoIP network

a SIP text parser - while IP fields can be accessed directly due to its fixed binary format, SIP uses free-form text. After parsing, a SIP ALG could apply security policy rules based on SIP message header fields e.g., drop packets with a certain transaction ID or user agent identification. 3) A *Circuit-Level Gateway* works on the session level and hides information of protected networks, a common example are web proxies.

An essential component of both the packet filter and application-layer gateway is the actual definition of the security policy, which generally consist of a list of *rules* in the form "if condition then action", where condition can be e.g., a comparison to an IP address or port and action is either accepting or discarding. All defined rules at the firewall form its *ruleset*. Rules can be static e.g., the network security operator has defined these rules manually, or updated dynamically by a firewall controller - commonly an Intrusion Detection System (IDS). At each passing packet, the firewall searches the ruleset for applying rules (the condition *matches* the examined packet), and performs the defined action on the packet.

Firewall rules can be distinguished between *single-mapped* and *multi-mapped* rules. A single-mapped rule matches exactly one entity e.g., an IPv4-address condition for address 84.1.0.1 would only match traffic coming from exactly this address. On the other hand, rules utilising pattern to match multiple entities are called multi-matched rules. For example by using regular expressions, the IPv4 address condition 84.1.0.* would match traffic in the IPv4 range from 84.1.0.1 to 84.1.0.254 (excluding network and broadcast address).

III. Problem Space and Scope

Protecting a VoIP network at the ingress point will be a crucial task in the future. With advanced flooding attacks like DoS or spam call generation, an IDS needs to be deployed to provide attack mitigation features (figure 1). The IDS analyses all passing traffic and dynamically updates the network firewall in realtime to cope with immediate attacks [9]. To detect such advanced attacks it needs sophisticated detection algorithms, which are a topic of ongoing research [10] [11].

However, this setup poses a threat for a *self-inflicting DoS*. Given a high load of malicious traffic to the service, the IDS will continuously add new rules to the firewall to prevent this traffic from reaching the service. As the size of the firwall ruleset increases, the performance of the firewall will decrease, due to two factors: Searching for matching rules becomes more complex as the firewall ruleset grows. And, especially for SIP ALGs, each passing SIP message need to be parsed before a decision can be made. For example, we show in [5], that the throughput rate of a firewall decreases with an increasing number of rules. Given a constant traffic flooding rate of about 170 Mbit/s, the rate drops to 130Mbit/s after the introduction of 100 IP-layer firewall rules, and drops further to 70 MBit/s after the insertion of SIP ALG rules. Other authors witness similar decrease in performance of protection systems [12] [13].

Hence, security solutions are faced by the following dilemma, which holds true especially in high-traffic scenarios: *If no flooding protection is applied, the service's performance will degrate due to the attack; however, applying IDS protection might also degrate service's performance due to limited firewall throughput rates.*

There a several possibilities to address this conflict in high-flooding scenarios: 1) Define more *intelligent protection algorithms* for the IDS. Ideally, the protection algorithm should generate as few firewall rules as possible but still provide effective protection. This would increase the complexity of the detection scheme and hence introduce a new performance penalty. Additionally, algorithm optimisations have to be developed separately for each new threat. 2) Design and implement a *high-performant firewall*. Different designs for firewalls exist [14] to increase firewall performance. However, adding additional features like SIP ALG processing, reduces performance again. 3) *Limit the number of rules* at the firewall, independent of the used IDS detection algorithm. This is the scope of this work in the context of flooding attacks. By applying a maximum size limit of the ruleset, a certain operating level of the firewall is guaranteed.

Each of the solutions have certain advantages and drawbacks. Most importantly, there is likely no ultimate solution that allows *optimum security* together with *maximum performance*. Hence, a solution to this problem is likely a tradeoff between these two factors.

IV. Firewall Ruleset Optimiser

A. Algorithm Overview

To reduce the size of the ruleset at the SIP firewall we define σ_{max} as an upper maximum bound for the number of rules the SIP firewall will accept. Its value depends on the performance of the used firewall and on the traffic scenario of the VoIP network. A higher performant firewall can tolerate a higher value for σ_{max}. The value can e.g., determined by stress tests of the firewall. The value should be set in such a way that whenever $|Ruleset| \leq \sigma_{max}$ (where $|Ruleset|$ indicates the size of the firewall's rules), the firewall will yield acceptable performance for the defined setup and scenario.

We introduce a *ruleset optimiser* which constantly translates the input ruleset R_i, generated by an IDS, to the output ruleset R_o, which strictly satisfies the requirement $|Ruleset| \leq \sigma_{max}$. R_o will then be applied at the firewall (See figure 2).

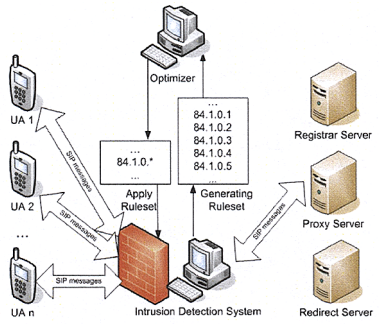

Fig. 2. The optimiser re-constructs the ruleset generated by IDS in favour of the throughput of IDS

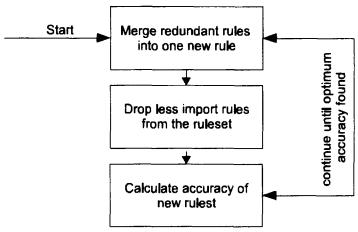

Fig. 3. Overview of the ruleset optimiser

By limiting the maximum ruleset size, the firewall's effectiveness will likely degrate by a certain degree e.g., rules might not be deployed at the firewall because of the maximum size has been reached. We accept this lesser accuracy for flooding scenarios as an unwanted, but necessary sidecondition, as this at least provides partial service protection, in comparison to no protection at all. Also, for DoS attacks, it is acceptable to let some DoS packets pass through, as they will only increase the load at the proxy, but not degrate its performance, if most other packets will be blocked.

The basic idea of our algorithm depends on the three steps *ruleset merging*, *rule dropping*, and *accuracy calculation for optimisation*, as seen in figure 3.

Firstly, we assume that most rules generated by the IDS will be single-mapped. For example, considering IP addresses, there will be IPv4 address field ruleset conditions for entries "81.1.0.1", "94.7.56.1", and "81.1.0.79". Looking at SIP vendor names there might be SIP user-agent header field conditions like "Cisco ATA 186 v2.16.1 ata18x", "Twinkle/1.1", and "Cisco ATA 186 v2.16.2 ata18x". We merge several single-mapped rules into one multi-mapped rule using a redundancy metric.

Secondly, if after this step $|R_o| > \sigma_{max}$, additional rules will be dropped until $|R_o| \leq \sigma_{max}$. We define a metric to select more relevant rules for merging and less relevant ones for dropping.

Using both ruleset merging and dropping, the accuracy of the ruleset changes during the transition. For example, by combining multiple single-mapped rules into one multi-mapped rule, additional targets might be included in the matching ruleset. Contrary, by dropping rules from the ruleset, targets are excluded from the matching ruleset. Hence, we finally apply a metric to calculate a minimum change in accuracy from R_i to R_o.

To meet both performance and security requirements, the optimiser has to ensure, that

1) $|R_o| \leq \sigma_{max}$ during operation, and
2) the accuracy change through the generation of R_o from R_i is minimal.

B. Algorithm Details

1) Redundancy Metric: Within the merging step, multiple single-mapped rules will be merged into one multi-mapped rule. The actual process depends on the scope of the firewall rule, and has to be defined for each examined field individually. We demonstrate this here with two examples, which can easily be extended for other fields of interest.

a) IPv4 Address Field: The most common use in firewalls is the matching of IP address fields. We apply merging at the subnet level i.e., several single-mapped address rules are mapped into one multi-mapped rule covering one whole subnet. Ideally, if there are 254 single-mapped address rules covering e.g., the address range from 84.1.0.1 to 84.1.0.254 in R_i, we can replace these 254 rules with one multi-mapped rule 84.1.0.* in R_o.

Admittedly, this example covers an ideal situation: there are exactly 254 IPv4 addresses in the same subnet in R_i, so one can simply use the subnet address in R_o instead of 254 individual IPv4 addresses. However, merging with less than those 254 rules would change the accuracy of the resulting R_o. We introduce the variable χ to determine the threshold after which single-mapped rules are replaced by a multi-mapped one. Rules in R_i can be merged into a multi-mapped rule only if the number of mergeable rules is greater than χ. For example, if there are three address rules "84.1.0.1", "84.1.0.20" and "84.1.0.113" from the same subnet, and given that $\chi = 3$, they will be merged into "84.1.0.*". However, if $\chi = 4$, they will not be merged. The value of χ will be computed in the optimisation step.

b) SIP User Agent Header Field: Many SIP header-fields are free-from text fields e.g., the User-Agent string that denotes the generator of the SIP message. Its usage in SIP firewall would be highly beneficial e.g., to filter ill-configured user agents. Entries vary considerably due to the diversity of vendors and versions. We have seen through observation a common pattern how the string is generated, consisting of three parts, UA name, sub-name and version ususally in left to right order, commonly separated by whitespace. Based on this observation, single-mapped UA string rules can be merged into one multi-mapped one.

As an example, we can separate a UA string into two parts by the last non-alphabet, non-numeric character. We call the left part *name part* and the right part *version part*. Firstly, we subgroup the rules which share the same name part. For example, "Zultys ZIP 2 3.43" and "Zultys ZIP 2 3.47" can be subgrouped because they share the same name part "Zultys ZIP 2 3.", however "sipsak 0.8.11" and "sipura/SPA" can not

be subgrouped as their name parts are different. Again, χ indicates how many different UA strings can be merged into a new one. Only if more than χ rules can be found sharing the same name part, a new merged rule is created for it.

2) Significance Metric: We define a significance value F_{rq} as the number of times a rule is matched within the IDS. For instance, given a rule that allows incoming traffic from IP address "10.8.0.1" which is matched at the IDS by incoming traffic 10 times since the last optimisation, we set $F_{rq} = 10$. Using a Most Frequently Used (MFU) policy, we are able to order the rules in R_i by significance. This ensures that at least the offenders which the most overhead traffic will be denied access to the network.

3) Accuracy Changes: We introduce the term *space* to indicate how many entities are mapped to possible targets. For example, the space of a single-mapped IPv4 rule is always 1, while it is higher for multi-mapped rules (i.e., it is 254 for the above defined subnet rules). For UA strings we look at the size of the UA string. We can then define $S_{ruleset}$ as the space of all rules in a ruleset combined. Ideally, S_{R_i} should be close to S_{R_o} as this would indicate an accurate conversion.

Now we can calculate *false positives* and *false negatives* within the two rulesets. We define F_p as the increase of space through rule merging in R_o:

$$F_p = \sum i, \forall i \in S_{R_o}, i \notin S_{R_i} \tag{1}$$

Similarly, we can define F_n as the number of false negatives:

$$F_n = \sum i, \forall i \in S_{R_i}. i \notin S_{R_o} \tag{2}$$

Both merging and dropping will decrease accuracy in the target ruleset R_o, either by the introduction of false-positives (caused by merging) or false-negatives (caused by dropping). Hence, with the conversion from R_i to R_o, the target space will change, causing R_o to become less reliable.

For example, given R_i with 600 rules and setting σ_{max} to 20 and considering only single-mapped rules, only 20 rules with the highest F_{rq} can be considered for R, and information contained in the 580 lower significant remaining rules would be lost, so $S_{R_i} = 600$ and $S_{R_o} = 20$, and the false negative index $F_n = S_{R_i} - S_{R_o} = 580$.

The change in accuracy can be denoted by a (preliminary) *inaccuracy index*, as the sum of F_p and F_n. The higher this index gets, the more information is lost during the conversion from S_{R_i} to S_{R_o}.

However, F_p and F_n will generally not considered equally in all situations. For example, if in a DoS scenario it would be acceptable to let a certain degree of malicious traffic pass, but regular users should rather not be affected, F_p should be rather low, while higher values for F_n are acceptable for a firewall in blacklisting mode. Other scenarios might require a different setup. To reflect this, we introduce a weighting factor η and and define

$$\mu = F_p + \eta F_n \tag{3}$$

TABLE I
INTRODUCED VARIABLES

R_i	original ruleset consisting of a list of single-mapped rules
R_o	output ruleset after optimising, consiting of a list of single- and multi-mapped rules
σ_{max}	size constraint of R_o
F_p	false positive index introduced by optimisation
F_n	false negative index introduced by optimisation
χ	a threshold number to indicate the minimun number of single-mapped rules needed before they can be merged into one multi-mapped rule
η	a weighting factor to indicate the importance of F_p compared to F_n

where μ is the final inaccuracy index, which should be as low as possible.

All defined variables are listend in table I.

4) Formal Specification: The formal specification of the algorithm is listed as algorithm 1. Given the input values R_i, σ_{max} and χ the algorithm calculates the optimised ruleset R_o using a temporary ruleset R_{tmp}. R_i will be separated into subsets with each subset containing its own mergeable rules. Then, the size of each subset is calculated, and if the size is greater or equal than χ, all rules will be merged into a multi-mapped rule, which in turn is inserted into R_{tmp}. Otherwise, the rules in the subnet will not be merged, and will be inserted into R_{tmp} independently. F_{rq} of a multi-mapped rule is calculated as the sum of each single-mapped rule's F_{rq}. Finally, rules in the R_{tmp} are sorted according to their F_{rq} and the top σ_{max} rules with highest F_{rq} are selected to generate R_o. The methods "SubGroup" and "Merge" are depended on the type of rules (e.g., IPv4 address rule, UA header string).

```
   Input: R_i, σ_max, χ
   Output: R_o
1  R_o = φ;
2  R_tmp = φ;
3  SubGroup(R_i): R_i = R_i1 ∪ R_i2 ∪ R_i3 ∪ ... ∪ R_in;
4  foreach j(j ∈ {1, ..., n}) do
5      if |R_ij| ≥ χ then
6          r_multi = Merge(R_ij);
7          R_tmp = R_tmp ∪ r_multi;
8      end
9      else
10         R_tmp = R_tmp ∪ R_ij;
11     end
12 end
13 SortByFrqAcend(R_tmp);
14 while |R_o| ≤ σ_max do
15     R_o = R_o ∪ Pop(R_tmp);
16 end
```

Algorithm 1: Converting R_i to R_o

We call the procedure in code lines 3 to 12 "merging" and the procedure in code lines 13 to 15 "dropping".

Finally, we formalise the optimisation model as follows.

$$min : \mu = F_p + \eta F_n \tag{4}$$

$$s.t. : R_o = f(R_i, \sigma_{max}, \chi) \tag{5}$$

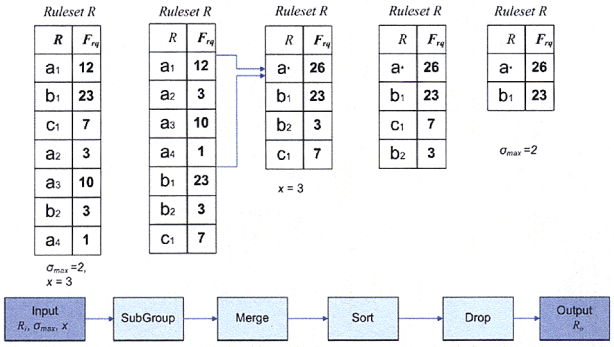

Fig. 4. Example process of the optimisation algorithm - here, the four similar rules $a_1 - a_4$ are merged into one new rule and rules with lesser frequency are dropped.

$$|R_o| \leq \sigma_{max} \tag{6}$$

Here, (4) is the minimisation objective function to achieve, indicating the goal to find an optimal result with the inaccuracy index μ minimised. (5) and (6) are constraint functions, with (5) showing how to reach the output ruleset R_o using algorithm 1 as f, and constraint (6) indicating that the size of output R_o must be less than σ_{max}.

5) Example Run: The general optimisation process using a concrete example can be seen in figure 4. Here, different rules ($a_x . b_x . c_x$) with their frequency are shown. First, similar rules are subgrouped, so that e.g., all a_x rules are put together. Then, similar rules $a_1 - a_4$ are merged into one new rule, as there are 4 individual a-rules here, passing the threshold value $\chi = 3$. However, the b-rules are not merged, as there are only two of them. The four new rules are sorted by their frequency, and finally the two rules with the lowest frequency are dropped to satisfy $\sigma_{max} = 2$.

V. Application

A. Calculation with Real-Life Traffic

For testing and verification, we have applied the ruleset optimiser to test-traffic from different real-life SIP VoIP providers. Traffic was recorded for several hours, resulting altogether in about four Gbyte of tracefile data. As a simple test, we generate for each traffic sample an input ruleset R_i that matches every single packet.

We generated 6 different samples from that traffic to be used as R_i: ruleset A - C containing each 450, 600, and 1034 IPv4 address rules, respectively. Furthermore, rulesets D - F with 51, 86, 107 SIP UA rules. We have developed a prototype optimiser written in Perl to input these 6 samples as R_i. The program optimises the ruleset and generates R_o, including the calculation of F_p and F_n. The optimiser runs on a 1 Ghz Linux machine with 512 MByte RAM.

For the tests, we arbitrarily set $\sigma_{max} = 20$ and $\eta = 20$. The result is shown in figure 5. The inaccuracy index μ is calculated for each values for χ ranging from 1 to 254. Here, we can clearly see the optimum value for χ as a local

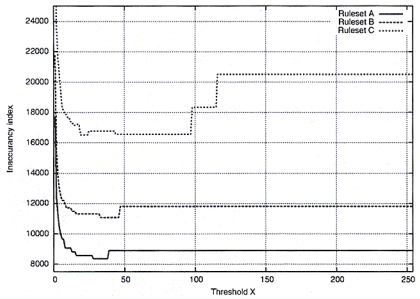

Fig. 5. The inaccuracy index of outcome ruleset varies according to threshold χ for ruleset A and B, when $\eta = 20, \sigma_{max} = 20$

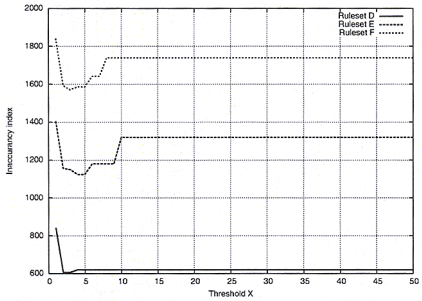

Fig. 6. The inaccuracy index of outcome ruleset varies according to threshold χ for ruleset C, when $\eta = 20. \sigma_{max} = 20$

minimum. For example, for ruleset A, the optimum value for χ is around 38. Setting the wrong value for χ, either by setting it to low or to high causes the accuracy to degrate. Moreover, the optimum value for χ changes for different rulesets.

To calculate the false positive F_p for the SIP UA rules, we arbitrarily assume that each multi-mapped rule contains 50 entries at most. For instance, if a group of 5 single-mapped UA rules are merged into one multi-mapped rule, the False positive is 50 - 5 = 45. In this way, we can calculate F_p for the given R_o. Figure 6 shows our test result with ruleset D, E and F. The result is similar to the previous one.

B. Optimisation

To find the optimum value for χ, we looped over each value of χ to find out the local minimum. This is a very time-consuming process and finding the right value for χ is an NP-hard problem [15]. We therefore apply a high efficient heuristic algorithm to calculate the best value for χ in reasonable time. Here, we employ Golden Section Search (GSS) [16] to solve this problem. GSS is a simple and efficient 1-dimensional optimisation method which does not require derivatives. It can deal with our unimodal objective by rapidly narrowing our search interval and still finding the optimum result.

We ran several calculations of the optimisation process both with the full loop search and with GSS optimisation applied. Seen in table II, we can see that GSS can improve calculation time by around 90% with our script prototype. While reducing calculation time, the accuracy of the resulting μ is not influenced by the application of GSS.

TABLE II
CALCULATION TIME COMPARISON OF THE ALGORITHMS, T2 WITH GSS APPLIED

Ruleset	T1 (s)	T2 (s)	Solution with GSS
A	51.51	3.31	$\chi_{optimal}$ = 38
B	95.69	6.29	$\chi_{optimal}$ = 46
C	229.04	17.71	$\chi_{optimal}$ = 97
D	0.23	0.04	$\chi_{optimal}$ = 3
E	0.49	0.13	$\chi_{optimal}$ = 5
F	0.76	0.18	$\chi_{optimal}$ = 3

VI. Related Works

Researchers have proposed different ways to improve the performance of a firewall, mainly two approaches exist: improving performance by generating an optimised order of firewall rules or reducing the size of the ruleset.

A proper order of rules is helpful to reduce the number of rule comparisons required per packet. Fulp [17], or Hamed et al. [18] propose such a mechanism based on different metrics, like traffic statistics. However, they do not reduce the size of the ruleset.

Like our work, researchers have proposed mechanism to reduce the overall ruleset size. Gupta [19] introduces forward or backward redundant rules that can be eliminated from the ruleset. A backward redundant rules is defined as an existing rule r appearing earlier than r', which is a subset of r. On the other hand, if there exists a rule r appearing after r', which is a subset of r, it is forward redundant. Liu [20] proposes further methods to eliminate backward and forward redundant rules. Similar to our work, Yoon et al. [21] propose an aggressive reduction algorithm to find a group of rules and replace it with a smaller new group. However, their work does not to allow the space of the generated ruleset to differ from the original one, hence it is not guaranteed that the generated ruleset will be significantly smaller.

All of these propositions work only at the network level and would need further enhancements to work in SIP ALGs.

VII. Conclusion and Future Work

Flooding attacks will be a severe threat for Internet services based on SIP. While new methods are currently designed to counter these threats, all components of a security solution have to be prepared to take the excessive overload traffic that is generated by this threat.

In this work we have proposed a new firewall operation algorithm to balance between optimum security and optimum performance, as both aspects together are almost impossible to achieve. So, instead of trying to prevent and deny every uncommon message, we accept a small fraction of these messages and thus ensure the ongoing operation of the service.

Our algorithm is only a first step in this direction. We have shown a way to reduce the load at the firewall and still retaining most of the firewall's effectiveness. The feasibility of this solution has shown with real-life traffic captures. However, our next step would be to actually apply this algorithm to a firewall and evaluate its behaviour with ongoing flooding traffic. Depending on the situation, there is also optimisation potential for the proposed parameters of the algorithm e.g., by taking different message fields into account or utilise the Time-to-Live property for a more accurate calculation of the most import rules in the firewall. Performance of the calculation can also be increased by iteratively calculating the new optimised ruleset, instead of a complete recalculation each time new rules are inserted.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Spark, M. Handley, and E. Schooler. *Session Initiation Protocol*, 2002. RFC 3261.

[2] IP Multimedia Subsystem (IMS); Stage 2. Technical Report TS 23.238 (Release 8), 3GPP, 2007.

[3] VoIP Security and Privacy Threat Taxonomy. http://www.voipsa.org.

[4] D. Geneiatakis, A. Dagiouklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, S. Ehlert, and D. Sisalem. Survey of Security Vulnerabilities in Session Initiation Protocol. *IEEE Communications Surveys and Tutorials*, 8(3):68–81, September 2006.

[5] J. Fiedler, T. Kupka, S. Ehlert, T. Magedanz, and D. Sisalem. VoIP Defender: Highly Scalable SIP-based Security Architecture. In *Principles, Systems and Applications of IP Telecommunications (IPTComm 2007)*, New York, USA, July 2007.

[6] M. Nassar, S. Niccolini, R. State, and T. Ewald. Holistic VoIP Intrusion Detection and Prevention System. In *Principles, Systems and Applications of IP Telecommunications (IPTComm 2007)*, New York, USA, July 2007.

[7] Borderware SIPassure VoIP / SIP Security Solution. http://www.borderware.com/products/sipassure/.

[8] W. Stallings. *Network Security Essentials: Applications and Standards, 3rd edition*. Pearson Education, 2007.

[9] I. Green, T. Raz, and M. Zviran. Analysis of Active Intrusion Prevention Data for Predicting Hostile Activity in Computer Networks. *Communications of the ACM*, 2007.

[10] H. Sengar, D. Wijesekera, H. Wang, and S. Jajodia. Fast Detection of Denial of Service Attacks on IP Telephone. In *Proceedings of IEEE IWQoS2006, New Haven, CT*, 2006.

[11] S. Ehlert, C. Wang, T. Magedanz, and D. Sisalem. Specification-based Denial-of-Service Detection for SIP Voice-over-IP Networks. In *Third International Conference on Internet Monitoring and Protection (ICIMP2008)*, Bucharest, Romania, July 2008.

[12] Y. Qiu, J. Zhou, and F. Bao. Design and Optimize Firewalls for Mobile Networks. In *Proceedings of IEEE 60th Vehicular Technology Conference*, 2004.

[13] M. Luo, T. Peng, and C. Lekie. CPU-based DoS Attacks Against SIP Servers. In *IEEE/IFIP Network Operations and Management Symposium (NOMS 2008)*, Salvador, Bahia, Brazil, April 2008.

[14] B. Potter. Open source firewall alternatives. *Network Security*, 2006(1), January 2006.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman Co., New York, NY, USA, 1979.

[16] R. L. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998.

[17] E. W. Fulp. Optimization of Network Firewall Policies Using Ordered Sets and Directed Acyclical Graphs. In *Proceedings of IEEE Internet Management Conference*, 2005.

[18] H. Hamed and E. Al-Shaer. Dynamic Rule-ordering Optimization for High-speed Firewall Filtering. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, 2006.

[19] P. Gupta and N. McKeown. Packet Classification on Multiple Fields. In *Conference on Application, Technologies, Architectures and Protocols for Computer Communication*, 1999.

[20] A. X. Liu and M. G. Gouda. Removing Redundancy from Packet Classifiers. In *Proceedings of ACM SIGCOMM*, 2004.

[21] M. Yoon, S. Chen, and Z. Zhang. Reducing the size of rule set in a firewall. In *Proceedings of IEEE International Conference on Communications, ICC' 07*, 2007.