ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

DSSF: Decision Support System to Detect and Solve Firewall Rule Anomalies based on a Probability Approach

Suchart Khummanee[1] Phatthanaphong Chomphuwiset[2] and Potchara Pruksasri[3]

ABSTRACT

Currently, establishing a private network on the Internet is highly hazardous as attackers continuously scan computers for vulnerabilities within the connected network. The firewall ranked the highest as a network device is selected to protect unauthorized accesses and attacks. However, firewalls can effectively protect against assaults based on adequately defined rules without any anomalies. In order to resolve anomaly problems and assist firewall administrators manage the rules effectively, in this paper, a prototype of a decision support system has been designed and developed for encouraging administrators to optimize firewall rules and minimize deficiencies that occur in rules by using a probability approach. The experimental results clearly show that the developed model encourages experts and administrators of firewalls to make significant decisions to resolve rule anomalies. As a result, expert's confidence increases by 14.8%, and administrators' confidence soars similarly about 44.2%. The accuracy of correcting rule anomalies is 83%.

Article information:

Keywords: Firewall rule anomaly, Decision support system, Rule analysis, Probability of firewall rules

Article history:
Received: April 28, 2021
Revised: July 23, 2021
Accepted: September 27, 2021
Published: March 5, 2022
(Online)

DOI: 10.37936/ecti-cit.2022161.246996

1. INTRODUCTION

Firewalls are an essential security system for connecting to modern networks as they are designed to protect sites from intrusion and keep malicious attacks from penetrating the computer networks as shown in Figure 1. However, firewall protection measures are gradually reduced if the configured rules do not cover all activities taking place on the networks. Therefore, designing comprehensive firewall rules is of utmost importance for network protection. Basically, a firewall rule consists of a group of six conditional statements [1] that are used to determine what data can or cannot pass (Blocked), including source and destination IP address, source and destination port, protocol, and action. The number of firewall rules depends on the complexity of each organization's policy. As the number of rules increases, the rule anomalies also increase as well. In theory, a firewall rule anomaly is caused by any two rules that overlap but have different actions. For example, the first rule might allow every one of a company to access the Internet, but the second rule does not permit someone inside this company to surf the net. When considering both rules' behaviour, the second rule is never processed because the first rule prevents the second rule from from ever being used. Such an anomaly is called Shadowing. According to research based on firewall rule anomalies, such as Al-Shaer et al. [1], five types of rule anomaly have been identified: shadowing, correlation, generalization, redundancy, and irrelevancy. These are described in the Firewall background and related works section. Recently, a new type of anomaly has been introduced. This new anomaly is semantics loss of rules [2]. In a nutshell, when merging any two rules that overlap, they may lose their original meaning.

Dealing with rule anomalies, firewall rule researchers have presented unusual rule detection techniques, as well as methods for resolving conflicting rules, such as Al-Shaer et al. [3]. The firewall masters first contributed an algorithm to detect the five types of anomalies that occur in rules, called the finite state diagram. However, it is the only method used for detecting anomalies, and it does not offer factual corrections. In subsequent studies, Alex X. Liu et al. [4] proposed a firewall decision diagram (FDD) to solve anomalies effectively, which was developed based on [3]. This approach always processes the preceding rules, which allows the firewall to execute the

[1,2,3]The authors are with Department of Computer Science, Faculty of Informatics, Mahasarakham University, Mahasarakham, Thailand 44000, E-mail: suchart.k@msu.ac.th, phatthanaphong.c@msu.ac.th and potchara.p@gmail.com

conflicting rules, and it still does not eliminate all rule conflicts. Thus, FDD has become a prototype in research on rule-based conflicts. Next, they [5] identified the root cause of conflict in rules and presented a solution to do it, called SDD, where firewall rule actions must only be within the same domain. For example, all the rules' actions are in an accepted form only (Domain = *accept*). On the other hand, if the domain is a denial (Domain = *deny*), all of the rule's actions are only denied. The following study involves the propositional logic concept [6]. They claimed that this approach could reduce the number of conflicting rules without any policy change, but it is still unable to address the semantics loss anomaly. FAME [7] is a management framework for detecting and resolving rule anomalies based on the average risk values. It is not designed to consider the anomaly rules as a whole, but it is considered rule-to-rule, respectively. The risk values are calculated from the Common Vulnerability Scoring System (CVSS) [8] which does not consider attacking vulnerabilities in the overview, but only considers some rules. Tools [9], [10] for analyzing the policies such Lumeta and Fang can detect rule anomalies, but are not yet able to correct them. Hari et al. [11] built an algorithm for detecting and resolving conflicts to enhance packet filtering capabilities. However, the algorithm can only detect some conflicts. Firewall behaviour-based grouping [12] was also used to resolve conflicts with firewall rules, and the authors claim that the method takes less processing time and has higher packet matching than FIREMAN [13]. FPQE [14] is an automated method used to resolve rule conflicts without administrative intervention. The rules are corrected by deletion if they are detected as a redundant anomaly, but rule-swapping is applied when a shadowing or correlation anomaly is detected. Additionally, some algorithms [15], [16] can detect and diagnose firewall abnormalities, but these methods are not based on actual evidence. In most of the methods mentioned above, no method perfectly corrects for anomalies in the firewall rules. Therefore, the responsibility to fix the anomalous rules is often the burden of the firewall administrator. In a realistic situation, firewall administrators have different skills in dealing with the anomaly rules that inevitably affect rule conflicts.

To encourage firewall administrators to have confidence for solving rule anomalies, this paper has designed and developed a system to support decision-making based on probability theory, which is computed in conjunction with the actual evidence: the number of matching packets of each rule, evidence for rule-making, expertise for building rules, and protocol priorities. This paper is organized as follows: Section 2 gives an overview of firewalls and related work. Section 3 presents the main contributions. Section 4 explains our research methodology. Finally, Section 5 presents the conclusion of this paper.

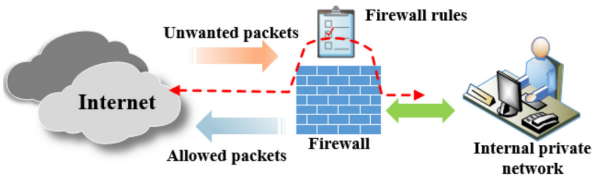


Fig.1: Firewall functions and connectivity.

2. FIREWALL BACKGROUND AND RELATED WORK

This section discusses the relevant theories for resolving firewall rule conflicts, including firewall fundamentals, data normalization, Bayes' theorem, and others.

2.1 Firewall symbols

The symbols used in this paper related to firewalls are shown in Table 1.

Table 1: Symbols and their meanings.

Symbol	Description
r_i	Any firewall rule
C_i	The part of the rule condition
A_i	The decision part of the rule
$f_i, ..., f_d$	Data fields within the condition part (C_i)
$D(f_i)$	The data range of f_i
r_x, r_y	r_x rule, and r_y rule
A_x, A_y	The decision of r_x and r_y
C_x, C_y	The condition of r_x and r_y
R	All rules in the firewall
P_i	Any packets flowing in and out
r_{x_m}, r_{y_m}	The meaning of r_x and r_y
r_{z_m}	New rules resulting from merging rules
S_IP, D_IP	Source and Destination IP address
S_Port, D_Port	Source and Destination port number
Pro	The protocol number
$P(r_i)$	Probability of any rule
x_1	The number of packet matching against rules
x_2	The weight of evidence
x_3	The weight of rule maker's expertise
x_4	The protocol priority

2.2 Rule definition and anomalies

Theoretically, a firewall rule consists of two main parts: the conditional part and the action part. Let r be a firewall rule, C as a conditional part, and A is an action part, then the pattern of a firewall rule:

$$r : C \Rightarrow A \tag{1}$$

In fact, the firewall always has more than a single rule. Therefore, Equation (1) is modified to Equation (2) as follows:

$$r_i : C_i \Rightarrow A_i \tag{2}$$

Where r_i as any rule, C_i is a condition and A_i is an action of rule r_i by $i \in [1, n]$, and n is a non-negative integer. Given f_i representing the domain of positive integers is a finite range, denoted $D(f_i)$. For example, the domain of the source ($D(f_1)$) and destination address ($D(f_3)$) in IP$_{v4}$ packet is [0, 2^{32} -

1], source ($D(f_2)$) and destination port ($D(f_4)$) is [0, 2^{16} - 1] and protocol ($D(f_5)$) is [0, 2^8 - 1]. C_i defines a set of packet fields over the fields f_1 through f_d specified as $f_1 \in F_1 \wedge f_2 \in F_2 \wedge ... \wedge f_d \in F_d$ where F_i is a subset of $D(f_i)$. A_i is either *accept* or *deny* for each rule. If all conditions (f_i) in C_i are true, the action is either *accept* or *deny* depending on the specified firewall rules as:

$r_i : (f_1 \wedge f_2 \wedge f_3 \wedge ... \wedge f_d)_i \Rightarrow accept_i$ or
$r_i : (f_1 \wedge f_2 \wedge f_3 \wedge ... \wedge f_d)_i \Rightarrow deny_i$

Given P_i as an IP$_{v4}$ packet over the d fields, $f_1, ..., f_d$, P_i is a tuple of $d(p_1, p_2, ..., p_d)$ where each $p_i (1 \leq i \leq d)$ is an element of $D(f_i)$. An IPv4 packet $(p_1, p_2, ..., p_d)$ matches r_i if and only if the condition $p_1 \in f_1 \wedge p_2 \in f_2 \wedge ... \wedge p_d \in f_d$. A set of rules $(r_1, ..., r_i)$ is valid when there is at least one rule matching against p_i. In order for the firewall to work properly, the condition of the last firewall rule is set to $f_1 \in D(f_1) \wedge ... \wedge f_d \in D(f_d)$, where every packet must be matched and the action is always dropped as shown in r_3 of Table 2. This last rule is called the implicit deny rule. Table 1 shows examples of firewall rules over the three condition fields $C(f_1, f_2, f_3)_i$ where $D(f_1) \in [0, 2^{32} - 1]$, $D(f_2) \in [0, 2^{16} - 1]$ and $D(f_3) \in [0, 2^8 - 1]$:

Table 2: Examples of the firewall rule.

$r_1 : f_1 \in [100, 200] \wedge f_2 \in [150, 400] \wedge f_3 \in [0, 80] \Rightarrow accept$
$r_2 : f_1 \in [0, 300] \wedge f_2 \in [300, 100] \wedge f_3 \in [6, 80] \Rightarrow accept$
$r_3 : f_1 \in [0, 2^{32} - 1] \wedge f_2 \in [0, 2^{16} - 1] \wedge f_3 \in [0, 2^8 - 1] \Rightarrow deny$

r_1 and r_2 in Table 2 are redundant because the condition parts of both rules overlap, but their actions are the same (*accept*). Besides, r_3 is a super set of r_1 and r_2, and they have different actions. Thus, r_3 is in conflict against r_1 and r_2. Basically, any packet p_i that reaches the firewall is always matched against the first rule, and if it cannot match the first rule, the packets are shifted and matched with the next rule in order, etc. If packets cannot be matched against any firewall rules, the last rule drops them automatically. As mentioned in the introduction section, rule anomalies are classified into six categories which will now be described in detail.

2.2.1 The shadowing anomaly

r_x is shadowed by r_y, if and only if $r_x \subseteq r_y$, and there are different actions as illustrated in Figure 2.

$$\begin{aligned} r_x : C_x \Rightarrow A_x, r_y : C_y \Rightarrow A_y \\ r_x, r_y \in R \wedge (C_x \subseteq C_y) \wedge \\ \neg(A_x \Leftrightarrow A_y) \wedge (x \neq y) \end{aligned} \quad (3)$$

R is all rules, and r_y is the rule executed before r_x.

Note: The arrow indicates the direction of the packet being matched against any rules in the firewall. The straight line r_y is the boundary of the r_y rule. The straight line r_x is the boundary of the r_x rule. The shaded area is the area that is obscured by the rules above.

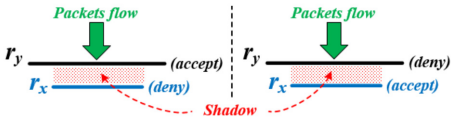


Fig.2: Shadowing anomaly.

2.2.2 Correlation anomaly

r_x is correlated against r_y if their intersection is not equal to $\emptyset$, r_x - $r_y \neq \emptyset$, r_y - $r_x \neq \emptyset$, and their actions are different as represented in Figure 3.

$$\begin{aligned} r_x : C_x \Rightarrow A_x, r_y : C_y \Rightarrow A_y \\ r_x, r_y \in R \wedge (C_x \cap C_y \neq \emptyset) \wedge (C_x - C_y \neq \emptyset) \wedge \\ (C_y - C_x \neq \emptyset) \wedge \neg(A_x \Leftrightarrow A_y) \wedge (x \neq y) \end{aligned} \quad (4)$$

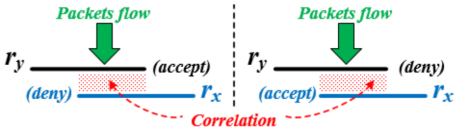


Fig.3: Correlation anomaly.

2.2.3 Generalization anomaly

r_x is generalized by r_y if and only if $r_y \subset r_x$ and there are different actions (Figure 4), where r_y is matched before r_x.

$$\begin{aligned} r_x : C_x \Rightarrow A_x, r_y : C_y \Rightarrow A_y \\ r_x, r_y \in R \wedge (C_x \supset C_y) \wedge \\ \neg(A_x \Leftrightarrow A_y) \wedge (x \neq y) \end{aligned} \quad (5)$$

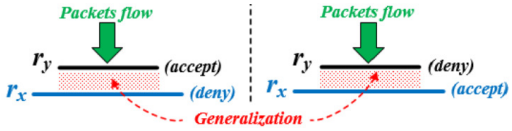


Fig.4: Generalization anomaly.

2.2.4 Redundancy anomaly

r_x is redundant against r_y if and only if $r_y \subseteq r_x$ and their actions are the same, as shown in Figure 5.

$$\begin{aligned} r_x : C_x \Rightarrow A_x, r_y : C_y \Rightarrow A_y \\ r_x, r_y \in R \wedge (C_x \supseteq C_y) \wedge (A_x \Leftrightarrow A_y) \wedge (x \neq y) \end{aligned} \quad (6)$$

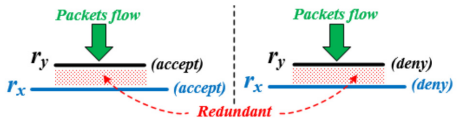


Fig.5: Redundant anomaly.

2.2.5 Irrelevance anomaly

An irrelevance anomaly (Figure 6) occurs in the firewall if there are no packets to match with any rules in the firewall (Disjoint). The cause of this anomaly is that the firewall administrator did not understand the network connection properly.

$$\begin{aligned} r_x : C_x \Rightarrow A_x, r_y : C_y \Rightarrow A_y \\ r_x, r_y \in R \wedge P_i \not\subset (C_x) \wedge P_i \not\subset (C_y) \wedge (x \neq y) \end{aligned} \quad (7)$$

P_i is a packet being processed by the firewall.

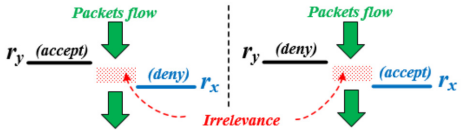


Fig.6: Irrelevance anomaly.

2.2.6 Semantics loss anomaly

The semantics loss anomaly [2] occurs due to the merger of any two rules, making the original meaning of some rule lose its meaning. According to Figure 7 (Left side), r_x and r_y were merged into r_z, resulting in all meaning of the r_x lost immediately. As in Figure 7 (Right side), r_x collapses with r_y, resulting in some meaning of r_x being lost.

$$\begin{aligned} r_x : C_x \Rightarrow A_x, r_y : C_y \Rightarrow A_y \\ r_x, r_y \in R \wedge r_{x_m} + r_{y_m} \wedge (x \neq y) \Rightarrow \\ r_{x_m} \vee r_{y_m} \vee r_{x_m+y_m} \vee r_{z_m} \end{aligned} \quad (8)$$

Let r_{x_m}, r_{y_m} be the meaning (m) of r_x and r_y respectively, "+" be a merging operator, $r_{x_m+y_m}$ is a combination of r_{x_m} and r_{y_m}, and r_{z_m} is a new meaning arising from merging of r_{x_m} and r_{y_m} together.

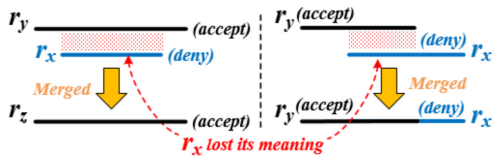


Fig.7: Semantics loss anomaly.

Note: Equations 3, 4 and 5 occur when rules overlap but decisions are different. Equations 6 and 8 occur when rules overlap but decisions are the same. Equation 7 means no packets can match any rules.

2.3 Data normalization

Since special data fields in the firewall rules are calculated as a probability in each rule, they have different sizes and ranges. Therefore, in this research it is necessary to normalize the data using the equation called Min-Max feature scaling [17]. It performs the scaling of variable-sized data ranges into the range [0, 1]. This technique is called unity-based normalization. Also, it can be used to scale the finite range of values in the dataset between any arbitrary points t_{max} and t_{min} as follows:

$$m^{'} = \frac{m - r_{min}}{r_{max} - r_{min}} \times (t_{max} - t_{min}) + t_{min} \quad (9)$$

Let $m^{'}$ presents the data is considered to be normalized by $m \in [r_{min}, r_{max}]$. r_{min} and r_{max} are specified as the minimum and maximum values of the evaluated data range. t_{min} and t_{max} are the minimum and maximum of the target range to be scaled.

2.4 Bayesian probability theory

The core of this research is to use probability to calculate the weights for each firewall rule, so Bayes' theory is considered the key mechanism of this research. Bayes' theorem [18] is a mathematical formula used to determine the conditional probability of events. Thus, it is a valuable tool for calculating conditional probabilities.

Let $A_1, A_2, ..., A_k$ be events that partition the sample space S, i.e., $S = A_1 \cup A_2 \cup ... \cup A_k$ and $A_i \cap A_j = \emptyset$ when $i \neq j$. Let B be an event on that space for which $P_r(B) > 0$. The Bayes' theorem is expressed in Equation 10.

$$P_r(A_j|B) = \frac{P_r(A_j)P_r(B|A_j)}{\sum_{j=1}^{k} P_r(A_j)P_r(B|A_j)} \quad (10)$$

This formula can be used to reverse conditional probabilities. If we know the probabilities of the events A_j and the conditional probabilities $P_r(B|A_j)$, $j = 1, ..., k$, the formula can be used to compute the conditional probabilities $P_r(A_j|B)$.

2.5 Moving average (MA)

The network traffic that flows in and out through the firewall is highly uncertain and fluctuates for several reasons such as daily user behaviour, attacks from worms and viruses, distributed denial-of-service attacks (DDoS), etc. Therefore, this research uses the moving average (MA) [17] technique to smooth out traffic disturbances from short-term aging. The following are two primary moving averages: Simple

Moving Average (SMA) and Exponential Moving Average (EMA). SMA is used to calculate the average of traffic over n periods.

$$SMA = \frac{a_1 + a_2 + a_3 + ... + a_n}{n} \quad (11)$$

a is an average in period n, and n is the number of periods. EMA is a moving average that considers the weighted average of a series of recent traffic to reflect the ongoing trend in the network traffic, and the Equation 12 is used to calculate EMA.

$$EMA_n = P_n \times k + EMA_{n-1} \times (1 - k) \quad (12)$$

EMA_n = the exponential moving average of the desired time. P_n = data of the desired period. EMA_{n-1} = exponential moving average of the previous period. k = smoothing factor (Calculated from $\frac{2}{T+1}$). T is the period used to calculate the smoothing factor.

2.6 Converting IP_{v4} to Integer

In this paper, a tree structure (m-array) is used to detect anomalies in firewall rules. The data stored in the tree's nodes are commonly integers to increase efficiency and ease processing. However, the IP_{v4} addresses in the firewall rules are formatted as $A_1.A_2.A_3.A_4$, where $A_{(1-4)} \in [0, 255]$. Therefore, they must be converted to integers before being stored and processed in the tree. Equation 13 is used to convert IP_{v4} addresses to integers.

$$IP'_{v4} = A_1 \times 2^{24} + A_2 \times 2^{16} + A_3 \times 2^8 + A_4 \times 2^0 \quad (13)$$

IP'_{v4} is a new IP_{v4} address to be converted to Integer.

2.7 Arithmetic mean and Cohen's kappa

The arithmetic mean in Equation 14, also called the sample mean ($\overline{x}$), is used to evaluate the satisfaction of firewall experts and firewall administrators with our proposed firewall, and Cohen's kappa coefficient ($\hat{K_F}$) [19] is applied to measure the inter-rater reliability using Equation 15.

$$x = \frac{1}{n} \times \sum_{i=1}^{n} x_i \quad (14)$$

$\overline{x}$ is the arithmetic mean. n is the number of data points. x_i is the value of the element at the ith position.

$$\hat{K_F} = \frac{\bar{P_a} - \bar{P_e}}{1 - \bar{P_e}} \quad (15)$$

$\bar{P_a}$ indicates the relative observed agreement among raters. $\bar{P_e}$ denotes the hypothetical probability of chance agreement.

2.8 Confusion Matrix

The Confusion Matrix [20] evaluates the predictive results of the developed model (or the results from the program) compared against the actual target values. True Positive (TP) is what the program predicts as "true", and the result is "true". True Negative (TN) is what the program predicts is "false", and the result has a "false" value. False Positive (FP) is what the program predicts as "true", but the result is "false". Finally, False Negative (FN) is what the program predicts as "false", but the result is "true". Thus, accuracy is measure of correctness that the results can be predicted precisely. The accuracy is computed with Equation 16.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (16)$$

3. MAIN CONTRIBUTIONS

This paper develops a system that promotes decision-making to enable administrators to solve rule anomalies occurring in firewall rules with confidence based on the available evidence by using probability as the system's core. This approach differs from a traditional firewall in that it makes recommendation when deciding to resolve rule conflicts, but for a conventional firewall, these decisions are solely the responsibility of the system administrator. Moreover, the entire firewall framework has been developed, starting from the lowest level of the operating system (System call) to the top level of the system –the firewall rule management application (Firewall User Interface: GUIF).

4. RESEARCH METHODOLOGY

The design and development of decision-making aids in solving rule anomalies consist of ten steps split into three phases, as shown in Figure 8.

4.1 PHASE I: Design and develop algorithms to detect firewall rules

4.1.1 Proof of the theory of probability for firewall rules (Step 1)

Let r be a firewall rule, and let x be an attribute field of the rule, and let S be a sample space. Then the conditional probability of r given x is shown in Equation (17), and a Vann diagram illustrates this in Figure 9(a).

$$P(x|r) = \frac{P(r \cap x)}{P(r)}, or \quad P(r|x) = \frac{P(r \cap x)}{P(x)} \quad (17)$$

According to Figure 9(b), given r_i as any rule, and x_k is any attribute (Special fields) of r_i, then

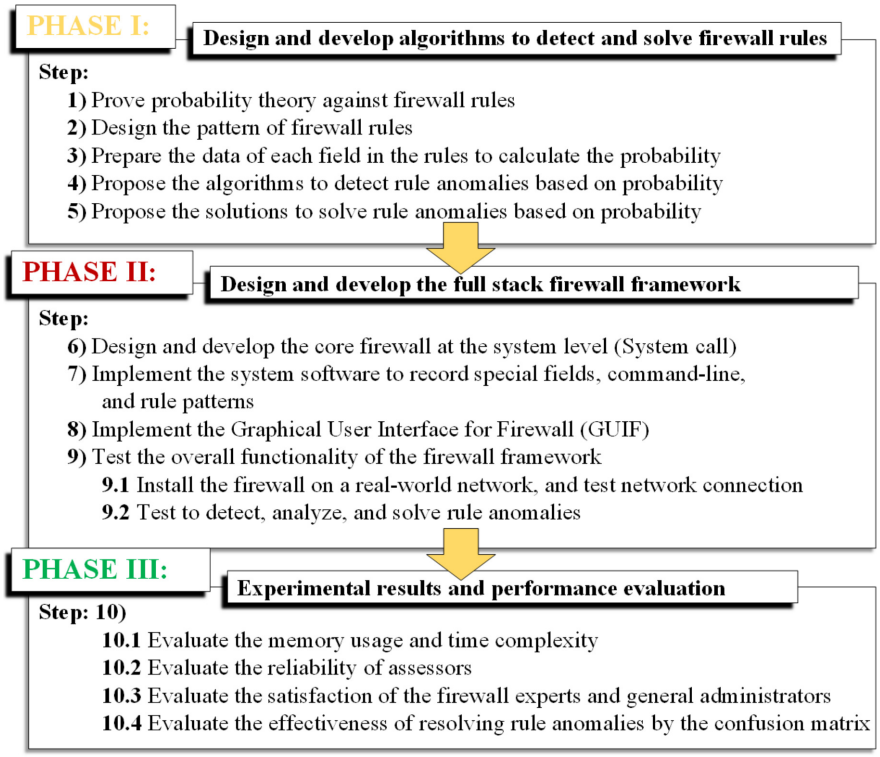
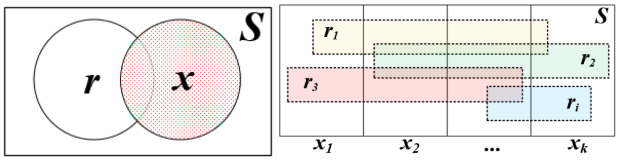


Fig.8: The steps of our research methodology.



(a) r given x in a Venn diagram. (b) r_i given x_k in the firewall rule space.

Fig.9: Conditional probability of r given x(a), and r_i given x_k(b).

the conditional probability of any firewall rule can be expressed in Equation 18.

$$\begin{aligned} x_i \cap x_k = \emptyset; \forall i,k; i \neq k, \\ x_1 \cup x_2 \cup x_3 \cup ... \cup x_k = S = 1, \\ P(r_i) = P(r_i \cap x_1) \cup P(r_i \cap x_2) \cup ... \cup P(r_i \cap x_k) \end{aligned} \tag{18}$$

From Equation (17), $P(r|x) = \frac{P(r \cap x)}{P(x)}$, thus $P(r \cap x) = P(x)P(r|x)$, or $P(x|r) = \frac{P(r \cap x)}{P(r)}$, then $P(r \cap x)$ is $P(r)P(x|r)$. According to the firewall rules since we know the value of $P(x)$, we choose $P(r \cap x) = P(x)P(r|x)$, and substitute i and k into the equation to get Equation 19.

$$P(r_i \cap x_k) = P(x_k)P(r_i|x_k) \tag{19}$$

Next, substitute Equation (19) to Equation (18) to get Equation 20.

$$\begin{aligned} P(r_i) = P(x_1)P(r_i \mid x_1) \cup P(x_2)P(r_i \mid x_2) \cup \\ ... \cup P(x_k)P(r_i \mid x_k), \\ P(r_i) = \sum_{i,k=1}^{n} P(x_k)P(r_i|x_k) \end{aligned} \tag{20}$$

x_k is any property considering when giving $P(r_i)$. Finally, we can substitute Equations (17), (19), and (20) into Bayes' rule in Equation (10), which is used widely in Bayesian inference as follows.

$$P(x_k|r_i) = \frac{P(r_i \cap x_k)}{P(r_i)} = \frac{P(x_k)P(r_i|x_k)}{\sum_{i,k=1}^{n} P(x_k)P(r_i|x_k)} \tag{21}$$

4.1.2 Design the pattern of firewall rules (Step 2)

According to r_i in Equation 2, in traditional firewalls, C_i has five fields ($f_1 \wedge f_2 \wedge ... \wedge f_5$), where f_1 = source IP$_{v4}$ address (S_IP), f_2 = source port (S_Port), f_3 = destination IP$_{v4}$ address (D_IP), f_4 = destination port (D_Port), and f_5 = protocol (Pro) respectively. However, such a firewall rule is not suitable for processing with the designed decision-making system. Thus, it is necessary to add the following four fields to the firewall rules: the number of packets matching ($f_6 = x_1$), evidence for creating rules ($f_7 = x_2$), rule-making expertise ($f_8 = x_3$), and pro-

Table 3: The scope of each data field in a firewall rule.

Field name	Description	Data range
$f_1(S\_IP)$	Source IP_{v4} address	$0-(2^{32}-1)$
$f_2(S\_Port)$	Source Port	$0-(2^{16}-1)$
$f_3(D\_IP)$	Destination IP_{v4} address	$0-(2^{32}-1)$
$f_4(D\_Port)$	Destination Port	$0-(2^{16}-1)$
$f_5(Pro)$	Protocol	$0-(2^{8}-1)$
$f_6(x_1)$	Number of packets matching	$0-n$ $(n \geq 0)$
$f_7(x_2)$	Evidence of creating rules: 0 = No evidence 1 = Administrator 2 = Head of the department 3 = Owner of the organization	0 – 3
$f_8(x_3)$	Rule-making expertise: 0 = Newbie 1 = Normal 2 = Professional 3 = High expertise	0 – 3
$f_9(x_4)$	Protocol priority: 1 = Low priority ... 9 = High priority	1 – 9
Action	Rule decision: 0 = *deny* (Dropped packets) 1 = *accept* (Allowed packets)	0 or 1

tocol priority ($f_9 = x_4$). Therefore, the new rule format is as follows.

$r_i : C(f_1 \wedge f_2 \wedge f_3 \wedge f_4 \wedge f_5 \wedge f_6 \wedge f_7 \wedge f_8 \wedge f_9)_i \Rightarrow A_i$, or $r_i : C(S\_IP \wedge S\_Port \wedge D\_IP \wedge D\_Port \wedge Pro \wedge x_1 \wedge x_2 \wedge x_3 \wedge x_4)_i \Rightarrow A_i$.

The scope of the data fields for each rule is shown in Table 3. Each of the fields in the rule has a range of data: S_IP and $D\_IP \in [0, 2^{32}-1]$, S_Port and $D\_Port \in [0, 2^{16}-1]$, $Pro \in [0, 2^{8}-1]$, the number of packets matching $(x_1) \in [0, n]$, evidence to verify rule creation (x_2), the rule maker's expertise points $(x_3) \in [0, 3]$, the weight indicating the priority of the protocol $(x_4) \in [1, 9]$, where n is not a negative integer, and the protocol weight value of 9 is the most significant. For example, if r_1 contains data fields like $S\_IP = 192.168.1.0/24$, $S\_Port = any$, $D\_IP = any$, $D\_Port = 80-85$, $Pro = all$, the packets matching $(x_1) = 2553$, evidence for creating rules $(x_2) = 1$, rule maker's expertise $(x_3) = 2$, protocol priority $(x_4) = 6$, and action = *accept*, then it can be written as a new firewall rule:

r_1 : 192.168.1.0/24 $\wedge$ *any* $\wedge$ *any* $\wedge$ 80-85 $\wedge$ *all* $\wedge$ 2553 $\wedge$ 1 $\wedge$ 2 $\wedge$ 6 $\Rightarrow$ *accept*, or

$r_1 : [3232235776, 3232236031] \wedge [0, 65535] \wedge [0, 4294967295] \wedge [80, 85] \wedge [0, 255] \wedge 2553 \wedge 1 \wedge 2 \wedge 6 \Rightarrow 1$

any and *all* indicate all members in each field of firewall rules. For example, *any* of $S\_IP = [0, 2^{32}-1]$ and *all* of $Pro = [0, 2^{8}-1]$, and *accept* is equal to 1. To write a rule in the most widely used pattern in practice, it can be written as follows:

r_1: 192.168.1.0/24 *any* *any* $80-85$ *all* 2553 1 2 6 *accept*

The examples of the new rule pattern that be used to illustrate the functionality of the prototype firewall developed are shown in Table 4.

4.1.3 Prepare the data of each field in the rules before calculating the probability (Step 3)

Some of the data fields in the firewall rules are not yet applicable to our proposed model. Thus, we need to convert them into appropriate data before calculating each rule's probability.

Converting IP_{v4} addresses: we use Equation 13 to convert IP_{v4} addresses to positive integers before storing these IP addresses (IP integer) on the tree structure. For example, 192.168.0.1/24 (192.168.1.0 - 192.168.1.255) is transformed from 3232235776 to 3232236031 (256 addresses).

Normalizing the packet matching: the packet matching is the number of matches between any packet and any firewall rule. The matched rule decides between *accept* or *deny*. The count of matches runs from the first rule creation up to the current time, with the firewall logging it to the log file every time a match occurs for each firewall rule. For example, referring to the Matching column (x_1) in Table 4, r_1 has a count of 2553 matches since the firewall rule was created until the present time. To normalize the number of packets matched against r_1 in the range 0 to 1, Equation 9 is used.

$$r_1(x_1^{'}) = \frac{2553-528}{3208-528} \times (1.0-0.0) + 0.0 = 0.756$$

$x_1^{'}$ is the number of matched packets to r_1 that has been normalized by the minimum matched packets of all rules in the firewall $(r_{min} = r_6) = 528$, maximum matched packets of all rules $(r_{max} = r_4) = 3208$, minimum target value $(t_{min}) = 0$, and maximum target value $(t_{max}) = 1$.

In real-world situations on computer networks, the number of packets that flow inward and outward through the firewall are continually fluctuating and unstable, caused by several factors such as network attacks, worms, computer usage behaviour, network usage rush hours, etc. Therefore, we use Equations 11 and 12 to reduce the number of extra packets fluctuate, achieve smoothness and consistency, and reduce packet disturbances over the firewall. Assume that the number of packets matched against r_i is logged in the log file every hour, thus within one day, it is logged 24 times, as in the following example: 2120, 2300, 1895, 2405, 2345, 1850, 1900, 2642, 2500, 1758, 1650, 2235, 2000, 2106, 1975, 2120, 2236, 3754, 2145, 3205, 1458, 2244, 2528, and 2225 respectively. We calculate the data of the first five hours by SMA (Equation 11) and the remaining hour's data by EMA (Equation 12). With the data range calculated by SMA, the user can adjust according to their needs. Let T be the number of matched packets in EMA (In this research, T equals 5). Thus, the smoothing factor (k) is 0.333 $(\frac{2}{5+1})$. Substitute k into the EMA equation to get the following result.

Table 4: Examples of the new firewall rules.

r_i	S_IP	S_Port	D_IP	D_Port	Pro	(x_1)	(x_2)	(x_3)	(x_4)	Action
1	192.168.1.0/24	*any*	*any*	80 - 85	*all*	2553	1	2	6	*accept*
2	192.168.1.10 - 50	*any*	202.28.34.20 - 60	80	*all*	1536	3	2	3	*deny*
3	192.168.1.20 - 40	*any*	202.28.34.30 - 70	80 - 90	*all*	2127	2	1	8	*deny*
4	192.168.1.20 - 30	*any*	202.28.34.20 - 30	80 - 82	*all*	3208	1	2	5	*deny*
5	192.168.1.0/24	*any*	202.28.34.0/24	30 - 90	*all*	1212	3	0	2	*accept*
6	192.168.1.0/24	*any*	*any*	*any*	*all*	528	0	3	9	*deny*

Remarks: **Matching**(x_1) = the number of packets matching, **Evidence**(x_2) = the evidence for creating rules, **Rule Maker**(x_3) = rules-making proficiency score, and **Priority**(x_4) = protocol priority

$$SMA_{1-5} = \frac{(2120 + 2300 + 1895 + 2405 + 2345)}{5} = 2213.0$$
$$EMA_6 = 1850 \times (0.333) + 2213 \times (1 - 0.333) = 2092.0$$
$$EMA_7 = 1900 \times (0.333) + 2092 \times (1 - 0.333) = 2028.0$$
$$\dots$$
$$EMA_{23} = 2528 \times (0.333) + 2285 \times (1 - 0.333) = 2366.0$$
$$EMA_{24} = 2225 \times (0.333) + 2366 \times (1 - 0.333) = 2319.0$$

The results obtained after the SMA and EMA calculations are shown in Figure 10. Notice that the number of packets matched against the rules is significantly smoother. All calculated results are rounded to positive integers because the unit of matched packets is the count number. The SMA again calculates the results obtained after the 24-hour EMA calculation to find the mean for each day (x_1) of each rule (r_i) as follows:

$$r_1(x_1) = \frac{\sum_{i=5}^{n} EMA_i}{n - H} = \frac{2213 + \dots + 2319}{24 - 5} = 2344$$

H is the period used in the past (Hours).

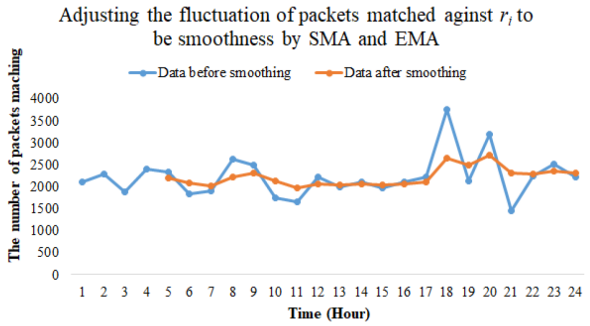


Fig.10: Smoothing the number of matched packets by SMA and EMA.

After calculating the average of x_1 each day, it is then further normalized to be in the range 0 to 1 $(x_1^{'})$ by Equation 9 as described above. x_1 of other rules that have been successfully normalized are shown in Table 5.

Normalizing the weight of evidence: This refers to documents, pieces of paper used to identify who approves to create firewall rules. Moreover, it specifies what level of privilege they have. In this paper, the evidence used to establish rules are divided into four levels (0-3). In level 0, there is no verification document. In level 1, the document has been verified by the administrator. In level 2, supervisors approve the documents. In the last level (3), the evidence approved by the owner of the organization. In this paper, Min-Max [17] is used to normalize the weight range to be form 0 to 1. For example, referring to the Evidence(x_2) column of r_1 in Table 4, it is equal to 1 when it is normalized by Equation 9:

$$r_1(x_2^{'}) = \frac{1 - 0}{3 - 0} \times (1.0 - 0.0) + 0.0 = 0.333$$

$m = 1$, $r_{min} = 0$, $r_{max} = 3$, $t_{min} = 0.0$, and t_{max} is equal to 1.0. The remaining weight of evidence in the other rules can be calculated using the same method shown in Table 5.

Normalizing the weight of rule maker's expertise: expertise in creating rules means a comprehensive understanding of how to design secure rules, non-vulnerability, and preventing anomalies. There are four levels of rule-building expertise, as well as evidence-building calculations. The newbie admins are those who have recently been assigned to configure firewalls and have the least experience (Level=0). System admins who have experience about 1 to 5 years to configure firewalls are called regular admins(1). For those who have a lot of experience and training or firewall customization, with working hours of 5 – 10 years are called professional admins(2). Finally, those who have received a lot of training and certificates about firewalls and have at least ten years of experience are known as expert admins(3). In other words, the experts always design less error-prone rules. The normalization of rule-building expertise is based on Equation 9. For example, Rule maker(x_3) column of r_1 in Table 4 is equal to 2. Thus, it can be normalized as follows.

$$r_1(x_3^{'}) = \frac{2 - 0}{3 - 0} \times (1.0 - 0.0) + 0.0 = 0.66$$

The way to normalize remaining rules is shown in Table 5.

Table 5: Examples of firewall rules after rule preparation.

r_i	S_IP	S_Port	D_IP	D_Port	Pro	(x_1)	(x_2)	(x_3)	(x_4)	Action
1	3232235776-..36031	0-65535	0-4294967295	80-85	0-255	0.75	0.33	0.66	0.37	1
2	3232235786-..35826	0-65535	3390841364-..41404	80	0-255	0.37	1.00	0.66	0.75	0
3	3232235796-..35816	0-65535	3390841374-..41414	80-90	0-255	0.59	0.66	0.33	0.12	0
4	3232235796-..35806	0-65535	3390841364-..41374	80-82	0-255	1.00	0.00	0.66	0.50	0
5	3232235776-..36031	0-65535	3390841344-..41599	30-90	0-255	0.25	1.00	0.00	0.87	1
6	3232235776-..36031	0-65535	0-4294967295	0-65535	0-255	0.00	0.00	1.00	0.00	0

Normalizing protocol priorities: protocols communicated on computer networks are always prioritized for the most effective communication. For example, video conferencing must guarantee a continuous signal of communication throughout the conversation. On the other hand, sending and receiving e-mail does not have to be urgent. Protocol priorities do not have to be the same depending on the policies of each organization. In this paper, prioritization of protocols is based on 3 GPP QoS Class Identification QCI categories [21] by IP Multimedia having the highest priority (Highest = 1), and Chat, FTP, and P2P having the lowest priority (Lowest = 9). Priority(x_4) column of r_1 in Table 4, it is a teleconference application with a priority of 6. Notice that the priority of the protocols calculated must always reverse priorities, such as from 9 to 1 and from 1 to 9. For example, 6 is reversed to 4. After processing with Equation 9, the result is as follows.

$$r_1(x_4^{'}) = \frac{4-1}{9-1} \times (1.0 - 0.0) + 0.0 = 0.37$$

$m = 4$ (Teleconference), $r_{min} = 1$, $r_{max} = 9$, $t_{min} = 0.0$, and t_{max} is equal to 1.0.

4.1.4 Propose algorithms to detect rule anomalies based on probability (Step 4)

This section is the last step of PHASE I; it constructs a tree structure based on the firewall rules adapted from the Path Selection Tree (PST) [2] to detect anomalies. Algorithms 1 and 2 begin by first constructing a root node of the PST. After that, the field S_IP of r_1 in Table 5 is established as the first node in the tree structure, namely S_IP_1 as shown in Figure 11 (a). In S_IP_1 node, source IP$_{v4}$ addresses of r_1 are recorded in this node by $S\_IP_1 \in [3232235776, 3232236031]$. Next node ($S\_Port_1$) stores source ports of r_1 ranging from 0 to 65535. Next, the node is recorded by destination IP$_{v4}$ addresses ranging from 0 to 4294967295, called D_IP_1. The node, named D_Port_1, contains a group of destination ports between 80 and 85 ($r_1 : [80, 85]$). The final node of PST is Pro_1, which keeps all protocols of r_1 ranging from 0 to 255 ($r_1 : [0, 255]$). In the action node, a bottom rectangular box in the tree contains an acceptance decision ($r_1 : \{1\}$) of r_1, and it also records the name of the rule on this path as $< r_1 >$.

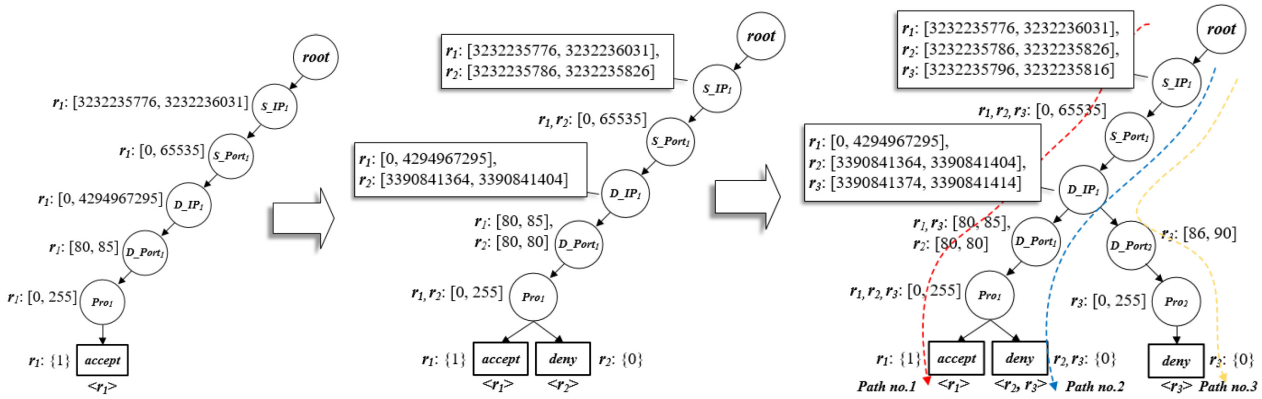
Algorithm 1 : The Path Seclection Tree (PST)

Input: Rules $\{r_1, r_2, ..., r_n\}$, where $n \in \mathbb{Z}^+, \ltimes \neq 0$
Output: the Path Selection Tree (PST)
if PST = NULL **then**
  # Create the root node and first path of PST tree
  $r_1 \leftarrow$ Read firewall Rules
  node* root $\leftarrow$ new node(NULL), node* head $\leftarrow$ root
  while $f \leftarrow$ Read each field from r_1 until NULL **do**
    head.child = new node(f)
    head = head.child
  end while *# The first path was successfully created*
  while *rule* $\leftarrow$ Read each rule from Rules **do**
    head $\leftarrow$ root.child
    $f \leftarrow$ Read first field from *rule*
    Call Subtree(head, f, *rule*)
  end while
end if
Return PST
End

Algorithm 2 : Subtree

Input: head, *field*, *rule*
Output: sub-tree
 # First case: $f \not\subset$ all fields in flat level
if head.data - f == $\emptyset$ **and** head.sibling == NULL **then**
  head.sibling $\leftarrow$ new node(f)
  head $\leftarrow$ head.sibling
  while $f \leftarrow$ Read each field from *rule* **do**
    head.child $\leftarrow$ new node(f)
    head $\leftarrow$ head.child
  end while
  Return
else
  if head.data - f == $\emptyset$ **and** head.sibling $\neq$ NULL **then**
    head $\leftarrow$ head.sibling
    Call Subtree(head, f, *rule*)
  end if
  Return
end if *# Finished first case*
 # Second case: f == all fields in deep level
if head.data - f == 0 **and** head.child == NULL **then**
  Return
else
  if head.data - f == 0 **and** head.child $\neq$ NULL **then**
    head $\leftarrow$ head.child
    $f \leftarrow$ head.data
    Call Subtree(head, f, *rule*)
  end if
end if
 # Third case: $f \subset$ of all fields in flat and deep level
if head.data - $f \neq \emptyset$ **then**
  $d \leftarrow$ head.data - f
  head $\leftarrow$ head.sibling
  Call Subtree(head, d, *rule*)
  head $\leftarrow$ head.child
  Call Subtree(head, f, *rule*)
end if
End

The next rule (r_2) is imported into the PST as illustrated in Figure 11(b). $r_2(S\_IP)$ is a subset of $r_1(S\_IP)$. Therefore, r_2 uses the same route as r_1 and also records that r_2 is equal to $[3232235786, 3232235826]$ in S_IP_1. Likewise, $r_2(S\_Port) = r_1(S\_Port)$, thus it is also recorded to



((a)) After inserting r_1. ((b)) After inserting r_2. ((c)) After inserting r_3.

Fig.11: Inserting rules r_1(a), r_2(b), and r_3(c) into PST.

S_Port_1 node ($r_2 : [0, 65535]$), and travels over the same route as r_1. r_2 (D_IP) is a subset of r_1 (D_IP). Hence D_IP of r_2 ($r_2 : [3390841364, 3390841404]$) is appended into D_IP_1 as well. In the case of $r_2(D\_Port)$, $r_1(D\_Port)$ is a superset of $r_2(D\_Port)$, so the data of D_Port_1 node is updated to be $r_1 : [80, 85], r_2 : [80, 80]$ respectively. In the last field of r_2, the action of r_1 and r_2 are not the same, so the action path needs to be separated into two paths. The first path belongs to r_1, which is chosen to *accept* ($r_1 : \{1\}$), and the second route belongs to r_2 where the action is *deny* ($r_2 : \{0\}$).

For inserting r_3 (Figure 11(c)) into the PST, there is not much different from inserting r_2. It is slightly different in the position of the destination port level in the tree. Since $r_3(D\_Port)$ is a superset of $r_1(D\_Port)$ and $r_2(D\_Port)$, some destination ports of r_3 have to be separated into another node, namely D_Port_2, which contains destination ports ranging from 86 to 90 ($r_3(D\_Port) - r_1(D\_Port)$) like $r_3 : [86, 90]$. Note that some destination ports of r_3 are on both path no.1 (Leftmost) and path no.2. The destination ports of r_3 on path no.1 are combined with r_1 and r_2 on D_Port_1 node as $r_1, r_3 : [80, 85], r_2 : [80, 80]$. The action of r_3 is not allowed in both paths (Path no.2 and no.3), where $r_3 : \{0\}$. The rest of the rules (r_4, r_5, r_6) in Table 5 are processed similarly to the rules described earlier, using algorithms 1 and 2. When all the rules have been successfully added to PST tree, the results are illustrated in Figure 12.

After creating firewall rules on the tree, the next step is to calculate each rule's probability values in the tree. Given S as a sample space equal to 1.0, it is equally divided into four parts(x_1, x_2, x_2, x_4), each equal to 25%(0.25), as shown in Figure 13. The percentage of each part can be determined by each organization's policy. For example, if the weighting of the rule-creating approver is 50% ($x_2 = 0.5$), the protocol priority is 30% ($x_4 = 0.3$), then the rest are evenly divided ($x_1 = 0.1$ and $x_3 = 0.1$). Therefore, the weights are equal to 0.1:0.5:0.1:0.3, respectively. In this paper, the experiment was carried out by dividing the sample space S into four equal parts as shown in Figure 13, resulting in the ratio of the weights to be $0.25(x_1)$:$0.25(x_2)$:$0.25(x_3)$:$0.25(x_4)$. We use the weight ratio (0.25) to re-adjust the target data ranges (t_{min}, and t_{max}) of the x_1, x_2, x_3, and x_4 fields of the firewall rule in Table 4 from [0, 1] to [0, 0.25], and normalize them again with Equation 9, which results in the new weights for each field which are shown in Table 6. When $x_1, ..., x_4$ are already calculated (As shown in Table 6), substitute these weights into Equations 20 and 21 to calculate the probability values for each rule on the tree.

Table 6: New x_1−x_4 weights of r_i after normalizing from [0, 1] to [0, 0.25].

r_i	Matching(x_1) 25%(0.25)	Evidence(x_2) 25%(0.25)	Rule maker(x_3) 25%(0.25)	Priority(x_4) 25%(0.25)
r_1	0.189	0.083	0.167	0.094
r_2	0.094	0.250	0.167	0.188
r_3	0.149	0.167	0.083	0.031
r_4	0.250	0.083	0.167	0.125
r_5	0.064	0.250	0.000	0.219
r_6	0.000	0.000	0.250	0.000

Given $P(x_1, x_2, x_3, x_4) = 0.25$, and $P(r_i|x_k)$ in Table 6 such as $P(r_1|x_1) = 0.189, P(r_1|x_2) = 0.083$, $P(r_4|x_3) = 0.167$, etc., substituting these data into Equation 20 generates the total probability values of all rules.

$$\begin{aligned} P(r_i) = {} & P(x_1)P(r_1|x_1) + P(x_2)P(r_1|x_2) + \\ & P(x_3)P(r_1|x_3) + P(x_4)P(r_1|x_4) + \\ & P(x_1)P(r_2|x_1) + P(x_2)P(r_2|x_2) + \\ & P(x_3)P(r_2|x_3) + P(x_4)P(r_2|x_4) + \\ & P(x_1)P(r_6|x_1) + P(x_2)P(r_6|x_2) + \\ & ... + P(x_3)P(r_6|x_3) + P(x_4)P(r_6|x_4) \end{aligned}$$

$$\begin{aligned} P(r_i) = {} & (0.25 * 0.189) + (0.25 * 0.083) + (0.25 * 0.167) + \\ & (0.25 * 0.094) + (0.25 * 0.094) + (0.25 * 0.250) + ... + \\ & (0.25 * 0.000) + (0.25 * 0.250) + (0.25 * 0.000) = \mathbf{0.767} \end{aligned}$$

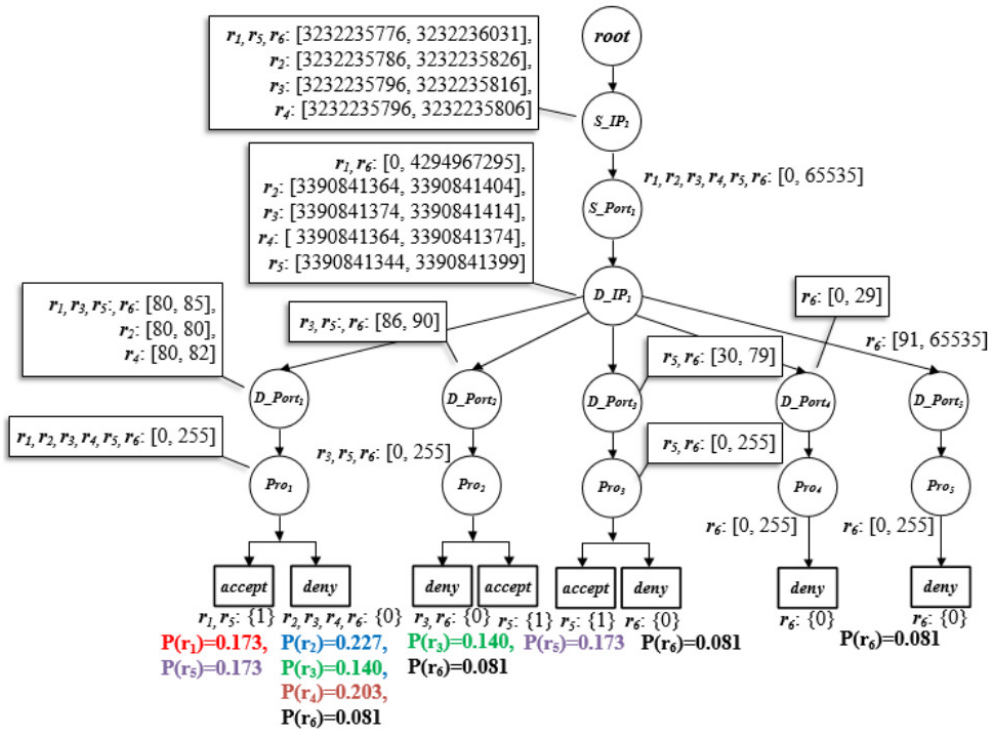


Fig.12: Complete PST tree structure after inserting all rules.

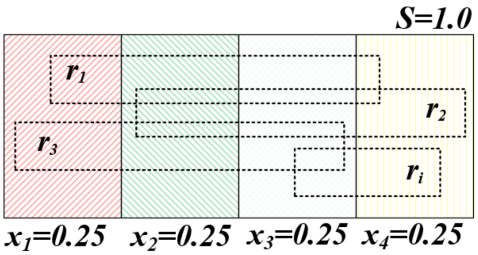


Fig.13: Divide S into four special fields ($x_1 - x_4$).

The next step is to substitute $P(r_i)$ into Equation 21 to calculate each rule's probability.

The probability of r_1 is:

$$P(x_1|r_1) = \frac{P(x_1)P(r_1|x_1)}{P(r_i)} = \frac{0.25 * 0.189}{0.767} = 0.062$$
$$P(x_2|r_1) = \frac{P(x_2)P(r_1|x_2)}{P(r_i)} = \frac{0.25 * 0.083}{0.767} = 0.027$$
$$P(x_3|r_1) = \frac{P(x_3)P(r_1|x_3)}{P(r_i)} = \frac{0.25 * 0.167}{0.767} = 0.054$$
$$P(x_4|r_1) = \frac{P(x_4)P(r_1|x_4)}{P(r_i)} = \frac{0.25 * 0.094}{0.767} = 0.031$$
$$P(r_1) = P(x_1|r_1) + P(x_2|r_1) + P(x_3|r_1) + P(x_4|r_1)$$
$$= 0.062 + 0.027 + 0.054 + 0.031 = \mathbf{0.173}$$

The probability of r_2 is:

$$P(x_1|r_2) = 0.031, P(x_2|r_2) = 0.081, P(x_3|r_2) = 0.054,$$
$$P(x_4|r_2) = 0.061$$
$$P(r_2) = P(x_1|r_2) + P(x_2|r_2) + P(x_3|r_2) + P(x_4|r_2)$$
$$= 0.031 + 0.081 + 0.054 + 0.061 = \mathbf{0.227}$$

The probability of r_3 is:

$$P(x_1|r_3) = 0.049, P(x_2|r_3) = 0.054, P(x_3|r_3) = 0.027,$$
$$P(x_4|r_3) = 0.010$$
$$P(r_3) = P(x_1|r_3) + P(x_2|r_3) + P(x_3|r_3) + P(x_4|r_3)$$
$$= 0.049 + 0.054 + 0.027 + 0.010 = \mathbf{0.140}$$

The probability of r_4 is:

$$P(x_1|r_4) = 0.081, P(x_2|r_4) = 0.027, P(x_3|r_4) = 0.054,$$
$$P(x_4|r_4) = 0.041$$
$$P(r_4) = P(x_1|r_4) + P(x_2|r_4) + P(x_3|r_4) + P(x_4|r_4)$$
$$= 0.081 + 0.027 + 0.054 + 0.041 = \mathbf{0.203}$$

The probability of r_5 is:

$$P(x_1|r_5) = 0.021, P(x_2|r_5) = 0.081, P(x_3|r_5) = 0.000,$$
$$P(x_4|r_5) = 0.071$$
$$P(r_5) = P(x_1|r_5) + P(x_2|r_5) + P(x_3|r_5) + P(x_4|r_5)$$
$$= 0.021 + 0.081 + 0.000 + 0.071 = \mathbf{0.173}$$

Finally, the probability of r_6 is:

$$P(x_1|r_6) = 0.000, P(x_2|r_6) = 0.000, P(x_3|r_6) = 0.081, P(x_4|r_6) = 0.000$$
$$P(r_6) = P(x_1|r_6) + P(x_2|r_6) + P(x_3|r_6) + P(x_4|r_6) = 0.000 + 0.000 + 0.081 + 0.000 = \mathbf{0.081}$$

The probabilities for each firewall rule that have been wholly calculated are shown in Figure 12. For the process of detecting rule anomalies, we use algorithm 3 to show each pair of anomaly rules in the tree. The results of the algorithm shown next.

For each pair of anomaly rules over path no.1 (D_Port_1): (r_1, r_5),(r_2, r_3),(r_2, r_4),(r_2, r_6) = Redundancy, (r_1, r_2),(r_1, r_4),(r_5, r_2),(r_5, r_3),(r_5, r_4) = Shadowing, (r_1, r_3) = Correlation, and (r_1, r_6),(r_5, r_6) = Generalization.

For each pair of anomaly rules over path no.2 (D_Port_2): (r_3, r_6) = Redundancy, (r_6, r_5) = Shadowing, and (r_3, r_5) = Generalization.

For each pair of anomaly rules over path no.3 (D_Port_3): Generalization=(r_5, r_6).

There are no rule anomalies over paths no.4 and no.5. The semantic loss arises from the merged rules. Thus, testing for such anomaly can be done by passing the rule pair to be tested to algorithm 4. The answer given by the algorithm is neither "No" nor "Possible" since it is challenging to detect such an anomaly.

Algorithm 3 : Detecting rule anomalies

1: **Input**: PST tree
2: **Output**: Each pair of the anomaly rules list
3: **if** PST not NULL **then**
4: *# User Depth-first search (Pre-order traversal)*
5: **while** Read path$_i$ from PST until NULL **do**
6: $p_i \leftarrow$ path$_i$ from PST
7: i = i + 1
8: **end while** *# Finish traversal all routes*
9: **while** $C_x, A_x \leftarrow p_i(C_x, A_x)$ and $C_y, A_y \leftarrow p_{i+1}(C_y, A_y)$ **do**
10: **if** $(C_x \subseteq C_y) \wedge \neg(A_x \Leftrightarrow A_y) \wedge (x \neq y)$ **then**
11: **Print** r_x is shadowed by r_y over p$_i$ and p$_{i+1}$
12: **else**
13: **if** $(C_x \cap C_y \neq \emptyset) \wedge C_x - C_y \neq \emptyset) \wedge (C_y - C_x \neq \emptyset) \wedge \neg(A_x \Leftrightarrow A_y) \wedge (x \neq y)$ **then**
14: **Print** r_x is correlation against r_y over p$_i$ and p$_{i+1}$
15: **end if**
16: **else**
17: **if** $(C_x \supset C_y) \wedge \neg(A_x \Leftrightarrow A_y) \wedge (x \neq y)$ **then**
18: **Print** r_x is generalized by r_y over p$_i$ and p$_{i+1}$
19: **end if**
20: **else**
21: **if** $(C_x \supseteq C_y) \wedge (A_x \Leftrightarrow A_y) \wedge (x \neq y)$ **then**
22: **Print** r_x is redundant against r_y over p$_i$ and p$_{i+1}$
23: **end if**
24: **else**
25: **if** $pack_i \not\subset (C_x) \wedge pack_i \not\subset (C_y) \wedge (x \neq y)$ **then**
26: **Print** Irrelevance anomaly over p$_i$ and p$_{i+1}$
27: **end if** *# where $pack_i$ = any packet*
28: **end if**
29: **end while**
30: **end if**
31: **End**

Algorithm 4 : Detecting semantic anomalies

1: **Input**: r_x and r_y $(x \neq y)$
2: **Output**: Result $\in$ {No, Possible}
3: **if** $r_{x_m} \oplus r_{y_m} \wedge (x \neq y) \Rightarrow r_{x_m} \vee r_{y_m} \vee r_{x_m+y_m} \vee r_{z_m}$ **then**
4: **Print** May be Semantics loss between r_x and r_y ***("Possible")***
5: **Return**
6: **end if**
7: **Print** No Semantics loss between r_x and r_y ***("No")***
8: **End**

4.1.5 Propose the solutions to solve rule anomalies based on probability (Step 5)

Rule anomalies in firewalls have several different solutions. For example, redundancy is solved by merging them. However, this approach may instead lead to the occurrence of semantic loss anomaly. Other anomalies, such as shadowing, correlation, and generalization, should not be merged because their decisions are different. In some cases, the administrators solve the problems by swapping the order of the rules. The consequences, such as vulnerabilities, conflicts, etc., will depend on the administrator's skills and expertise. Therefore, to encourage administrators to confidently fix anomalous rules, this research has taken the probability of each rule to analyze such anomalies to improve efficiency in solving the problems. For example, path no.1 in Figure 12 (Leftmost), r_1 and r_5 are redundant. When administrators have to merge these two rules, it can lead to a semantic loss anomaly. Thus, admins should merge a rule with a low probability value to a higher probability rule. The remaining fields are processed as explained in the following paragraphs.

Rule merging in scenario 1: From Figure 12, r_1 and r_5 (Path no.1) have the same probability ($r_1 = r_5 = 0.173$), so they can be merged to a new rule knowing r_1 is more significant than r_5. If both rules have the same probability, choose the rule that was created first as more significant.

$$r_1 \oplus r_5 = r_1(S\_IP) \cup r_5(S\_IP) \wedge r_1(S\_Port) \cup r_5(S\_Port) \wedge (r_1(D\_IP) \cup r_5(D\_IP) \wedge r_1(D\_Port) \cup r_5(D\_Port) \wedge r_1(Pro) \cup r_5(Pro) \wedge r_1(x_1) + r_5(x_1) \wedge r_1(x_2) \wedge r_1(x_3) \wedge r_1(x_4) \Rightarrow r_1(accept)$$

$$r_1 \oplus r_5 = [3232235776, 3232236031] \cup [3232235776, 3232236031] \wedge ([0, 65535] \cup [0, 65535] \wedge [0, 4294967295] \cup [3390841344, 3390841599] \wedge [80, 85] \cup [30, 90] \wedge [0, 255] \cup [0, 255] \wedge (2553 + 1212) \wedge r_1(1) \wedge r_1(2) \wedge r_1(6) \Rightarrow r_1(1) = [3232235776, 3232236031] \wedge [0, 65535] \wedge [0, 4294967295] \wedge [30, 90] \wedge [0, 255] \wedge (6318) \wedge (1) \wedge (2) \wedge (6) \Rightarrow 1$$

Where $\oplus$ is the merging operation, and *accept* is equal to 1. Note that the number of packets matching is cal-

culated from the sum of $r_1(x_1)$ and $r_5(x_1)$, for x_2, x_3, and x_4 taken from a higher probability rule.

Rule merging in scenario 2: In Figure 12, r_2 and r_3 (Path no.1) have different probabilities ($r_2(0.227) > r_3(0.140)$), so they can be merged to a new rule, where r_2 is more critical than r_3.

$$\begin{aligned} r_2 \oplus r_3 = r_2(S\_IP) \cup r_3(S\_IP) \wedge r_2(S\_Port) \cup \\ r_3(S\_Port) \wedge r_2(D\_IP) \cup r_3(D\_IP) \wedge r_2(D\_Port) \cup \\ r_3(D\_Port) \wedge r_2(Pro) \cup r_3(Pro) \wedge \\ r_2(x_1) + r_3(x_1) \wedge r_2(x_2) \\ r_2(x_3) \wedge r_2(x_4) \Rightarrow r_2(deny) \end{aligned}$$

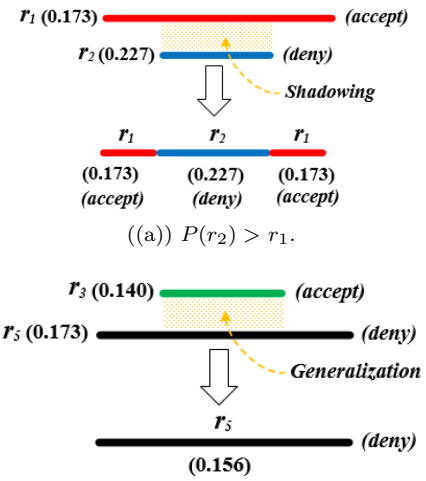
$$\begin{aligned} r_2 \oplus r_3 = [3232235786, 3232235826] \cup [3232235796, \\ 3232235816] \wedge [0, 65535] \cup [0, 65535] \wedge \\ [3390841364, 3390841404] \cup [3390841374, \\ 3390841414] \wedge [80, 80] \cup [80, 90] \wedge [0, 255] \cup \\ [0, 255] \wedge (1536 + 2127) \wedge r_2(3) \wedge r_2(2) \wedge \\ r_2(7) \Rightarrow r_2(0) \\ = [3232235786, 3390841404] \wedge [0, 65535] \wedge \\ [3390841364, 3390841414] \wedge [80, 90] \wedge [0, 255] \wedge \\ (3663) \wedge (3) \wedge (2) \wedge (7) \Rightarrow 0 \end{aligned}$$

Note that the protocol priority must always be reversed, such as 1 to 9, 3 to 7, and 9 to 1, etc., and *deny* is equal to 0.

Rule swapping: switching the rule order of r_1 and r_3 should not be done because r_1 already has a greater probability value than r_3 ($r_1 = 0.173, r_3 = 0.140$). Besides, administrators can swap the order of r_1 and r_4 rules because r_4 (0.203) has a higher probability, but the result of switching rules may lead to other anomalies.

Rule removing: administrators can immediately remove r_3 from the set of firewall rules because it is duplicated by r_2, and r_3 (0.140) also has lower probability than r_2 (0.227).

Rule editing: this approach is applied to rules that make conflicting decisions, such as Shadowing, Correlation, and Generalization. For example, in Figure 12, if administrators need to edit r_1 and r_2 (Shadowing), then r_2 is the primary choice of rule of editing because it has a greater probability. The edited rules may appear as more than two rules, and some information of r_1 will be combined with r_2, as shown in Figure 14(a). On the other hand, r_3 and r_5 are the generalization anomaly, where r_5 has a greater probability than r_3. Thus, r_3 is fully combined into r_5 by using the merging operation as shown in Figure 14(b). As a result, the tree structure and all probabilities have to be fully recalculated.



((a)) $P(r_2) > r_1$.

((b)) $P(r_5) > r_3$.

Fig.14: Editing the shadowing(a) and generalization(b).

4.2 PHASE II: Design and develop the full stack firewall framework for decision making

4.2.1 Design and develop the core firewall at the system level (Step 6)

This phase explains how to design and develop a full-stack firewall framework to support our designed firewall's functionality, called DSSF (Decision Support System for Firewall). Most of the firewalls used today do not have the features that can support our proposed firewall. For example, they do not support special firewall fields such as the number of matched packets (x_1), evidence of created rules (x_2), rule-maker expertise (x_3), and protocol priority (x_4). Moreover, they are also unable to save such fields to the Logfiles. For the reasons mentioned above, we need to build a whole system firewall starting from the low level (System call) to the high level (Graphical User Interface: GUI). The system call controls the incoming and outgoing packets when the firewall is connected to networks. A GUI was developed to allow administrators to control firewall operations easily. The overview of the designed firewall framework is shown in Figure 15. Two network cards are mounted on a computer motherboard in the physical layer to separate incoming and outgoing packets for hardware-level security. Incoming packets are forwarded from a network card to the operating system level through a system call by Netfilter Kernel Module (NKM). This module serves to forward packets or drop them off. Passing packets, forwarding them from NKM to the upper layer, is handled by the Linux kernel. Therefore, we need to enable the IP forwarding feature of the Linux kernel by issuing the command "sysctl - w net.ipv4.ip forward = 1" or putting the command "net.ipv4.ip forward = 1" into a file configuration named /etc/sysctl.conf, then restarting

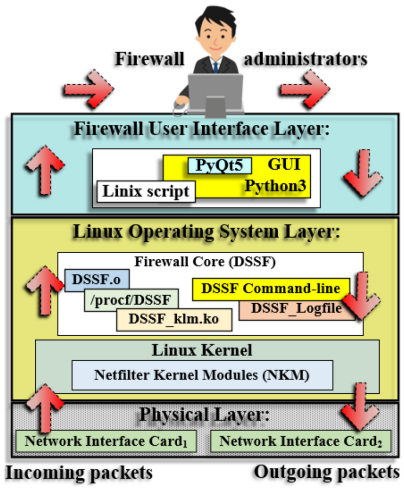


Fig.15: Full-stack firewall framework.

the computer. In the operating system layer, we have developed a firewall to control incoming and outgoing packets using the C/C++ language (GCC 4.4.7), and GNU Make 3.8 on 64-bit Linux kernel version 2.6, and we called it as DSSF. DSSF consists of two parts: the system call of user space (DSSF.o) and the system call of kernel space (DSSF_klm.ko). DSSF.o parses the syntax of commands from administrators to the kernel space, such as inserting, editing, and removing rules. If the syntax of firewall rules is correct, they are immediately passed to the kernel space by using a file system named /procf. DSSF_klm.ko executes the rules passed by administrators from the /procf file by using the procf_read() and procf_write() function. The main functions of DSSF_klm.ko are packet forwarding, dropping, and capturing based on the built-in filtering functions of NKM, as illustrated in Figure 16. Unlike traditional firewalls, there is no decision-maker (DSSF_Core).

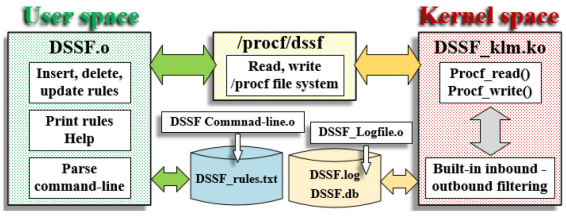


Fig.16: DSSF system call to Linux Kernel.

4.2.2 Implement the system software to record special fields, command-line, and rule patterns (Step 7)

In the operating system layer, we have implemented system software called DSSF_Logfile that records the number of real-time packets matching against any rules (x_1) to a logfile (DSSF.log). All control and execution of the firewall is handled by the command-line interface, namely the DSSF command-line, whose syntax is shown in Table 7. The software fetches and interprets the commands from administrators. The commands can be executed by the graphic user interface (GUI) or line-by-line typing (Command-line) from the keyboard. DSSF supports creating, deleting, modifying, displaying rules, and executing various firewall commands. The examples of using the DSSF command-line are described in Table 8.

Table 7: The DSSF command-line syntax.

Command	Description	Example of how to use
dssf	Firewall name	dssf
–in, –out	Inbound, outbound interface	–in –out
–srcip	Source IP address	–srcip 192.168.1.0 –srcip any
–mask	Subnet mark of IP address	–mask 255.255.255.0
–destip	Detination IP address	–destip 200.0.0.0 –destip any
–srcport	Source Port	–srcport 1234 –srcport 100-200 –srcport 1234, 1350
–destport	Destination Port	–destport any –destport 443, 8080
–proto	Protocol	–proto tcp –proto udp –proto all
–action	Decision of rule	–action accept –action deny
–delete	Delete any rule	–delete 5
–print	Print all rules	–print
–help	Help	–help
–E, –S, –P	Define the special field E = Evidence (0 - 3) S = Admin skill (0 - 3) P = Protocal priority (1 - 9)	–E 2 –S 1 –P 6

Table 8: The command-line interface of Decision Support System for Firewall (DSSF).

Examples of command-line and their description
./dssf –out –srcip 192.168.1.0 –mask 255.255.255.0 –destip any –destport 80 –E 1 –S 2 –P 3 –action accept **Meaning:** to allow network 192.168.1.0 (256 IP) to use HTTP protocol by E = 2, S = 3, P = 3 at outbound interface
./dssf –in –destip 172.16.0.0 –mask 255.255.0.0 –destip any –destport 8080, 1234 –E 3 –S 0 –P 4 –action deny **Meaning:** to drop network 172.16.0.0 (65536 IP) to access servers that are open port 8080, 1234 by E = 3, S = 0, P = 4 at inbound interface
./dssf –in –srcip 10.10.10.10 –mask 255.255.255.255 –srcport 443 –proto udp –action accept **Meaning:** to allow a src IP address 10.10.10.10, a src port 443 to all dest IP addresses, UDP protocol, E = 0, S = 0, P = 0 at inbound interface
./dssf –out –E 1 –S 3 –P 8 –action deny **Meaning:** to drop all source IP addresses, all ports, all destinations, E = 1, S = 3, P = 8 at the outbound interface

Remarks: **E** = the evidence of creating rules, **S** = the rule-making expertise, **P** = the protocol priority, **in** = inbound interface, and **out** = outbound interface

The first example rule of Table 8, "./dssf –out –srcip 192.168.1.0 –mask 255.255.255.0 –destip any –destport 80 –E 1 –S 2 –P 3 –action accept", means to allow (*accept*) the network 192.168.1.0 (256 IP addresses) to any web servers (HTTP = port 80), and it is active over the outbound interface (–out). This rule contains the special fields: an administrator is a rule creator (–E=1=x_2), rule creator expertise is standard (–S=2=x_3), and the protocol is real-time gaming (–P=3=x_4). All created firewall rules are stored in a text file named DSSF_rules.txt. Finally, the rules are already built into the tree structure and ready to be processed. They are stored in a file named DSSF.db as shown in Figure 16.

4.2.3 Implement the Graphics User Interface for Firewall (GUIF) (Step 8)

At the top layer of the full-stack firewall framework (Figure 15) is software that allows administrators to control and display various firewall information via the graphical user interface (GUI) to facilitate the use of the firewall. This software is developed in Python version 3.8 in conjunction with PyQT5 running on Linux kernel version 2.4 on a 64-bit architecture, called GUIF. Python [22] is used to directly pass administrative commands to the shell script of the Linux operating system which in turn calls the firewall command-line and passes commands from GUIF to the core firewall (DSSF.o). GUIF developed from PyQT5 [23] permits easy control and executes the firewall commands. PyQT5 is designed with a complete set of tools that can generate graphics quickly and efficiently. The main program of GUIF is illustrated in Figure 17.

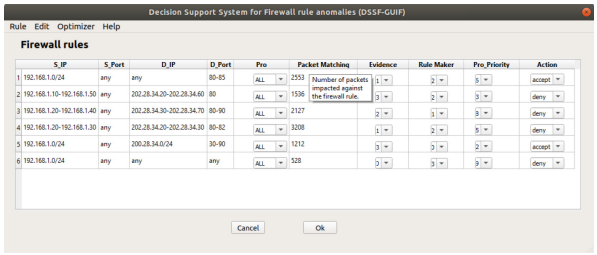


Fig.17: Main program of GUIF.

4.2.4 Test the overall functionality of the firewall framework (Step 9)

Install the firewall on the real-world network, and test the network connection (Step 9.1): DSSF has been tested on the real-world network at the Faculty of Informatics, Mahasarakham University (MSU), Thailand. Network connection starts from a private network, the network 192.168.1.0/24 (256 IP addresses) connecting our proposed firewall over the inbound interface. The outbound interface connects to the Faculty of Informatics' router gateway over 1 Gigabit per second bandwidth (Gbps). Next, our faculty router links to the computer center router's gateway. Finally, the computer center router's gateway connects to the Internet network via two service providers: UNINET and CAT (Internet Service Provider: ISP), as shown in Figure 18.

After successfully connecting the firewall to the network, we tested the firewall's functionality using the Ping (ICMP) protocol from a client inside the 192.168.1.0 network to any servers (Google.com). The experimental results are shown in Figure 19.

Test to detect, analyze, and solve rule anomalies (Step 9.2): This is the last step of PHASE II, which tests our proposed firewall functions such as anomaly rule detection, analysis of the overall rule anomalies, and probability-based resolution of conflicting rules. The various firewall functions are based on the methods introduced in PHASE I: rule detection and analysis are performed using the method in step 4. Merging, deleting, swapping, removing, and editing of rules use the methods contributed in step 5. An example, using GUIF to manage the rules as mentioned above is shown in Figure 20.

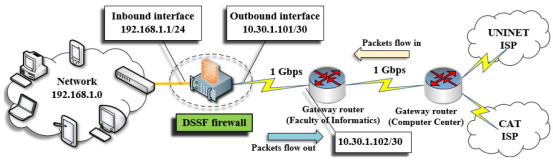


Fig.18: Connecting DSSF to the real-world networks.

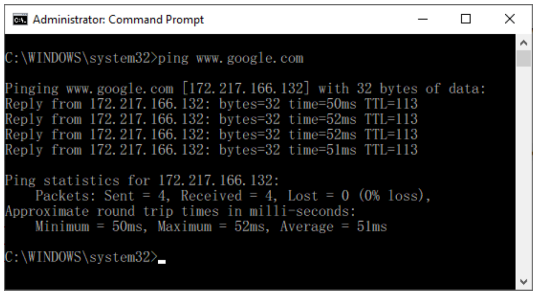
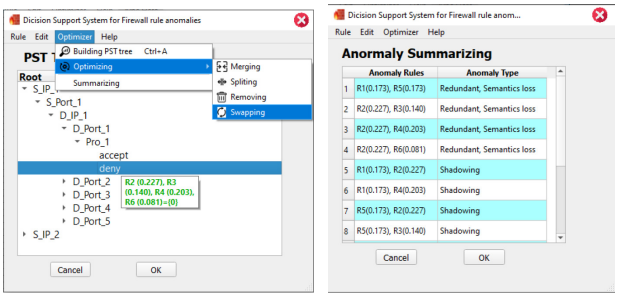


Fig.19: Testing the firewall connection to Google using Ping.



(a) Optimize rule anomalies. (b) Summarize rule anomalies.

Fig.20: Managing the firewall rules anomalies by GUIF.

4.3 PHASE III: Experimental results and performance evaluation (Step 10)

Evaluate the memory usage and time complexity: the PST is developed from an m-ary tree structure. Therefore, the processing speed of constructing the tree is $O(n)$, where n is the number of nodes over the tree. Besides, the m-ary tree can also be stored in breadth-first order as an implicit pointer-based data structure. Each node has an internal array for storing

pointers to each of its m children. Thus, the space usage of the m-ary tree structure is $O(m*n)$. Traversing the m-ary tree is very similar to binary tree traversal as the time complexity is also $O(log_m n)$.

Evaluate the reliability between assessors: in fact, assessors in different professions often do not have the same expertise. Therefore, for the most accurate assessment results, it is necessary to evaluate the assessors' reliability using Kappa statistics [19] (Equation 15). In this research, five experts and twenty-two general administrators of firewalls are used to evaluate inter-raters' reliability. The reliability assessment result between firewall experts is 0.63 (Substantial agreement), referring to Table 9, and general administrators is 0.38 (Fair agreement). The evaluation results show that the credibility among the firewall experts is good, and that of the general administrators is fair for general firewalls. Note that general administrators refer to people who have the basic ability to maintain and configure networks and computer systems such as routers, firewalls, switching, DNS, DHCP, etc.

Evaluate the satisfaction of the firewall experts and general admins: $\overline{x}$ (Equation 14) is used to estimate the satisfaction of the firewall experts and general administrators in managing rule anomalies with traditional firewalls (Guidance-free firewall) and our proposed firewall (Guided firewall with probability). In this paper, the number of assessors for evaluating firewall usage confidence is also equal to the reliability assessment. According to Table 10, the average ($\overline{x}$) of the five firewall experts' confidence to solve rule anomalies based on their skills over the traditional firewall is 3.58, and the proposed firewall based on the probability is 4.32. The results show that the firewall, including the advisory system, can increase firewall experts' confidence by about 14.8%. Likewise, general admins' average confidence value to fix rule anomalies over the traditional firewall is 2.37 (Fair), while the decision-making system increases confidence to 4.58 (Very good). According to such confidence tests, our decision-making system based on probability increases general administrators' confidence by 44.2%.

Evaluate the accuracy of resolving rule anomalies by the confusion matrix: to evaluate the accuracy of the developed firewall, we selected five firewall specialists to evaluate the system and used the confusion matrix (Accuracy) shown in Equation (16) to analyze the results. We defined the rule anomaly situation to be used to evaluate a total of 45 scenarios. The results obtained from this evaluation show that the firewall system developed correct guides for administrators in resolving rule anomalies is 83% of the time. It proves that the model is accurate enough to serve as a support system for firewall administrators. The summary of the analysis and evaluation of the firewall performance is shown in Table 10.

Table 9: Interpretation of the reliability between Inter-raters.

Kappa statistic value	Inter-rater reliability description
0	Agreement equivalent to chance
0.1 - 0.20	Alight agreement
0.21 - 0.40	Fair agreement
0.41 - 0.60	Moderate agreement
0.61 - 0.80	Substantial agreement
0.81 - 0.99	Near perfect agreement
1	Perfect agreement

Table 10: Summary of the analysis and performance evaluation between traditional firewalls vs DSSF.

Evaluation Methods	Type of evaluated firewalls	
	General	DSSF(Proposed)
Time to construct the tree	O(n)	O(n)
The space complexity	O($m*n$)	O($m*n$)
Time complexity (Tree traversal)	O($log_m n$)	O($log_m n$)
Reliability between firewall experts by $\hat{K_F}$	0.63 (Good)	0.63 (Good)
Reliability between general admins by $\hat{K_F}$	0.38 (Fair)	0.38 (Fair)
Satisfaction of firewall experts ($\overline{x}$)	3.58 (Good)	4.32 (Good)
Satisfaction of general admins ($\overline{x}$)	2.37 (Fair)	4.58 (Very good)
The accuracy to solve rule anomalies	N/A	0.83

Remarks: **N/A** = not evaluated for this research because this feature is not available in the firewall.

5. CONCLUSION

This research has designed and developed a probability-based advisory system for solving anomalies in firewall rules, called DSSF. It starts from the hardware level, with system calls, the core firewall in the kernel space inside the Linux operating system, the firewall in user space controlled by the command-line, and a graphical user interface (GUI). DSSF has been tested on real network connections and evaluated for a wide range of users through various testing methods. The results show that the our proposed firewall is an excellent substituting for traditional firewalls in solving rule anomalies with the firewall expert satisfaction value is 4.32, and the satisfaction of the firewall administrator is 4.58. The accuracy of the system in correcting the rule anomalies is 83%.

ACKNOWLEDGEMENTS

This Research was Financially Supported By Faculty of Informatics, Mahasarakham University Grant year 2021.

References

[1] E. S. Al-Shaer and H. H. Hamed, "Modeling and Management of Firewall Policies," in*IEEE Transactions on Network and Service Management*, vol. 1, no. 1, pp. 2-10, 2004.

[2] S. Khummanee, "The semantics loss tracker of firewall rules," *in IC2IT 2018 Recent Advances*

in Information and Communication Technology, vol. 769, pp. 220–231, 2018.

[3] E. Al-Shaer, H. Hamed, R. Boutaba and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *in IEEE Journal on Selected Areas in Communications*, vol. 23, no. 10, pp. 2069-2084, 2005.

[4] A. X. Liu, "Formal Verification of Firewall Policies," *2008 IEEE International Conference on Communications*, pp. 1494-1498, 2008.

[5] S. Khummanee, A. Khumseela and S. Puangpronpitag, "Towards a new design of firewall: Anomaly elimination and fast verifying of firewall rules," *The 2013 10th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 93-98, 2103.

[6] M. Rezvani and R. Aryan, "Analyzing and resolving anomalies in firewall security policies based on propositional logic," *2009 IEEE 13th International Multitopic Conference*, pp. 1-7, 2009.

[7] H. Hu, G. Ahn and K. Kulkarni, "Detecting and Resolving Firewall Policy Anomalies," in*IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 3, pp. 318-331, 2012.

[8] P.M. Mell, K.A. Scarfone and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," *The National Institute of Standards and Technology (NIST)*. Accessed 15 Aug 2019.

[9] A. Wool,, "Architecting the Lumeta firewall analyzer," *In SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pp. 7, 2001.

[10] A. Mayer, A. Wool and E. Ziskind, "Fang: a firewall analysis engine," *Proceeding 2000 IEEE Symposium on Security and Privacy*, pp. 177-187, 2000.

[11] A. Hari, S. Suri and G. Parulkar, "Detecting and resolving packet filter conflicts," *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, pp. 1203-1212 vol.3, 2000.

[12] L. Zhang and M. Huang, "A Firewall Rules Optimized Model Based on Service-Grouping," *2015 12th Web Information System and Application Conference (WISA)*, pp. 142-146, 2015.

[13] Lihua Yuan, Hao Chen, Jianning Mai, Chen-Nee Chuah, Zhendong Su and P. Mohapatra, "FIREMAN: a toolkit for firewall modeling and analysis," *2006 IEEE Symposium on Security and Privacy*, pp. 15 pp. 208-213, 2006.

[14] A. Saâdaoui, N. B. Y. B. Souayeh and A. Bouhoula, "Formal approach for managing firewall misconfigurations," *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*, pp. 1-10, 2014.

[15] W. Krombi, M. Erradi and A. Khoumsi, "Automata-based approach to design and analyze security policies," *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pp. 306-313, 2014.

[16] A. Saâdaoui, N. B. Y. B. Souayeh and A. Bouhoula, "Automated and Optimized FDD-Based Method to Fix Firewall Misconfigurations," *2015 IEEE 14th International Symposium on Network Computing and Applications*, pp. 63-67, 2015.

[17] Aggarwal, C.C., *Neural Networks and Deep Learning*, 1 ed. Springer International Publishing, Boca Raton, FL, USA, 2018.

[18] Morris, D., Koning, M., *Bayes' Theorem Examples: A Visual Introduction For Beginners*. Blue Windmill Media, 80 Strand, London, WC2R 0RL UK, 2016.

[19] Berry, K.J., Johnston, J.E., Paul W. Mielke, J., *The Measurement of Association: A Permutation Statistical Approach*. Springer International Publishing, Springer Nature Switzerland AG, 2018.

[20] Ting, K.M., *Confusion Matrix*. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning. pp. 209-209. Springer US, Boston, MA, 2010.

[21] CelPlan Technologies, I., GPP QoS Class Identification QCI categories. `http://www.celplan.com` (2019). Accessed 12 Sep 2019.

[22] Python Software Foundation, Python 3.8.7. `https://www.python.org/downloads/release/python-387/` (2019). Accessed 10 Feb 2020.

[23] Python Software Foundation, Python bindings for the Qt cross platform application toolkit. `https://pypi.org/project/PyQt5/` (2019). Accessed 12 Aug 2020.

Suchart Khummanee received the B.Eng. degree in Computer Engineering from the King Mongkut's Institute of Technology Ladkrabang, the M.Sc. degree in Computer Science from the Khon Kaen University, and the Ph.D. degree in Computer Engineering from the Khon Kaen University, Thailand. He is currently a full lecturer of Computer Science at the Mahasarakham University, Thailand. His research interests in the network security, computer networks, agricultural robotics, and Internet of things (IoT).

Phatthanaphong Chomphuwiset is a full-time lecturer at the Department of Computer Science, Mahasarakham University, Thailand. He received a PhD. in computing from the University of Leeds, UK. His main research interests fall into a variety of areas, including medical image processing, Computer Vision applications, and Machine Learning. One of his current research projects is to investigate techniques in machine learning and deep learning to process histopathological image.

Potchara Pruksasri has obtained both B.Sc. and M.Sc. of Computer Science at Khon Kaen University Thailand in 2000 and 2005 respectively. In 2005, he has been employed as a lecturer at Mahasarakham University, Thailand. He is currently working on his Ph.D. research at Computer Science Department, Faculty of Informatics, Mahasarakham University. His research focuses on information security and access control of the supply chain information system to secure data exchange of global supply chains.