A Novel Mechanism for Anomaly Removal of Firewall Filtering Rules

Chi-Shih Chao[1], Stephen J. H. Yang[2]

[1] Communications Eng. Department, Feng Chia University, Taiwan
[2] Computer Sciences and Information Eng. Department, National Central University, Taiwan
cschao@fcu.edu.tw, Stephen.Yang.Ac@gmail.com

Abstract

Firewalls are always treated as the core devices for network security to protect networks from being attacked. Still, properly configuring firewall rules is no easy task due to its laboring and time-consuming characteristic. In some cases, firewall rules need to be added, deleted, modified, or order-changed from time to time to fit in the dynamic of network traffic. As a result, firewalls are subject to rule anomalies caused by misconfigurations such that network security holes can be created accordingly, and then damage the managed networks and even worse the firewalls themselves. In this paper, an enhanced firewall rule management approach is proposed where it can not only pinpoint the anomalies among firewall rules effectively and efficiently, but also provide a novel mechanism for correct and speedy removal of rule anomalies. As a demonstration, a visualized firewall rule anomaly removal system has been realized and performance evaluations on anomaly removal have been also conducted, in which our developed mechanism shows its excellence and feasibility.

Keywords: Firewall rule anomaly diagnosis, Rule anomaly removal, DDoS on firewalls, System feasibility

1 Introduction

In Internet, firewalls play an extremely important role to defense the protected networks from attacks. Configuring a firewall properly is not at all a simple task since there are usually hundreds or even thousands of rules equipped in an enterprise-level firewall [1]. For this, it can often be found that there are conflicts among firewall rules, say rule anomalies [2], because firewall rule configurations would need to be changed or adapted to the dynamic of network traffic [3]. There are times that system administrators with little knowledge of network security would also make these anomalies on rules, and these rule anomalies can lead to security holes or vulnerabilities for network attackers [4]. This explains why researchers [5] like to compare the firewall configuring task to programming a distributed system in assembly language. In some cases, firewalls, themselves, could become targets of network attacks, e.g., DDoS (Distributed Denial of Service), and those rule anomalies can be used by attacks which send tons of small unmatched attacking packets to choke firewalls and the protected networks [4, 6].

Among the literature in this topic, E. Al-Shaer and H. Hamed first define a rule anomaly as a duplicate or multiple rule-matching for a packet in a rule set. Based on the concept, they formally define several different intra-/inter-ACL (Access Control List) anomalies among the firewall rules [2, 7]. Nevertheless, since a Finite-State-Machine (or FSM)-based comparison between each pair of filtering rules for anomaly diagnosis should be conducted, their rule anomaly inspection algorithm will meet an inefficiency as the number of rules or firewalls grows [8]. To lower the comparisons between firewall rules needed in [2], Y. Yin et al. [9] segment the IP address space, which is formed by the managed source and destination networks, into blocks where each block is precisely split by the IP addresses in the conditional field of each firewall rule. With these varying-sized blocks, a SIERRA tree is built and two conflict rules would be hanged on the same branch of the tree [10]. Network managers only needs to do the anomaly inspections or check-ups on rules in the same spatial block(s) (or on the same branches in the SIERRA tree), as opposing to wasting enormous time to conduct comprehensive pair-wise rule comparisons. Yet, this approach would lead to a fatal drawback in a networking environment with frequent rule updates. Besides, a clean-slate reconstruction of the SIERRA tree is very possibly unavoidable if a simple rule deletion or insertion is administored [8]. It is because space blocks are precisely sliced according to the IP addresses of each rule. So, once one rule changes, the whole spatial rule relationship would change, and the corresponding data structures could be reconstructed. This drawback also reveals that the local diagnosis results, i.e., the intra-ACL diagnosis results,

*Corresponding Author: Chi-Shih Chao; E-mail: cschao@fcu.edu.tw
DOI: 10.3966/160792642020072104005

can hardly be re-used for the diagnosis of inter-ACL rule anomalies. By the same token, modification or reconfiguration of firewall rules for new demands of network security could fail their system to go live in time for varying threats.

We have gotten involved in this topic for a decade and the research results were hailed as one of the most useable systems in the world by far [11]. At very beginning, a RAR tree (Rule Anomaly Relation tree) data structure was designed to rule out firewall rules having no relationship (or intersection) between each other in packet filtering so as to reduce the time-consuming pairwise rule comparisons needed by [2]. With the innate characteristic of this tree structure, local diagnosis results can be easily integrated for cross/inter-firewall rule anomaly diagnosis [12]. After two-year striving, an upgrade version was proposed where a new data structure was implemented, called ARAR tree (Adaptive Rule Anomaly Relation tree), which can not only slice the packet filtering space more properly and then fasten the diagnosis, but also keep the advantage of tree structure for system extensibility. After that, a further improved data structure was provided in our system development, named Enhanced ARAR tree, in which it incorporates more accurate coordinates to do slicing work on packet filtering space such that a better performance on diagnosis can be obtained [8]. Meanwhile, a user-friendly visualized interface had been up as part of our developed system prototype. Users can promptly do correct reactions to deal with firewall misconfigurations via the visualized diagnosis interface [8, 12].

Looking at all of the developments in this topic, most of research mainly put their focus on firewall rule anomaly diagnosis, rather than effective anomaly removal or recovery [11]. In this paper, considering the high possibility of network attacks caused by rule anomalies, an anomaly removal mechanism is proposed, which can eliminate redundancy anomalies and shadowing anomalies effectively and efficiently, and also identify the other types of anomalies among firewall rules correctly. For this purpose, based on our Enhanced ARAR tree structure and a new two-dimensional traffic-filtering matrix, an anomaly removal mechanism which contains a two-phased rule refinement procedure is designed and implemented. A visualized interface is also developed and integrated in our previously developed prototype system to effectively help users doing rule removal. The rest of this paper is organized as follows: We organize Session 2 to brief the basis of rule anomalies and how an Enhanced ARAR tree is created based on the collected firewall rules. In Session 3, we spell out how the tree structure combined with an associated matrix can be used in our developed mechanism to facilitate the correct and fast removal of firewall rule anomalies. As a demonstration, Section 4 presents the system implementation and performance evaluations for our developed mechanism, and Section 5 concludes this paper and shows some directions of our system development in the upcoming future.

2 Rule Anomalies and Enhanced ARAR Tree

Serving as an essential part in the context of this article, this section provides a short brief on rule anomalies and our Enhanced ARAR tree for those who are unfamiliar with what we have done before. For further information, please kindly refer to [2] and [8].

2.1 Firewall Rule Anomalies

For the anomalies between firewall rules, they are completely defined and classified by E. Al-Shaer et al [2]. Based on the <Action> field of ACL rules, filtering area, and rule order in ACL, there are five anomalies between each pair of firewall rules. As shown in Figure 1, shadow anomaly means the filtering area of rule R2 is totally covered by that of rule R1, but R1 and R2 have different values of <Action>. In this case, R2 is "shadowed" since it will not be triggered forever due to R1. In contrast, redundancy anomaly stands for the filtering area of R2 is totally covered by that of R1, and R1 and R2 have the same values of <Action>, e.g., both are "accept" or "deny." In this situation, R2 fully reveals its redundancy and unnecessity on filtering function. In addition, there is another situation which can be called rule redundancy, where the filtering area of R2 contains that of R1 and they two have the same value of <Action>. In most of cases, R1 can be removed since its filtering function can be completely replaced by R2. Generalization anomaly means the filtering area of R2 contains that of R1 and they two have different values of <Action>. Correlation anomaly is that R1 and R2 have intersection between each other and they two have different values of <Action>. At last is non anomaly, as shown in Figure 1, in which two rules have no interaction or have intersections but with the same action values.

2.2 Enhanced ARAR Tree

To fasten the anomaly diagnosis of firewall rules, a rule which does not affect the others can be ruled out from the time-consuming pairwise rule comparisons [8]. To do so, a two-dimensional traffic-filtering diagram is built, on which the filtering area of each firewall rule can be depicted as a rectangle at its proper location. Later, referring to the coding-tree data structures widely used in image/video compression, the traffic diagram will be split recursively and reverse-exponentially until a split block finds (a) there is no rule filtering space within it, (b) there is only one rule filtering space within it, or (c) there are more than two rule filtering spaces within it and the split block is

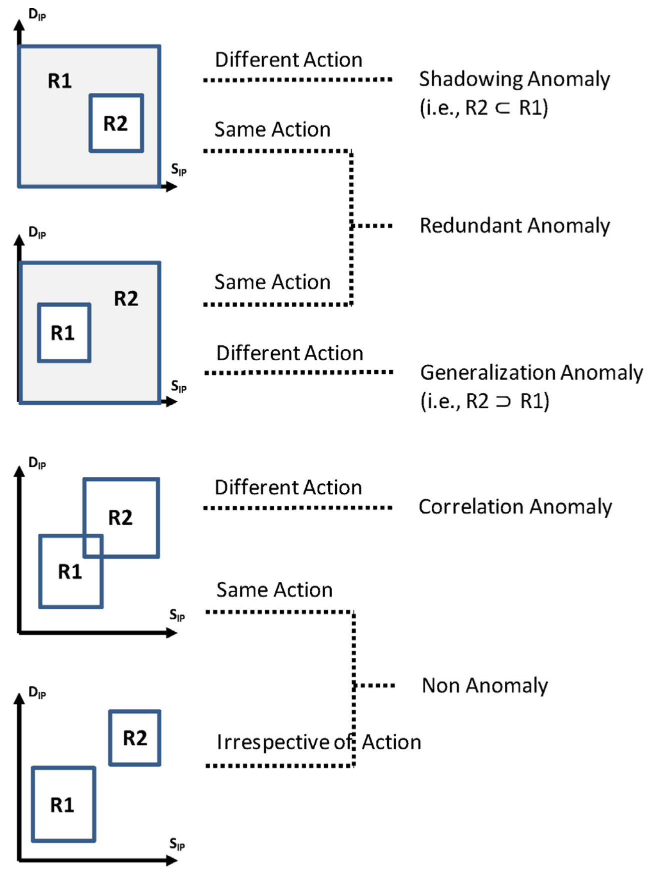


Figure 1. Types of firewall rule anomalies

Rule	S_{IP}	D_{IP}	Action	Total area
R0	1.192.0.0/10	2.0.0.0/9	Deny	35184372088832
R1	1.192.0.0/9	2.0.0.0/10	Accept	35184372088832
R2	1.0.0.0/9	2.64.0.0/10	Accept	35184372088832
R3	1.0.0.0/10	2.0.0.0/8	Accept	70368744177664
R4	1.128.0.0/10	2.0.0.0/8	Deny	70368744177664
R5	1.64.0.0/10	2.128.0.0/10	Deny	17592186044416
R6	1.64.0.0/10	2.128.0.0/9	Deny	35184372088832
R7	1.0.0.0/8	2.0.0.0/8	Accept	281474976710656
R8	1.128.0.0/9	2.0.0.0/8	Deny	140737488355328
R9	1.255.0.0/10	2.0.0.0/10	Accept	17592186044416

Figure 2. Example firewall rule set

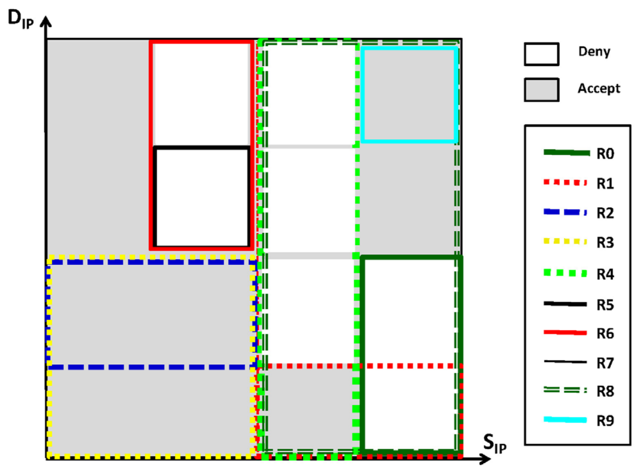


Figure 3. Two-dimensional traffic-filtering diagram

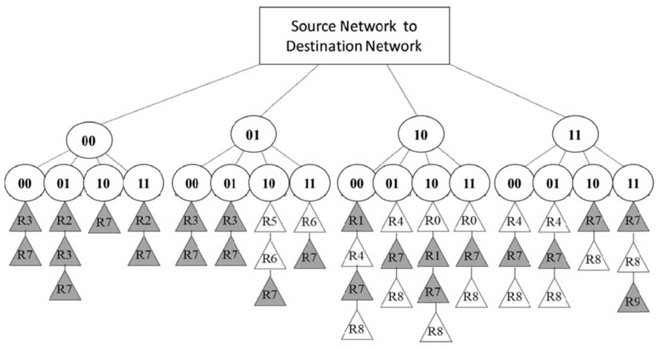


Figure 4. The corresponding enhanced ARAR tree of Figure 2

exactly the same as those rule filtering spaces. After that, the address space of a firewall rule can be recorded in our Enhanced ARAR tree in the form of □—○—...—○—△—...—△, where □ contains the IP address ranges of the source network domain and destination network domain of a designated routing path, ○ is used to indicate the split block(s) spanned by the address space of the rule, △ shows the label (or the order) of the rule. By dealing with each rule in this fashion, the Enhanced ARAR tree depicting the structural configuration can be created. Nodes which are hung under a tree branch stands for rules of the nodes can have filtering effect on the flows within the traffic area indicated by the branch. We call these rules are "related" and send them to further checks for types of anomalies [8]. Figure 2 is an example rule set which will be used throughout the paper and the filtering area of each rule in the traffic-filtering diagram is depicted in Figure 3. Figure 4 shows the corresponding Enhanced ARAR tree of the rule set in Figure 2 and Figure 5 contains detailed information about leaf nodes of the tree.

3 Rule Anomaly Removal

A firewall itself can become the target of DDoS attacks, while rule anomalies can raise the risk of exposing the firewall under such attacks [4, 6]. For example, redundancy anomalies in a firewall can dramatically slow down its performance of packet filtering because a great number of mismatches by malicious packets could be created by the anomalies. Section 3.1 and Section 3.2 comprise our two-phased procedure which is developed to effectively and efficiently remove these anomalies.

Node	S_{IP}	D_{IP}	Rule	area
0000	1.0.0.0/10	2.0.0.0/10	R3、R7	17592186044416
0001	1.0.0.0/10	2.64.0.0/10	R2、R3、R7	17592186044416
0010	1.64.0.0/10	2.0.0.0/10	R7	17592186044416
0011	1.64.0.0/10	2.64.0.0/10	R2、R7	17592186044416
0100	1.0.0.0/10	2.128.0.0/10	R3、R7	17592186044416
0101	1.0.0.0/10	2.192.0.0/10	R3、R7	17592186044416
0110	1.64.0.0/10	2.128.0.0/10	R5、R6、R7	17592186044416
0111	1.64.0.0/10	2.192.0.0/10	R6、R7	17592186044416
1000	1.128.0.0/10	2.0.0.0/10	R1、R4、R7、R8	17592186044416
1001	1.128.0.0/10	2.64.0.0/10	R4、R7、R8	17592186044416
1010	1.192.0.0/10	2.0.0.0/10	R0、R1、R7、R8	17592186044416
1011	1.192.0.0/10	2.64.0.0/10	R0、R7、R8	17592186044416
1100	1.128.0.0/10	2.128.0.0/10	R4、R7、R8	17592186044416
1101	1.128.0.0/10	2.192.0.0/10	R4、R7、R8	17592186044416
1110	1.192.0.0/10	2.128.0.0/10	R7、R8	17592186044416
1111	1.192.0.0/10	2.192.0.0/10	R7、R8、R9	17592186044416

Figure 5. Detailed information of leaf nodes in Figure 4

3.1 Anomaly Removal Phase I

In this phase, focus is put on the removal of shadow anomalies and redundancy anomalies, to ward off the firewall DDoS attacks. It is because these two rule anomalies can easily incur attacks using packet mismatches to choke firewalls, and can be removed without hesitation, whereas the other types of rule anomalies need the interventions from network experts or administrators to tackle them and providing identificatons can be the only as well as proper choice that diagnosis systems can make [7]. Be aware that, in this phase, we remove these two types of anomalies caused by "being shadowed." For the redundant anomalies which are caused by "not being shadowed," i.e., a rule with smaller filtering area which is fully cotained by another rule with same filtering action and lower order, they will be handled in Section 3.2. So, in this phase, if the filtering area of a rule is completely shadowed by some other rules, the rule should be removed from the rule set no matter what action the rule takes or what anomaly it has. On the basis of this fact, we can achieve the phase I anomaly removal with little extra effort while establishing our Enhanced ARAR tree.

```
Algorithm Rule_Anomaly_Removal_Phase_I (R)
Input:   A firewall rule set R
Output:  A firewall rule set R' without rules being
         shadowed.
Step 1:  While adding a new leaf node of a rule in the
         Enhanced ARAR tree, just check if it is the
         first leaf (or rule) of a branch.
Step 2:  If it is, nothing should be done; otherwise cut
         the shadowed filtering area indicated by that
         branch from the rule.
Step 3:  Once the size of the effective filtering area of
         the rule is found to become zero after the
         construction of the tree, the rule will be
         deleted from the original rule set due to its
         redundancy.
```

To clearly describe this process, the example rule set shown in Figure 2 is used. According to its corresponding Enhanced ARAR tree in Figure 4, there are rules R3 and R7 beneath the branch labeled (00, 00). Since the filtering portion of R7 indicated by that branch is covered (or shadowed) by that of R3, the effective filtering area of R7 can be subtracted by the area of that branch. In this fashion, the process is applied to each rule (or leaf) which is under the first rule (or leaf) of each branch of the Enhanced ARAR tree. After the construction of the whole tree, rules whose size of the entire effective filtering area becomes zero can be deleted from the original rule set. In our example, it can be found the size of the effective filtering area for R8 and R9 becomes zero (shown in Figure 6), which means they two can be matched by no packet in the rule set, and will be eliminated forever.

Rule	S_{IP}	D_{IP}	Action	Effective Area
R0	1.192.0.0/10	2.0.0.0/9	Deny	35184372088832
R1	1.192.0.0/9	2.0.0.0/10	Accept	17592186044416
R2	1.0.0.0/9	2.64.0.0/10	Accept	35184372088832
R3	1.0.0.0/10	2.0.0.0/8	Accept	52776558133248
R4	1.128.0.0/10	2.0.0.0/8	Deny	52776558133248
R5	1.64.0.0/10	2.128.0.0/10	Deny	17592186044416
R6	1.64.0.0/10	2.128.0.0/9	Deny	17592186044416
R7	1.0.0.0/8	2.0.0.0/8	Accept	52776558133248
R8	1.128.0.0/9	2.0.0.0/8	Deny	0
R9	1.255.0.0/10	2.0.0.0/10	Accept	0

Figure 6. Rule effective filtering area after phase I

3.2 Anomaly Removal Phase II

In this phase, a rule which can be completely replaced by some other rules having lower filtering orders will be removed. We have to be very careful that the filtering diagram of a rule set should not be changed from what it looks like originally, after any redundancy removal [7]. For this, a rule, on top of some other rules with the same filtering action and also being contained (or included) by them and without rules having different action in-between them, can be viewed as a redundant one also. In our work, an algorithm is developed to speed up the job in this phase: While establishing the Enhanced ARAR tree, we can simultaneously build a two-dimensional matrix which records what filtering blocks a rule spans and what filtering action the rule takes in those blocks. For example, according to the distribution and numbering of filtering blocks of the rule set shown in Figure 3 and Figure 7, respectively, a two-dimensional matrix can be built (shown in Figure 8) where the value of 0 represents "deny," 1 stands for "accept," and empty means no filtering effect on the designated block for a rule.

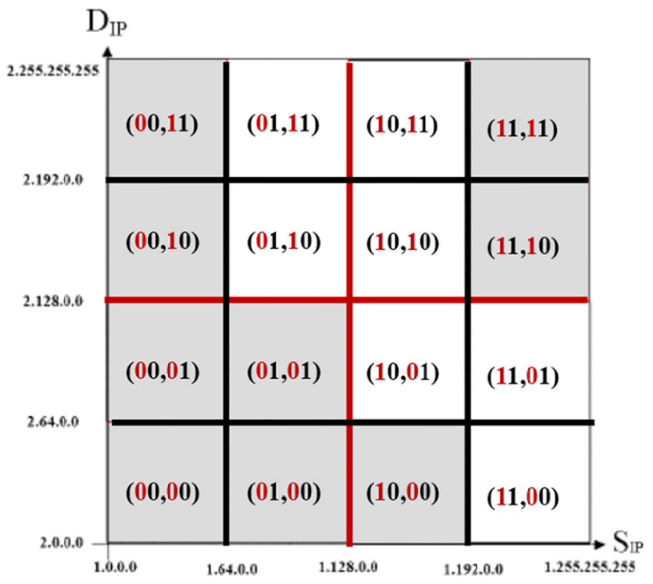
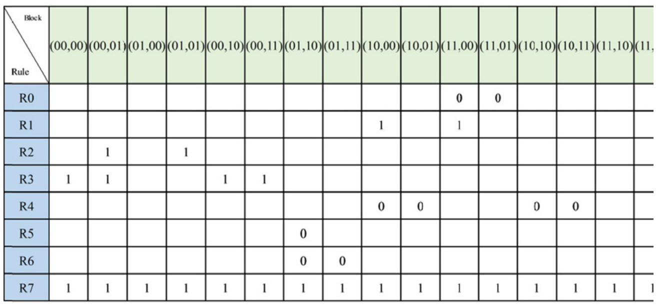


Figure 7. Block numbering of traffic filtering diagram



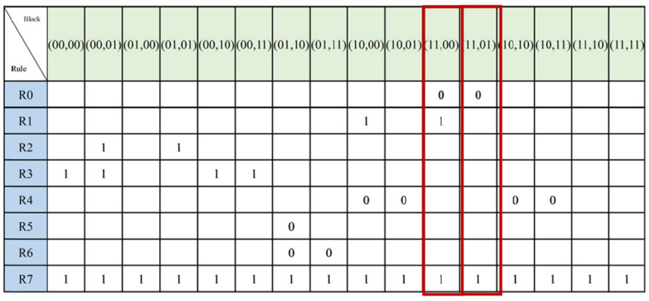
Block / Rule	(00,00)	(00,01)	(01,00)	(01,01)	(00,10)	(00,11)	(01,10)	(01,11)	(10,00)	(10,01)	(11,00)	(11,01)	(10,10)	(10,11)	(11,10)	(11,
R0											0	0				
R1									1		1					
R2		1		1												
R3	1	1			1	1										
R4									0	0			0	0		
R5							0									
R6							0	0								
R7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 8. Two-dimensional traffic filtering matrix

To correctly remove the rules which can be replaced by some other rules with lower orders, the following algorithm is used:

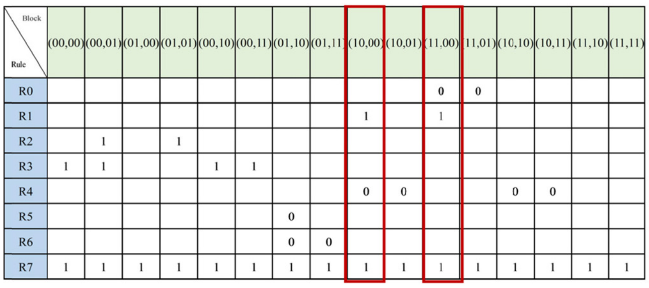
```
Algorithm Rule_Anomaly_Removal_Phase_II (M, R')
Input:   A firewall rule set R' passing the process of
         Phase I and its two-dimensional filtering
         matrix M
Output:  A refined firewall rule set R* without any
         redundant anomaly
Step 1:  From the top of the matrix M, check the rule
         with its non-empty blocks and look
         downwards for each of these blocks if there
         exists another non-empty block with the same
         filtering action and no block(s) with different
         action in-between them.
Step 2:  If the answer is yes, the rule can be removed
         safely from the matrix M and go to Step 3;
         Otherwise go to Step 3 directly.
Step 3:  Pick the next rule up from M and do the same
         check from Step 1 until all of the rules in the
         matrix have been checked.
```

As an illustration, in Figure 9, checks are first conducted for blocks (11,00) and (11,01) of R0. After the comparison between the values of block (11,00) in R0 and R1, it can be found that R0 cannot be removed since they two have different filtering actions in this block. In the second round of the checks, with the same fate as rule R0 has, R1 can also be found non-removable from the matrix since one of its filtering blocks, blocks (10,00), has a corresponding block right beneath it and with opposite filtering action; i.e., R1's (10,00) is "deny" and R4's (10,00) is "accept." So, after the check of R1, the rule set remains unchanged, as shown in Figure 10.



Block / Rule	(00,00)	(00,01)	(01,00)	(01,01)	(00,10)	(00,11)	(01,10)	(01,11)	(10,00)	(10,01)	(11,00)	(11,01)	(10,10)	(10,11)	(11,10)	(11,11)
R0											0	0				
R1									1		1					
R2		1		1												
R3	1	1			1	1										
R4									0	0			0	0		
R5							0									
R6							0	0								
R7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

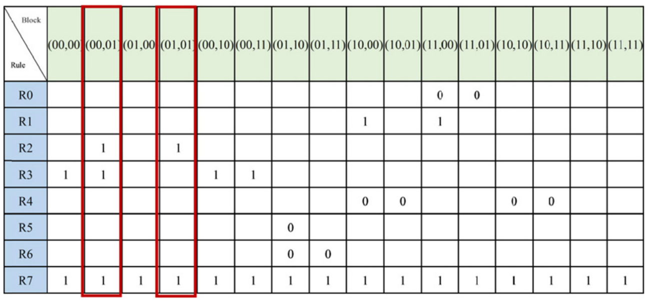
Figure 9. Redundancy check for rule R0



Block / Rule	(00,00)	(00,01)	(01,00)	(01,01)	(00,10)	(00,11)	(01,10)	(01,11)	(10,00)	(10,01)	(11,00)	(11,01)	(10,10)	(10,11)	(11,10)	(11,11)
R0											0	0				
R1									1		1					
R2		1		1												
R3	1	1			1	1										
R4									0	0			0	0		
R5							0									
R6							0	0								
R7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

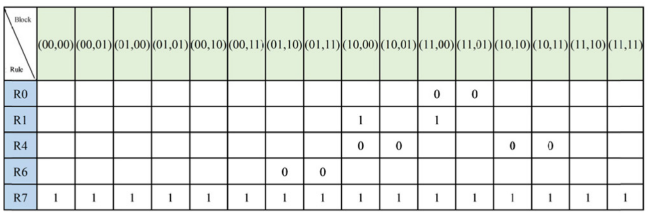
Figure 10. Redundancy check for rule R1

Now it is the turn for the check of rule R2. According to the Phase II algorithm, all of the filtering blocks of R2, blocks (00,01) and (01,01) in Figure 11, have their corresponding blocks right beneath them with the same filtering action, i.e., R3's blocks (00,01) and (01,01). It represents that R2 can totally be contained by R3 and they two have the same action of "accept." R2 can be deleted from the rule set forever. In this fashion, after checking the last rule in the rule set, a refined rule set (Figure 12) with the same network traffic filtering effect as the original rule set has can be obtained (please comparing Figure 3 with Figure 13), after our two-phased rule anomaly removal.



Block / Rule	(00,00	(00,01	(01,00	(01,01	(00,10)	(00,11)	(01,10)	(01,11)	(10,00)	(10,01)	(11,00)	(11,01)	(10,10)	(10,11)	(11,10)	(11,11)
R0											0	0				
R1									1		1					
R2		1		1												
R3	1	1			1	1										
R4									0	0			0	0		
R5							0									
R6							0	0								
R7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 11. Redundancy check for rule R2



Block / Rule	(00,00)	(00,01)	(01,00)	(01,01)	(00,10)	(00,11)	(01,10)	(01,11)	(10,00)	(10,01)	(11,00)	(11,01)	(10,10)	(10,11)	(11,10)	(11,11)
R0											0	0				
R1									1		1					
R4									0	0			0	0		
R6							0	0								
R7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 12. Rule set after two-phased redundancy removal

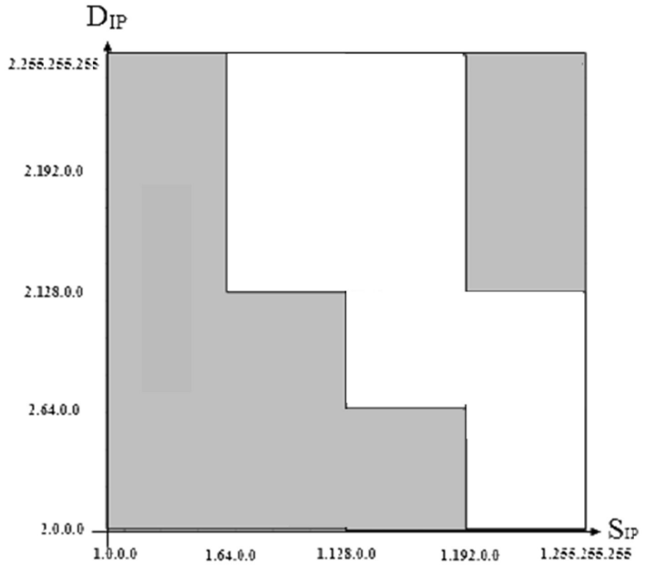


Figure 13. Traffic filtering diagram after two-phased redundancy removal

3.3 Anomaly Identification

In our work, the other three types of anomalies, i.e., generalization anomaly, correlation anomaly, and non anomaly can be identified also from a rule set by our mechanism with easy:

(1) For correlation anomaly and generalization anomaly, we can apply the same idea and process mentioned in Section 3.2 to the matrix refined from our two-phased anomaly removal mechanism to check if each pair of rules has intersection with each other. If there is, check further if the intersection is equal to any one of them. If the answer is yes, the diagnosis comes to generalization anomaly of the pair of rules; otherwise correlation anomaly. As an example in Figure 12 (the refined rule set), R4 has intersection with R7, which is totally equivalent to R4. R4 can be called being "generalized" by R7. Another example of rule intersection in Figure 12 happens between R1 and R4, in which the intersection is just part of each one of them two, i.e., block (10.00). We can call these two rules have a correlation anomaly, accordingly.

(2) For non anomaly, it can be easily figured out by slashing the rules with the four other anomalies, and the left can be concluded as rules with "non anomaly."

Be noticed that the three rule anomalies mentioned in this subsection cannot be removed or modified without the interaction or intervention with the administrators or experts of the managed network [7]. Thus, the only thing we can do for these three anomalies is to provide proper diagnosis identifications/ indications about them.

The analysis of the time-complexity of our mechanism can be broken down as follows:

(1) For the first phase removal of shadowing anomalies and redundancy anomalies in Section 3.1, it depends on the time of the establishment of an Enhanced ARAR tree and the number of anomalies. Thus, its time complexity can be represented as O(m*n), where m is the average number of occurred anomalies on a rule set with n rules.

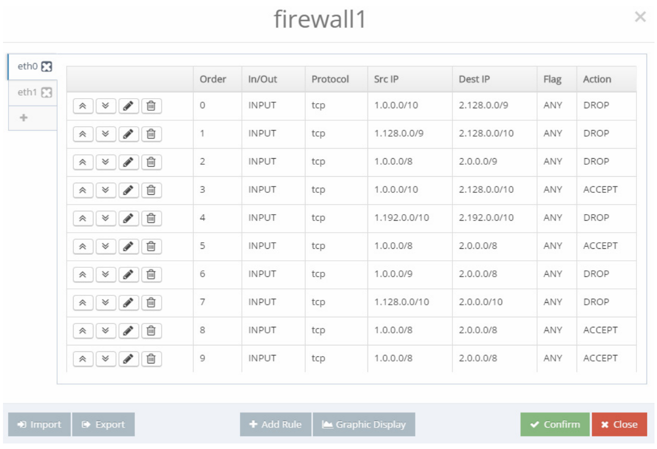
(2) The second phase anomaly removal sweeps out those redundant rules which cannot be dealt with in Phase I, by using the two-dimensional filtering matrix. Since each rule of the rule set would be checked at most once, the execution time of this phase can be bounded by O(m*n) if the two dimensional circular linked list for spare matrix [13] is used to do the matrix implementation. The reason that our Enhanced ARAR tree is not used in this phase is there would be a entire-tree traversal needed for each one of rules and a node in the tree can be visited twice for checking anoamlies, while one-time visit for each node in the two-dimensional filtering matrix is required.

(3) From (1) and (2), we can summarize the rule anomaly removal of our mechanism has time complexity of O(m*n).

(4) The last function of our mechanism, mentioned in this subsection, is to identify three other types of rule anomalies in the refined rule set. Since the calculation of intersections between each pair of rules in the refined set is required, the time complexity of this function goes to O(b^2*r^2), where b is the average number of blocks in a rule and r is the number of rules in the refined rule set.

4 System Implementation and Performance Evaluation

This paper is focused and centered on the introduction and elaboration of our newly developed rule anomaly removal mechanism, which has become part of our previously developed firewall rule anomaly diagnosis platform/system [8]. It is the first time the mechanism is shown in public, and also first time this function has been realized and implemented in this research field. For more information regarding the rule anomaly diagnosis, visualize the diagnosis results, do the diagnosis performance evaluation, make comparisons between [2], etc., please refer to [8]. Figure 14 is our firewall rule-editing interface (dedicated for service port 80 in the case) and Figure 15 shows its corresponding visualized two-dimensional traffic filtering diagram. The visualized interface shown in Figure 15 can do some interactions with users where we can highlight or hide a rule by clicking it, identify a suspicious filtering area by moving the mouse on top of it, and do anomaly removal by just pushing the "Optimize Confirm" button. Figure 16 shows the status before and after the execution of our anomaly removal mechanism, and we can find the number of rules in the rule set is refined from thirty, at the very beginning, down to five only.



	Order	In/Out	Protocol	Src IP	Dest IP	Flag	Action
	0	INPUT	tcp	1.0.0.0/10	2.128.0.0/9	ANY	DROP
	1	INPUT	tcp	1.128.0.0/9	2.128.0.0/10	ANY	DROP
	2	INPUT	tcp	1.0.0.0/8	2.0.0.0/9	ANY	DROP
	3	INPUT	tcp	1.0.0.0/10	2.128.0.0/10	ANY	ACCEPT
	4	INPUT	tcp	1.192.0.0/10	2.192.0.0/10	ANY	DROP
	5	INPUT	tcp	1.0.0.0/8	2.0.0.0/8	ANY	ACCEPT
	6	INPUT	tcp	1.0.0.0/9	2.0.0.0/8	ANY	DROP
	7	INPUT	tcp	1.128.0.0/10	2.0.0.0/10	ANY	DROP
	8	INPUT	tcp	1.0.0.0/8	2.0.0.0/8	ANY	ACCEPT
	9	INPUT	tcp	1.0.0.0/8	2.0.0.0/8	ANY	ACCEPT

Figure 14. Rule-editing interface

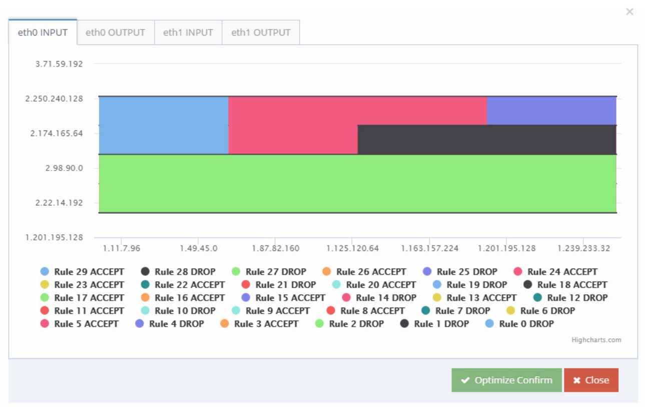


Figure 15. Visualized two-dimensional traffic filtering diagram

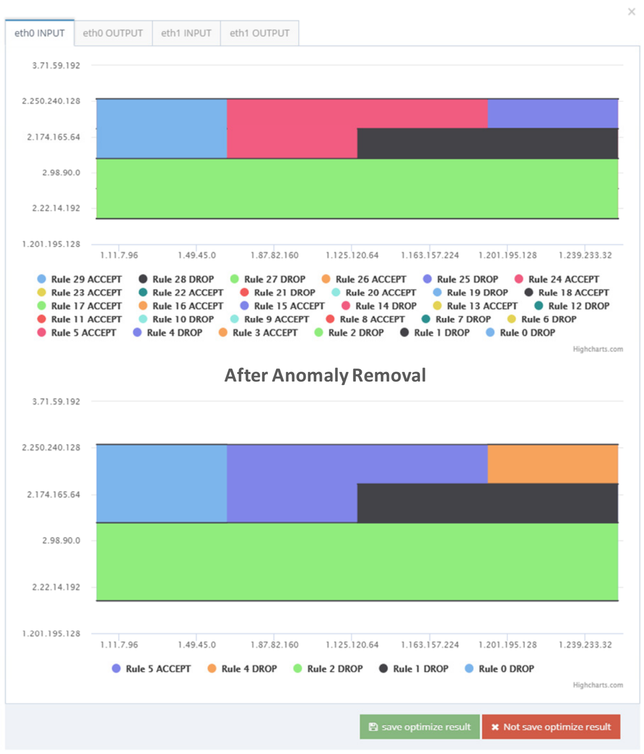


Figure 16. Before and after anomaly removal

Performance evaluation of our mechanism is also conducted and all of the experiments are run on our PC-based test platform which is equipped with an Intel core i5-4570 CPU @ 3.20GHz, 8GB RAM, and Windows 10 operating system. Experiments are grouped by the number of the rules in a rule set, and are designed systematically according the rule-generating policies defined in [14]. A hundred of experiments are performed for each of the experimental groups which contains 100 to 2000 rules in a rule set. Figure 17 and Figure 18 show the time in milliseconds and space in kilobytes, respectively, while doing our anomaly removal. Both reveal the excellence of our enhancement in the removal of anomalies in a firewall rule set.

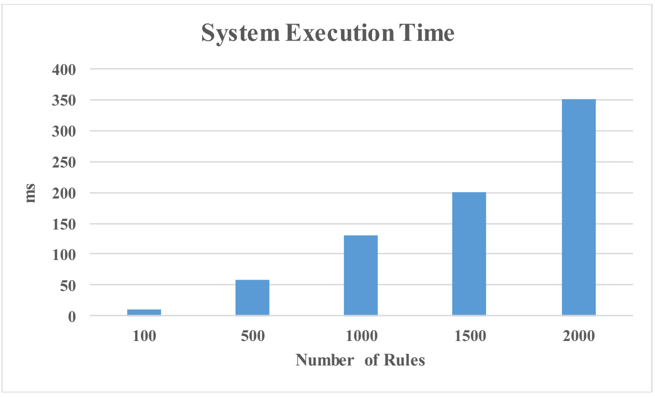


Figure 17. Time needed for anomaly removal

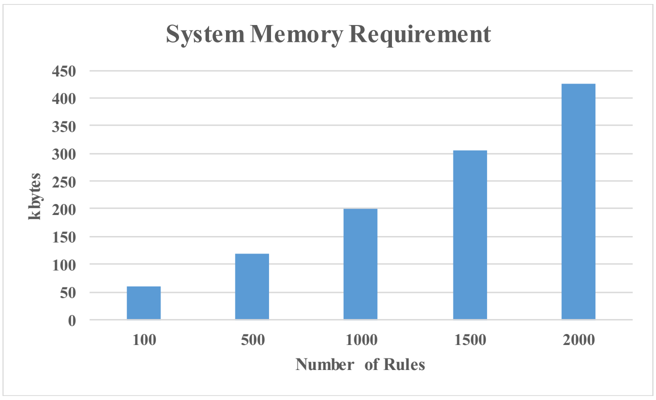


Figure 18. Space needed for anomaly removal

In addition to the mechanism provided in our work, several other approaches with different mathematical optimization strategies are also proposed and implemented for removal of firewall rule anomalies [15]. However, none of them can be competent at the job, because:

(1) For those algorithms solving problems on a phase-by-phased basis, e.g., greed method or dynamic programming, their solutions are hard to fulfill the requirements of anomaly removal. Depending on the accumulation of local optimizations or profits, these algorithms fail to achieve the goal of our anomaly removal where it can be found these algorithms may not always beget the same filtering effects as the input (original) rule set does.

(2) For the algorithms leveraging some approximating strategies, they are revealed not be suitable for this job since a difficult and error-prone process for the problem transformation and adaptation is needed. For example, genetic algorithm is tried in our work and found how to design proper adaptation function and mutation process is a long-term battle with huge uncertainty.

5 Conclusion and Future Work

Anomaly removal is an essential but not easy task for the management of firewall rules. Removing rules with anomalies from a firewall cannot only fasten the packet matching of firewall rules, but also reduce the impacts of the firewall while being attacked. Different from those research which have been up so far in this topic, in this paper, a novel anomaly removal approach with new data structure and algorithms is proposed, where it can not only provide a speedy and correct removal for redundant and shadowing anomalies, but also identify the other types of rule anomaly effectively. A visualized firewall rule anomaly removal mechanism has been realized based on the approach and the performance evaluations show its effectiveness and efficiency. As an augmented enhancement, the novel mechanism is integrated properly in our previously developed anomaly diagnosis system.

As the next steps, to accommodate the new demands of anomaly diagnosis, more interesting ingredients and many of technical challenges have to be taken into account [16], e.g., migrating the current mechanisms to IPv6/IoT networking environment, adding inspection functions for behavior mismatching among firewalls, and handling port configuration anomalies. For these, with hard working a whole new version is expected to be rolled out at the end of 2020.

Acknowledgments

This work is supported by MOST, R.O.C., under contract MOST-108-2221-E-035-030.

References

[1] E. Al-Shaer, *Automated Firewall Analytics*, Springer, 2014.

[2] E. Al-Shaer, H. Hamed, R. Boutaba, M. Hasan, Conflict Classification and Analysis of Distributed Firewall Policies, *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 10, pp. 2069-2084, October, 2005.

[3] K. Golnabi, R. K. Min, L. Khan, E. Al-Shaer, Analysis of Firewall Policy Rules Using Data Mining Techniques, *2006 IEEE/IFIP Network Operations and Management Symposium*, Vancouver, BC, Canada, 2006, pp. 305-315.

[4] A. X. Liu, A. R. Khakpour, J. W. Hulst, Z. Ge, D. Pei, J. Wang, Firewall Fingerprinting and Denial of Firewalling Attacks, *IEEE Transactions on Information Forensics and Security*, Vol. 12, No. 7, pp. 1699-1712, July, 2017.

[5] T. Wong, On the Usability of Firewall Configuration, *The 4th Symposium on Usable Privacy and Security*, Pittsburgh, PA, USA, 2008, pp. 180-185.

[6] T. Samak, A. El-Atawy, E. Al-Shaer, FireCracker: A Framework for Inferring Firewall Policies Using Smart Probing, *The IEEE International Conference on Network Protocols*, Beijing, China, 2007, pp. 294-303.

[7] E. Al-Shaer, *Automated Firewall Analytics: Design, Configuration and Optimization*, Springer, 2014.

[8] C. S. Chao, S. J.-H. Yang, Towards a Usable Anomaly Diagnosis System among Internet Firewalls' Rules, *Journal of Internet Technology*, Vol. 20, No. 3, pp. 789-799, May, 2019.

[9] Y. Yin, Y. Katayama, N. Takahashi, Detection of Conflicts Caused by a Combination of Filters Based on Spatial Relationships, *Journal of Information Processing Society of Japan*, Vol. 49, No. 9, pp. 3121-3135, September, 2008.

[10] Y. Yin, R. S. Bhuvaneswaran, Y. Katayama, N. Takahashi, Implementation of Packet Filter Configurations Anomaly Detection System with SIERRA, *The 7th International Conference on Information and Communications Security*, Bejing, China, 2005, pp. 467-480.

[11] A. Voronkov, L. H. Iwaya, L. A. Martucci, S. Lindskog, Systematic Literature Review on Usability of Firewall Configuration, *ACM Computing Survey*, Vol. 50, No. 6, Article No. 87, January, 2018.

[12] C. S. Chao, S. J.-H. Yang, A Novel Three-Tiered Visualization

Approach for Firewall Rule Validation, *Journal of Visual Languages and Computing*, Vol. 22, No. 6, pp. 401-414, December, 2011.

[13] E. Horowitz, S. Sahni, S. Anderson-Freed, *Fundamentals of Data Structures in C*, 2nd Ed., Silicon Press, 2007.

[14] E. Al-Shaer, A. El-Atawy, T. Samak, Automated Pseudo-Live Testing of Firewall Configuration Enforcement, *IEEE Journal on Selected Areas in Communications*, Vol. 27, No. 3, pp. 302-314, April, 2009.

[15] C. S. Chao, I. T. Jeng, A Genetic Algorithm for Firewall Rule Anomaly Optimization, *TANet2018*, 2018, Taichung, Taiwan, P-0003.

[16] M. M. Noor, W. H. Hassan, Current Research on Internet of Things (IoT) Security: A Survey, *Computer Networks*, Vol. 148, pp. 283-294, January, 2019.

Biographies

Chi-Shih Chao currently is an associated professor at the Communications Engineering Dept. of Feng Chia University, Taiwan. His research interests include network security, network fault management, high-speed networks, and wireless LANs. Dr. Chao received the Annual Best Paper Awards from Taiwan *TANet* in 2015 and *IMP* in 2016, respectively. He also serves for plenty of relevant conferences, journals, and industrial committees. In addition, he is a member of IEEE and Phi-Tau-Phi.

Stephen J. H. Yang is the Vice President of Asia University, Taiwan. He is also associated with the National Central University as the Distinguished Professor of Department of Computer Science & Information Engineering. Dr. Yang received his Ph.D. degree in Electrical Engineering & Computer Science from the University of Illinois at Chicago in 1995. Dr. Yang has published over 60 SSCI/SCI journal papers, his research interests include Big Data, learning analytics, Artificial Intelligence, educational data mining, and MOOCs. Dr. Yang received the Outstanding Research Award from Ministry of Science & Technology (2010) and Distinguished Service Medal from Ministry of Education (2015).