A Bit Vector-Based Diagnosis Mechanism for Firewall Rule Anomalies in IPv6 Networking Environment

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 what some consider to be the most essential devices which can safeguard networks. Misconfigurations of firewall rules often lead to rule anomalies which can be easily used by network attacks to paralyze the managed network. However, finding such rule anomalies is no easy task due to its time-consuming, laboring, strenuous characteristics. What's worse is, with the massive and increasing deployment of IPv6 in the current Internet, anomaly diagnosis for firewall rules becomes even harder. In this paper, a bit vector-based anomaly diagnosis approach is proposed and realized where it can pinpoint anomalies among IPv6 firewall rules not only effectively, but also much more efficiently and more easily. As a result, a visualized platform for our IPv6 firewall rule anomaly diagnosis has been implemented and comprehensive performance evaluations on anomaly diagnosis have been conducted also, in which our developed approach shows its excellence and feasibility.

Keywords: Rule anomalies in IPv6 firewalls, BST-based vectorization, Rule anomaly diagnosis, Diagnosis visualization and system usability

1 Introduction

As the development and applications of IoT and 5G become more popular nowadays, Internet has already embraced the increasing and massive deployment of IPv6. For this, network attacks become far fiercer than ever before since the amount of things that can be able to get access to Internet has become fairly large, which makes the amount of network attacks reach a new all-time high [1]. A network firewall is configured with packet filtering rules in its ACL (Access Control List) file which is used to inspect the passing packets and then decide whether they can pass or not, to shield the protected network as well as its estate from being attacked. The task of editing, ordering, and distribution of these rules should be done with caution and deliberation because misconfigurations of these rules can give rise to rule anomalies where malicious attackers can use them to launch unpredictable network attacks [2-3]. However, network administrators are often stranded by this task since it can be found that there are usually hundreds, or thousands of rules in an ACL file and these rules can affect mutually. It reveals why researchers like to compare the firewall configuring task to programming a distributed system in assembly language [4]. And now, the introduction of IPv6 lets this task much more troublesome, though really usable systems or platforms which can handle the task in IPv6 networking environment are still hard to be found.

The research work of E. Al-Shaer and H. Hamed are most acclaimed and renown worldwide in this field. They are the first ones who 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 [5-6]. 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 [7]. Y. Yin et al. [8] 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 [9]. 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. Still, a clean-slate reconstruction of the SIERRA tree is very possibly unavoidable if a simple rule deletion or insertion is done [7]. 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.

We have gotten involved in this field over a decade

*Corresponding Author: Chi-Shih Chao; E-mail: cschao@fcu.edu.tw
DOI: 10.53106/160792642021072204014

and the research results were hailed as one of the most useable systems in the world [10]. At 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 [5]. With the innate characteristic of this tree structure, local diagnosis results can be easily integrated for cross/inter-firewall rule anomaly diagnosis [11]. 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, called 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 [7].

In addition to the most-watched achievements mentioned above, an abundance of related works and efforts have already done in the field [10, 12-13]. But, till now, no clue shows they are capable of undertaking the rule anomaly diagnosis for IPv6 firewalls, not to mention a useful system which can truely facilitate diagnosis interactions with network administrators or experts, since the expansion of the IPv6 addressing makes the anoamly diagnosis too much of a hassle. To address the need, in this paper, our anomaly diagnosis approach is proposed and implemented. In contrast to the other achievements, a new data structure – bit vector is employed and realized in our work to depict the relationships among rules as well as the location of the filtering area of a rule on a two-dimensional traffic filtering diagram, where it can give a boost of figuring out rule anomalies for IPv6 firewall not only effectively, but also much more efficiently and easilierly. A brand new visualized interface is also developed and integrated in our developed prototype mechanism to effectively help users deal with rule anomalies within IPv6 firewalls. The rest of this paper is organized as follows: We organize Session 2 to brief the basis of rule anomalies and how our two-phased procedure can accomplish rule vectorization with a binary search tree-based (or BST-basd for short) methodology. In Session 3, we spell out how our rule anomaly diagnosis works via the calculated bit vectors on a two-dimensional traffic filtering diagram with simple OR plus AND logic operations. As a result, Section 4 presents the visualizsed graphic user interface of our prototype mechanism. Beyond that, system implementation and performance evaluations for our developed mechanism are also detailed. At last, Section 5 concludes this paper and shows some directions of our system development in the upcoming future.

2 Rule Anomalies and BST-Based Vectorization

2.1 Firewall Rule Anomalies

For the anomalies between firewall rules, they are completely defined and classified by E. Al-Shaer et al [5-6]. 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 can not be triggered due to the existence of R1. On the contrary, 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 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.

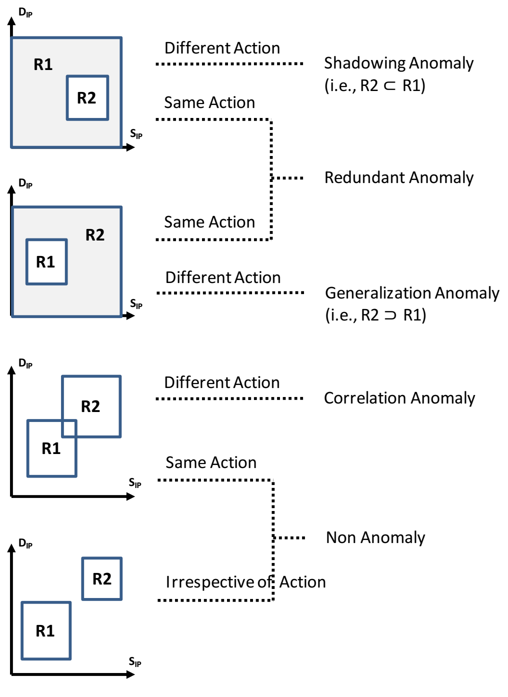


Figure 1. Types of firewall rule anomalies

2.2 Rule Vectorization – Phase I

To figure out the anomalies defined in Figure 1 in a firewall rule set, it can be found that the filtering area of each rule on a two-dimensional traffic-filtering diagram should first be located or indicated, no matter what diagnostic methods will be adopted. And then, the spatial relationship between each pair of rules on the diagram can be calculated by comparing the location coordinates of their filtering areas. Unlike in IPv4 firewalls, such coordinate comparing or locating for rules in IPv6 firewalls will cost abundant computing resources, since IPv6 addresses is 128-bit long, rather than 32-bit long. For the reason, a brand new coordinating system which can facilitate indicating the relationships between the filtering areas of IPv6 firewall rules in a two-dimensional traffic filtering diagram is needed. In our work, a BST-basd bitwise vectorization method is proposed to fasten the calculation of spatial relationships among IPv6 firewall rules. Figure 2 shows the example used for explaination of the conecpt behind our vectorization method which encodes the spatial relationships among filtering rules. In Figure 2, there are three lines, *x*, *y*, and *z* on the axis, with their corresponding durations, which can be depicted in the form of a BST shown in Figure 3. Each node E_i in the BST corresponds to an integer interval on the axis and in this fasion, the intergal duration of each line can be represneted as Figure 4; e.g, the duration of line *x* is [10, 20] and can be repesented as $\{E_1, E_2, E_3\}$ since its duration spans three intervals: Interval 1, 2, and 3.

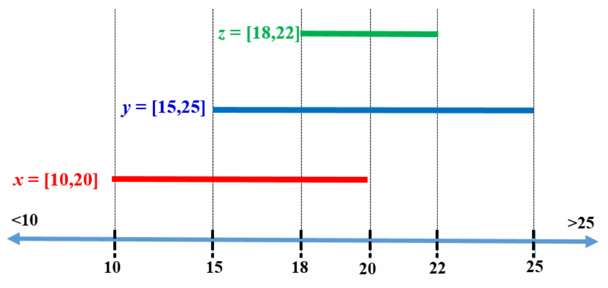


Figure 2. Example of three integer lines

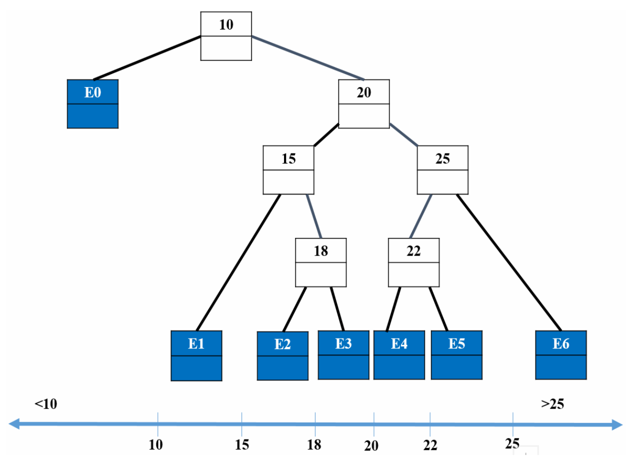
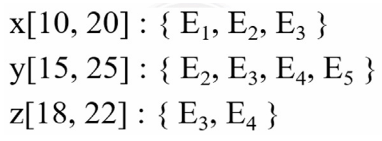


Figure 3. The BST corresponding to Figure 2



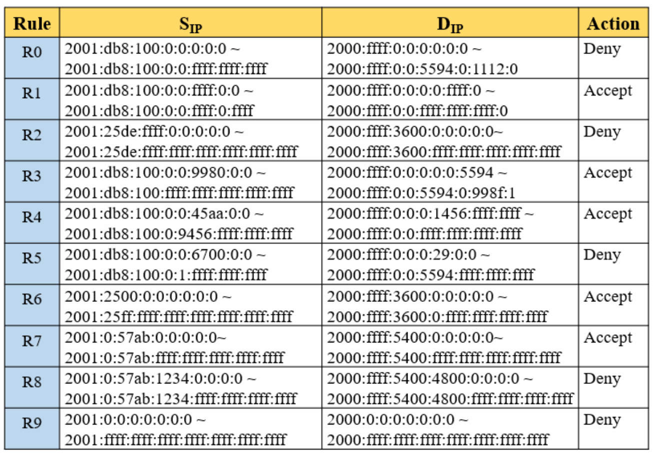
$x[10, 20] : \{ E_1, E_2, E_3 \}$

$y[15, 25] : \{ E_2, E_3, E_4, E_5 \}$

$z[18, 22] : \{ E_3, E_4 \}$

Figure 4. BST-based representation for each line duration in Figure 2

By utilizing the concept of our BST-based duration representation, two BST trees can be built; one for the S_{IP} fields and the other for the D_{IP} fields of a set of firewall filtering rules, to record the spatial locations of these rules on a two-dimensional traffic filtering diagram. The example IPv6 rule set used for explanation throughout this paper is shown in Figure 5 and the filtering area of each rule on the traffic filtering diagram are depicted in Figure 6. Figure 7 and Figure 8 are the corresponding BSTs for source IP fields and destinaion IP fields of the IPv6 rule set, built upon with the concept just mentioned, where R_i.S indicates the starting IP address of rule R_i and R_i.E indicates the ending IP address of rule R_i.



Rule	S_{IP}	D_{IP}	Action
R0	2001:db8:100:0:0:0:0:0 ~ 2001:db8:100:0:0:ffff:ffff:ffff	2000:ffff:0:0:0:0:0:0 ~ 2000:ffff:0:0:5594:0:1112:0	Deny
R1	2001:db8:100:0:0:ffff:0:0 ~ 2001:db8:100:0:0:ffff:0:ffff	2000:ffff:0:0:0:0:ffff:0 ~ 2000:ffff:0:0:ffff:ffff:ffff:0	Accept
R2	2001:25de:ffff:0:0:0:0:0 ~ 2001:25de:ffff:ffff:ffff:ffff:ffff:ffff	2000:ffff:3600:0:0:0:0:0~ 2000:ffff:3600:ffff:ffff:ffff:ffff:ffff	Deny
R3	2001:db8:100:0:0:9980:0:0 ~ 2001:db8:100:ffff:ffff:ffff:ffff:ffff	2000:ffff:0:0:0:0:0:5594 ~ 2000:ffff:0:0:5594:0:998f:1	Accept
R4	2001:db8:100:0:0:45aa:0:0 ~ 2001:db8:100:0:9456:ffff:ffff:ffff	2000:ffff:0:0:0:1456:ffff:ffff ~ 2000:ffff:0:0:ffff:ffff:ffff:ffff	Accept
R5	2001:db8:100:0:0:6700:0:0 ~ 2001:db8:100:0:1:ffff:ffff:ffff	2000:ffff:0:0:0:29:0:0 ~ 2000:ffff:0:0:5594:ffff:ffff:ffff	Deny
R6	2001:2500:0:0:0:0:0:0 ~ 2001:25ff:ffff:ffff:ffff:ffff:ffff:ffff	2000:ffff:3600:0:0:0:0:0 ~ 2000:ffff:3600:0:ffff:ffff:ffff:ffff	Accept
R7	2001:0:57ab:0:0:0:0:0~ 2001:0:57ab:ffff:ffff:ffff:ffff:ffff	2000:ffff:5400:0:0:0:0:0~ 2000:ffff:5400:ffff:ffff:ffff:ffff:ffff	Accept
R8	2001:0:57ab:1234:0:0:0:0 ~ 2001:0:57ab:1234:ffff:ffff:ffff:ffff	2000:ffff:5400:4800:0:0:0:0 ~ 2000:ffff:5400:4800:ffff:ffff:ffff:ffff	Deny
R9	2001:0:0:0:0:0:0:0 ~ 2001:ffff:ffff:ffff:ffff:ffff:ffff:ffff	2000:0:0:0:0:0:0:0 ~ 2000:ffff:ffff:ffff:ffff:ffff:ffff:ffff	Deny

Figure 5. Example IPv6 rule set

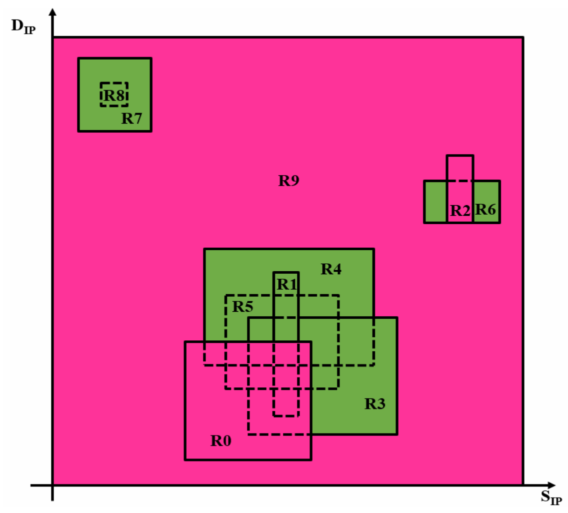


Figure 6. Corresponding traffic filtering diagram for Figure 5

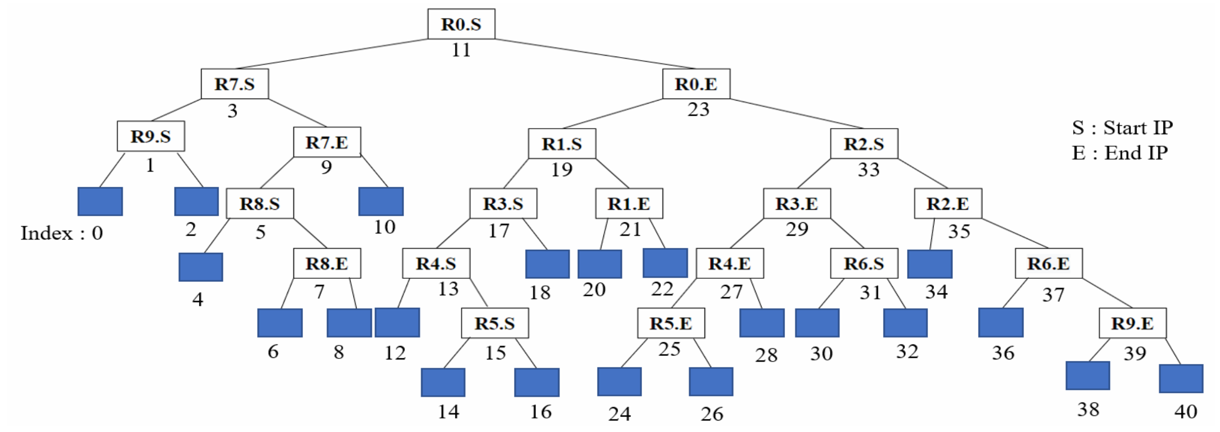


Figure 7. BST for rules' source IP fields

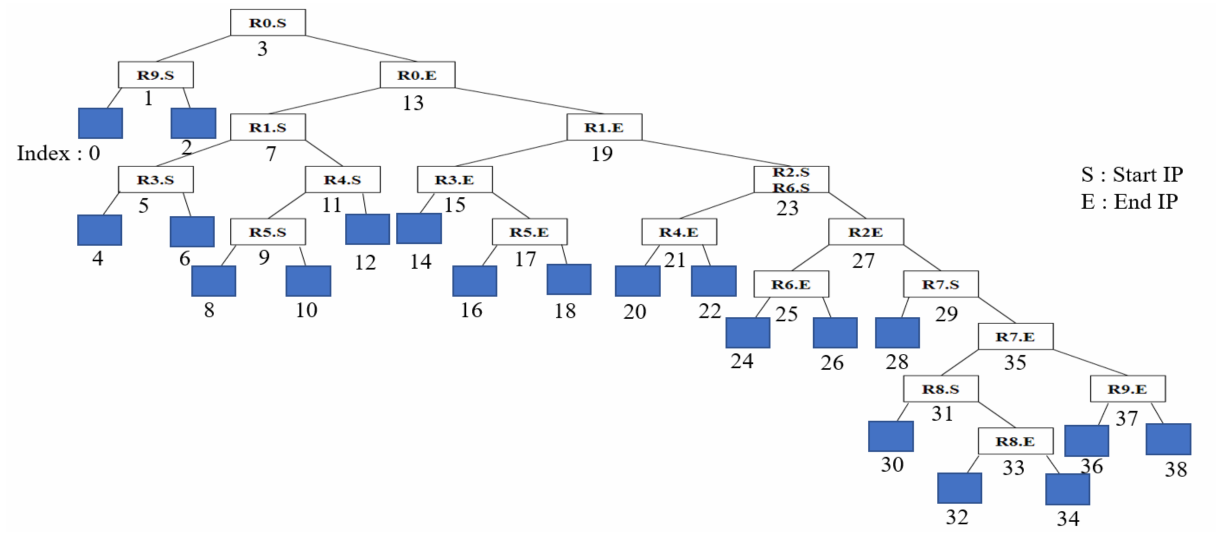


Figure 8. BST for rules' destination IP fields

2.3 Vectorization – Phase II

With the construction of BSTs in Phase I, in this phase, an index will be marked to each one of nodes of BTSs, using an in-order traversal of a binary tree:

Step 1: Travse the BST from the root down to the first External node (the left-most node), marking it as index 0. For example in Figure 7, we can travese the BST of the S_{IP} fields from root R0.S down to the left-most node of this tree; i.e., the left child of R9.S.

Step 2: Use this External node as the very beginning node, starting an in-order tree traversal, to index each one of the nodes in this BST, as the order of being visited. In Figure 7, the index value of R3.E is 29 standing for that it is the 29th node being visited by our in-order tree traversal.

Step 3: Find the corresponding node indexes in the BSTs for the source IP address and destination IP address of each filtering rule, and vectorize the filtering rule's duration. For example in Figure 7, the starting and ending IP addresses in the S_{IP} fields of rule R0 are R0.S and R0.E in the BST, which are indexed as 11 and 23, respectively. So, like Figure 9, nodes indexed 12, 14, 16, 18, 20, and 22 are set to 1 and the others are 0, representing R0 effective filtering durations in S_{IP}. In this way, nodes R1.S and R1.E are indexed as 19 and 21, respectively, the node indexed 20 can be set to 1 while the others are 0, and then we append the bit values to the results of Figure 9 (as shown in Figure 10).

Step 4: By the same token, after each of the filtering rules is vectorized and appended, two vector tables for the S_{IP} fields and D_{IP} fields of the rule set can be obtained (shown in Figure 11 and Figure 12), in which an interval indexed i with a bit vector with n elements means there are n rules in total in the input rule set and the jth element of the vector will be 1 if rule j has filterig effects on interval i.

3 Rule Anomaly Diagnosis

In comparison with [5] which needs conducting a FSM-based comparison between each pair of filtering rules for anomaly diagnosis, our approach wipes out those rules which have no relationship (i.e., no intersection) on the traffic filtering diagram and put the rest into our diagnosis procedure with a refined FSM, to fasten the rule anomaly diagnosis.

Index	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38
Vector	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Figure 9. Results after R0's S_{IP} vectorization

Index	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38
Vector	00	00	00	00	00	10	10	10	10	11	10	00	00	00	00	00	00	00	00

Figure 10. Results after R1's S_{IP} vectorization

Index	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38
Vector	00000 00001	00000 00101	00000 00111	00000 00101	00000 00001	10000 00001	10001 00001	10001 10001	10011 10001	11011 10001	10011 10001	00011 10001	00011 00001	00010 00001	00000 00001	00000 01001	00100 01001	00000 01001	00000 00001

Figure 11. Vectorization results for S_{IP} fields

Index	2	4	6	8	10	12	14	16	18	20	22	23	24	26	28	30	32	34	36
Vector	00000 00001	10000 00001	10010 00001	11010 00001	11010 10001	11011 10001	01011 10001	01001 10001	01001 00001	00001 00001	00000 00001	00100 01001	00100 01001	00100 00001	00000 00001	00000 00101	00000 00111	00000 00101	00000 00001

Figure 12. Vectorization results for D_{IP} fields

3.1 Anomaly Diagnosis

After our bit vector-encoding of the traffic filtering diagram, in the next step, we turn to figure out those rules which have spatial relationships; e.g., overlapping filtering effects shown in Figure 1, and get rid of the other rules which do not have the relationships, to accelerate the anomaly diagnosis. [8] propose an approach to isolate all of the problematic areas on the traffic filtering diagram, which could trigger rule anomalies, by conducting the AND logic operation on each pair of source vectors and destination vectors. For instance, in Figure 13, if we conduct bitwise AND logic operation on vector ❶ (01011010) from S_{IP} and vector ❷ (01011100) from D_{IP}, the result (01011000) with three non-zero elements in the vector signifies that rules R1, R3, and R4 all have filtering effects on this block indicated by these two vectors. It also means these three rules have spatial relationship among them and further checks for types of rules anomalies are needed. Given another example, if we AND vector ❸ (01000110) from S_{IP} and vector ❹ (01101100) from D_{IP}, we can obtain (01000100) which means there are two rules R1 and R5 within this block, and further inspection for rule anomaly types between R1 and R5 would proceed. Yet, their method revolves more around the isolation of problematic areas/blocks on the traffic filtering diagram, and thus extra data structures along with the corresponding algorithms are needed for the diagnosis of rule anoamlies.

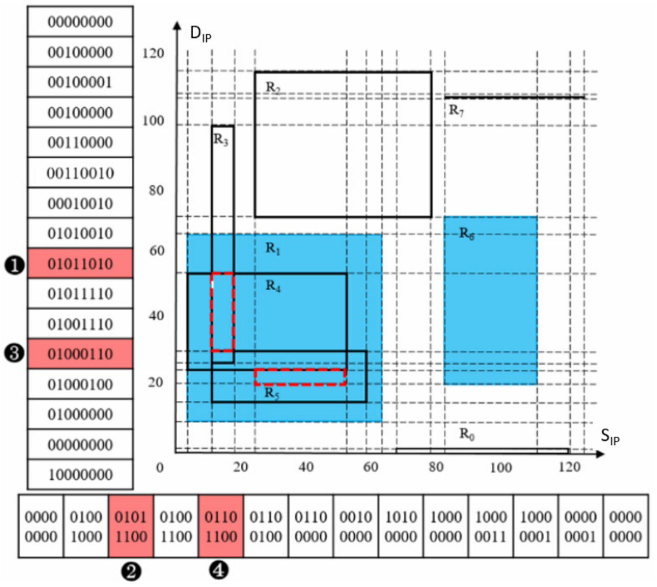


Figure 13. Isloation of problematic filtering areas

With our bit vector-encoding of the traffic filtering diagram, an efficient and effective rule anomaly diagnosis is developed, in which no efforts for extra data strucutures have to be done. After having the vectorization results (as shown in Figure 11 and Figure 12), the algorithm CalculateOverlap in Figure 14 is built and used by the filtering rules to calculate the spatial relationships on the traffic filtering diagram between one rule and the others. In the case of Figure 6, if we want to know the spatial relationships on the traffic filtering diagram between rule R0 and the other rules,

```
CalculateOverlap(Rule i's Sip_Vector , Rule i's Dip_Vector )

Input    Rule i's Sip_Vector , Rule i's Dip_Vector
Output   Overlap_Vec

 1.  Temp_Sip_Vec = Rule i's Sip_Vector [0]
 2.  Temp_Dip_Vec = Rule i's Dip_Vector[0]

 3.  foreach  x ∈ Rule i's Sip_Vector
 4.      Temp_Sip_Vec  |= x
 5.  end foreach

 6.  foreach  y ∈ Rule i's Dip_Vector
 7.      Temp_Dip_Vec  |= y
 8.  end foreach

 9.  Overlap_Vec = Temp_Sip_Vec & Temp_Dip_Vec

10.  return Overlap_Vec
```

Figure 14. Algorithm CalculateOverlap()

the corresponding vectors of S_{IP} and D_{IP} containing R0 which are indicated by red rectangle in Figure 15(a) and Figure 15(b) will first be OR-ed separately in algorithm CalculateOverlap, seen in Figure 16. And then conduct an AND logic operation to these two OR-ed results, we can get a bit vector (1101110001) at the bottom of Figure 16, which represents rules R1, R2, R3, R4, R5, and R9 have spatial relationships or overlapping filtering effects with R0. In the same fasion, we can get the spatial relationships on the traffic filtering diagram among all rules and then input the calculated bit vector for each one of them into our refined FSM (Finite State Machines) in Figure 17, for the final diagnosis results (shown in Figure 18).

3.2 Time Complexity Analysis

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

(1) For Phase I of our rule vectorization in Section 2.2, it depends on the time of the establishment of two BSTs; one for the S_{IP} fields and the other for the D_{IP} fields of a set of firewall filtering rules. Thus, its time complexity can be represented as O($n$log$n$), where n is the the number of rules in the filtering rule set.

(2) The Phase II of our rule vectorization in Section 2.3, it depends on the time of traversal as well as search of two built BSTs to accomplish the vectoriztion of a two-dimensional traffic filtering diagram, and hence its time complexity can also be represented as O($n$log$n$), where n is the the number of rules in the filtering rule set.

S_{IP}

Index	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38
Vector	00000 00001	00000 00101	00000 00111	00000 00101	00000 00001	10000 00001	10001 00001	10001 10001	10011 10001	11011 10001	10011 10001	00011 10001	00011 00001	00010 00001	00000 00001	00000 01001	00100 01001	00000 01001	00000 00001

(a) S_{IP}

D_{IP}

Index	2	4	6	8	10	12	14	16	18	20	22	23	24	26	28	30	32	34	36
Vector	00000 00001	10000 00001	10010 00001	11010 00001	11010 10001	11011 10001	01011 10001	01001 10001	01001 00001	00001 00001	00000 00001	00100 01001	00100 01001	00100 00001	00000 00001	00000 00101	00000 00111	00000 00101	00000 00001

(b) D_{IP}

Figure 15. Vectors for R0's duration in (a) S_{IP} and (b) D_{IP}

Temp_Sip_Vec =1000000001
Temp_Dip_Vec =1000000001

Temp_Sip_Vec = 1000000001 ∪ 1000100001 ∪ 1000110001 ∪ 1001110001 ∪ 1101110001 ∪ 100110001
Temp_Dip_Vec = 1000000001 ∪ 1001000001 ∪ 1101000001 ∪ 1101010001 ∪ 1101110001 OR

Overlap_Vec = *Temp_Sip_Vec* ∩ *Temp_Sip_Vec*
= 1101110001 ∩ 1101110001
= 1101110001 AND

Figure 16. Calculated vector for R0's spatical relationships with the other rules

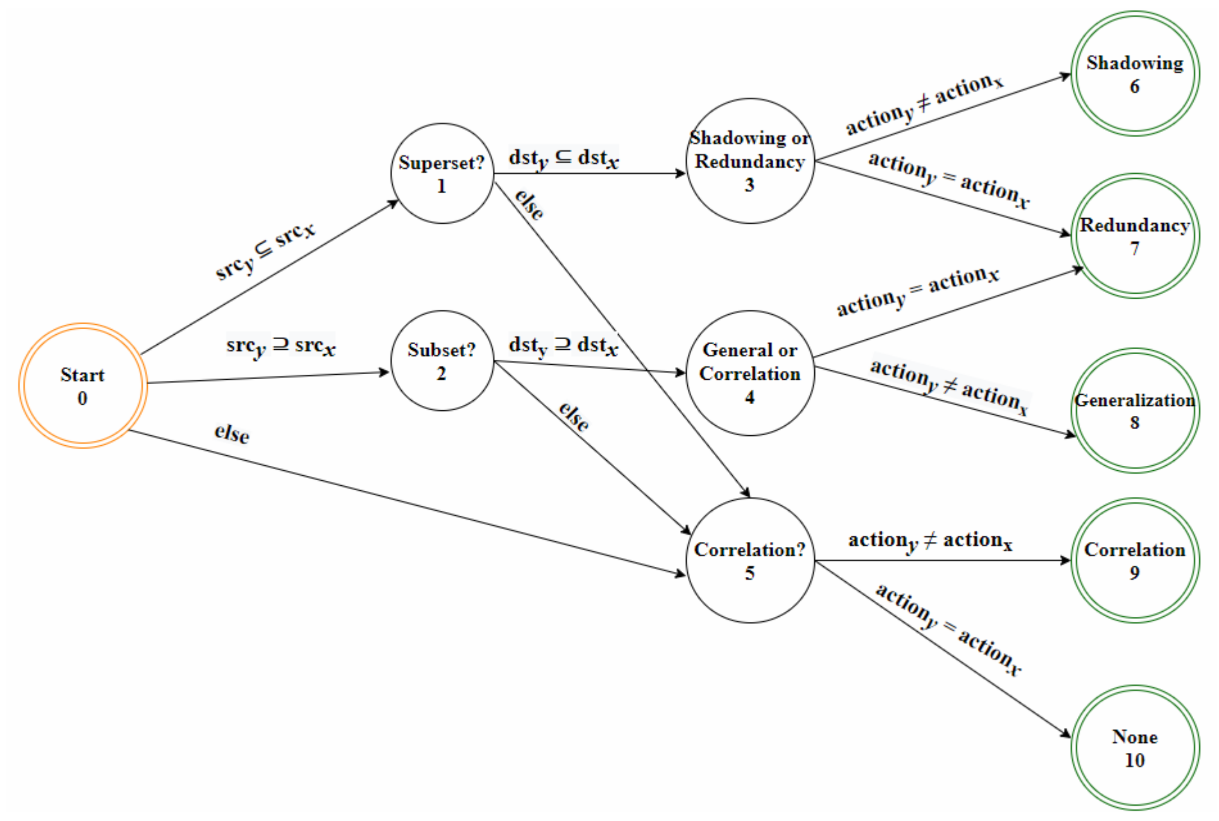


Figure 17. FSM for final rule anomaly diagnosis

Rule Pair	Anomaly	Rule Pair	Anomaly
R0, R1	Correlation	R3, R5	Correlation
R0, R3	Correlation	R3, R9	Generalization
R0, R4	Correlation	R4, R5	Correlation
R0, R9	Redundancy	R5, R9	Redundancy
R1, R9	Generalization	R6, R9	Generalization
R2, R6	Correlation	R7, R8	Shadowing
R2, R9	Redundancy	R7, R9	Generalization
		R8, R9	Redundancy

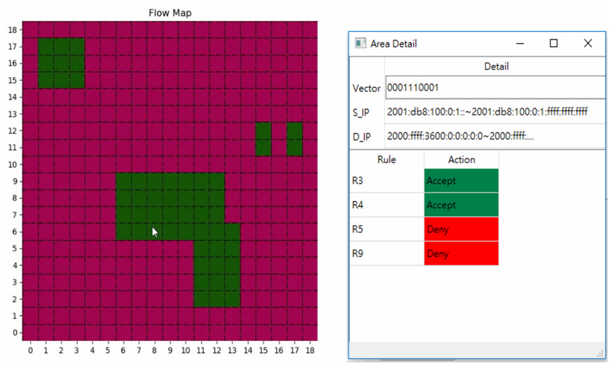
Figure 18. Final anomay diagnosis reults for rules in Figure 5

(3) In rule anomaly diagnosis mentioned in this section, to find rules which have spatial filtering relationships, Algorithm CalculateOverlap is designed and used. Since each rule in the rule set would go through the OR-plus-AND logic operations in CalculateOverlap with the vecrotizaton results (shown in Figure 11 and Figure 12), the time complexity goes to O(n^2), where n is the the number of rules in the rule set. And then, we input the rest of filtering rules into our refined FSM for final rule anomaly diagnosis, the time complexity would be O(m^2), where m is the number of rules without spatial relationships and smaller than n in most of the cases.

(4) From (1) to (3), we can summarize the rule anomaly diagnosis of our mechanism has time complexity of O(n^2).

4 System Implementation and Performance Evaluation

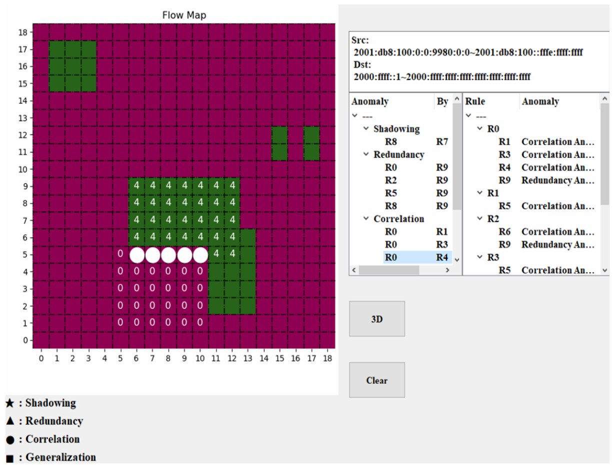
Figure 19 is our rule anomaly diagnosis mechanism for IPv6 firewalls with the visualized interface which shows the corresponding two-dimensional traffic filtering diagram of the input rule set in Figure 5 (dedicated for service port 80 in this case). While moving the mouse around on top of a block of the diagram, the corresponding vector of the block, S_{IP} and D_{IP} durations, and all the rules having effects on the block will be displayed at the right subwindow in Figure 19. Users can click the rule anoamly they are interested in, like Correlation Anomaly between R0 and R4 in Figure 20, and the visualized results containing anomaly type as well as locations/ relationships of these two rules will be displayed on the left subwindow in Figure 20. Our visualizaton interface also provides a 3-D augmented fucntion which provides users with 360 degree roation, zoom-in/out, and rule selection, for multiple and thorough diagnosis viewpoints (Figure 21).



	Detail
Vector	0001110001
S_IP	2001:db8:100:0:1::~2001:db8:100:0:1:ffff:ffff:ffff
D_IP	2000:ffff:3600:0:0:0:0:0~2000:ffff:...

Rule	Action
R3	Accept
R4	Accept
R5	Deny
R9	Deny

Figure 19. Traffic filtering block's information



Src:
2001:db8:100:0:0:9980:0:0~2001:db8:100::fffe:ffff:ffff
Dst:
2000:ffff::1~2000:ffff:ffff:ffff:ffff:ffff:ffff:ffff

```
Anomaly          By
---
  Shadowing
      R8         R7
  Redundancy
      R0         R9
      R2         R9
      R5         R9
      R8         R9
  Correlation
      R0         R1
      R0         R3
      R0         R4
```

```
Rule    Anomaly
---
  R0
     R1  Correlation An...
     R3  Correlation An...
     R4  Correlation An...
     R9  Redundancy An...
  R1
     R5  Correlation An...
  R2
     R6  Correlation An...
     R9  Redundancy An...
  R3
     R5  Correlation An...
```

★ : Shadowing
▲ : Redundancy
● : Correlation
■ : Generalization

Figure 20. Visualized diagnosis results between two certain rules

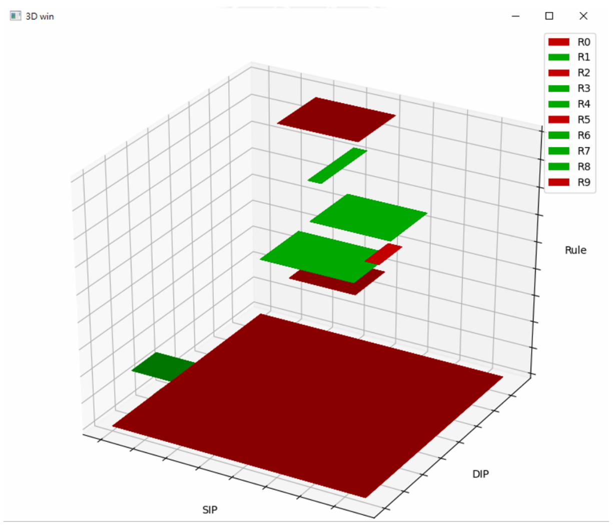


Figure 21. 3-D augmentation for rule anoamlies

Performance evaluation of our mechanism is conducted and comparisons are also made with two other anomaly diagnosis systems: one is from [5] using one-by-one (or pairwise) rule-comparing approach, and the other is our previously developed system running with a different data structure called Enhanced ARAR Tree [7]. 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. The following figures of experimental results show the time in seconds and memory space in kilobytes, respectively, where the time includes the creation of data structures and the execution of anomaly diagnosis, and rules are generated by ClassBenchv6 [14-15]. In addition, to obtain a thorough evaluation, our experiments take two distinct conditions of filtering spcace distributions into account: one is loose distriubion (Figure 22(a)), and the other is dense distribution (Figure 22(b)). On the basis of Figure 23 to Figure 26, it turns out that our diagnosis mechanism with the BST-based bitwise vectorization for IPv6 firewalls not only shows its excellence on execution time at a limited and tiny cost of memory space, but also reveals its exceptional stability in different filtering space distributions of rules.

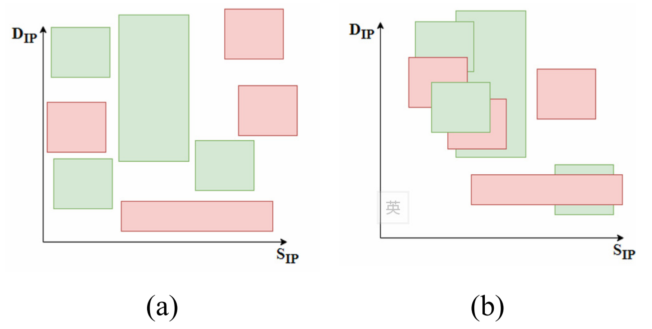


Figure 22. Filtering space distribution of rules

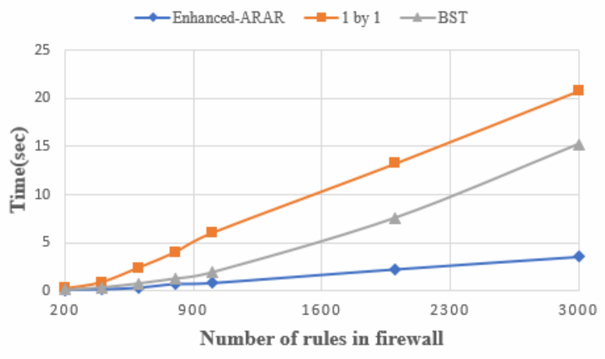


Figure 23. Time needed for rules in loose distribution

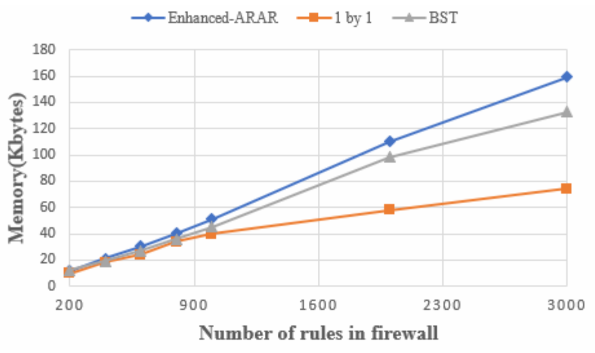


Figure 24. Space needed for rules in loose distribution

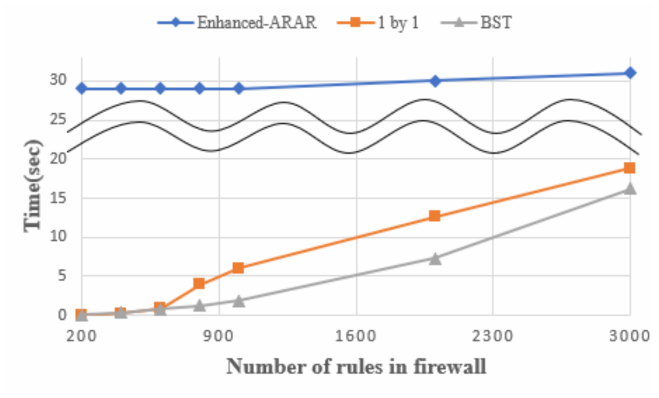


Figure 25. Time needed for rules in dense distribution

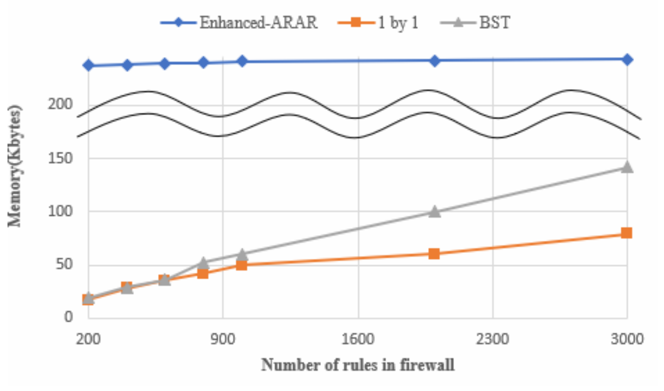


Figure 26. Space needed for rules in dense distribution

5 Conclusion and Future Work

With the demand of IP addresses is growing, IPv6 is emerging to solve the problem of IPv4 address exhaustion, whereas it hardens the work of anomaly diagnosis for rules in IPv6 firewalls. Our work aims at the development of a truly useful system for this challenge. Different from those researches which have been up so far in this topic, in this paper, a rule anomaly diagnosis mechanism dedicated for IPv6 firewalls is proposed, where it can provide a speedy and correct anomaly diagnosis along with its novel BST-based bitwise vectorization data structure and corresponding algorithms, and the performance evaluations show its effectiveness and efficiency. A visualized graphic user interface has also been designed, implemented, and integrated with our developed mechanism to complete our work towards the goal of real usability in IPv6 firewall management.

As the next steps, to accommodate the new demands of anomaly diagnosis, more interesting ingredients and many of technical challenges have to be considered, e.g., augmenting our system with functionality of parallel processing or data mining [16], adding inspection functions for behavior mismatching among firewalls, checking port configuration anomalies, and firewall diagnosis deployment in IoT networking environment [17]. For these, with hard working a new version is expected to be rolled out at the end of 2021. In addition, the emergence and development of SDN (Software Defined Network) technologies have impacted the rule anomaly diagnosis of firewalls and it has become one of our research targets in this field to go on.

Acknowledgments

This work is supported by MOST, R.O.C., under contract MOST-109-2221-E-035-058.

References

[1] 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.

[2] E. Al-Shaer, J. Lobo, L. Kalger, *Policies for Distributed Systems and Networks*, IEEE Press, 2008.

[3] H. Hamed, E. Al-Shaer, Taxonomy of Conflicts in Network Security Policies, *IEEE Communications Magazine*, Vol. 44, No. 3, pp. 134-141, March, 2006.

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

[5] 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.

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

[7] 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.

[8] Y. Yin, Y. Katayama, N. Takahashi, Detection of Conflicts Caused by a Combination of Filters Based on Spatial Relationships, *Journal of Information Processing*, Vol. 16, pp. 142-156, August, 2008.

[9] 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*, Beijing, China, 2005, pp. 467-480.

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

[11] 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.

[12] Y. Z. Cheng, W. P. Wang, J. X. Wang, H. D. Wang, FPC: A New Approach to Firewall Policies Compression, *Tsinghua Science and Technology*, Vol. 24, No. 1, pp. 65-76, February,

2019.

[13] C.-S. Chao, S. J. H. Yang, A Novel Mechanism for Anomaly Removal of Firewall Filtering Rules, *Journal of Internet Technology*, Vol. 21, No. 4, pp. 949-957, July, 2020.

[14] D. E. Taylor, J. S. Turner, ClassBench: A Packet Classification Benchmark, *IEEE/ACM Transactions on Networking*, Vol. 15, No. 3, pp. 499-511, June, 2007.

[15] 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.

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

[17] 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 2020, 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 PhD 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).