Future Generation Computer Systems 30 (2014) 116–126

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Improving cloud network security using the Tree-Rule firewall

Xiangjian He, Thawatchai Chomsiri *, Priyadarsi Nanda, Zhiyuan Tan

School of Computing and Communications, Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia

HIGHLIGHTS

- We identify five limitations of traditional Listed-Rule firewalls on large networks.
- We propose and design a Tree-Rule firewall model that has none of the five limitations.
- We examine Tree-Rule firewalls on LANs and demonstrate better performance than IPTABLES.
- We show efficient performance and benefits of Tree-Rule firewalls under a cloud environment.

ARTICLE INFO

Article history:
Received 31 December 2012
Received in revised form
3 May 2013
Accepted 28 June 2013
Available online 26 July 2013

Keywords:
Firewall
Tree-Rule firewall
Network security
Cloud security
Cloud computing

ABSTRACT

This study proposes a new model of firewall called the 'Tree-Rule Firewall', which offers various benefits and is applicable for large networks such as 'cloud' networks. The recently available firewalls (i.e., Listed-Rule firewalls) have their limitations in performing the tasks and are inapplicable for working on some networks with huge firewall rule sizes. The Listed-Rule firewall is mathematically tested in this paper to prove that the firewall potentially causes conflict rules and redundant rules and hence leads to problematic network security systems and slow functional speed. To overcome these problems, we show the design and development of Tree-Rule firewall that does not create conflict rules and redundant rules. In a Tree-Rule firewall, the rule positioning is based on a tree structure instead of traditional rule listing. To manage firewall rules, we implement a Tree-Rule firewall on the Linux platform and test it on a regular network and under a cloud environment respectively to show its performance. It is demonstrated that the Tree-Rule firewall offers better network security and functional speed than the Listed-Rule firewall. Compared to the Listed-Rule firewall, rules of the Tree-Rule firewall are easier to be created, especially on a large network such as a cloud network.

1. Introduction

Nowadays, 'cloud' as an architectural structure has been broadly employed so that users can use cloud computing with optimal benefits including fast processing and network speed, effective data distribution and low cost. However, cloud computing always encounters new problems such as problematic network security. Technically, cloud computing normally functions on the virtual system of a network connected to various organizations mostly working on the same physical network or sometimes on the same physical machine. At the point where the network security is critical, security devices, such as firewalls [1] and Intrusion Detection Systems (IDSs) [2], are applied in a cloud network.

1.1. Firewall background

There have been many studies about firewall rule conflicts (anomalies) that occur within rule sets (e.g., the rule set shown in Table 1 for a traditional Listed-Rule firewall). E-hab Al Shaer et al. [3] proposed several anomaly definitions including 'shadowing anomaly'. They defined 'Shadowed Rule' as a rule that cannot be matched by any packet. Therefore, this type of rules should be removed from the rule list without any changes to a firewall policy. Moreover, they also applied their definitions and theories for analyzing a distributed firewall [4]. The authors of [3–5] focused on mathematically analyzing firewall rules. Scott Hazelhusrt [6] used Binary Decision Diagrams (BDDs) to present and analyze rule sets. Pasi Eronen [7] proposed an Expert System that was based on Constraint Logic Programming (CLP) for users to write higher-level operations to detect common configuration mistake.

To get rid of the rule conflicts in a firewall, Lihua Yuan et al. proposed the Fireman Toolkit [8], which could help administrators to design and analyze firewall rules. However, their toolkit only mitigated some problems of traditional firewall, and the Fireman

* Corresponding author. Tel.: +61 426660669.
E-mail addresses: Xiangjian.He@uts.edu.au (X. He), Thawatchai.Chomsiri@student.uts.edu.au (T. Chomsiri), Priyadarsi.Nanda@uts.edu.au (P. Nanda), zhiyuan.tan@uts.edu.au (Z. Tan).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.06.024

Table 1
An example of rules on the Listed-Rule firewall.

No.	Protocol	Source IP	Destination IP	Dest. Port	Action
1	TCP	10.1.1.1	20.1.1.1	80	Accept
2	TCP	10.1.1.2	20.1.1.1	80	Deny
3	TCP	10.1.1.0/24	20.1.1.1	80	Deny
4	TCP	10.1.1.3	20.1.1.1	80	Accept
5	TCP	10.2.2.0/24	20.2.2.5	80	Deny
6	TCP	10.2.2.5	20.2.2.0/24	80	Deny
7	TCP	10.3.3.0/24	20.3.3.9	80	Accept
8	TCP	10.3.3.9	20.3.3.0/24	80	Deny
9	IP	0.0.0.0/0	0.0.0.0/0	0-65535	Deny

Toolkit was not a new type of firewall compared with the traditional Listed-Rule firewall (see, for example, Table 1). Liang Zhao et al. [9] proposed to use '*goto*' function inside Listed-Rule firewalls (e.g., a '*jump*' command in IPTABLES). Although their rule structure looks like a tree structure, their sub-rules (or nodes) contain Listed-Rules. Therefore, their firewalls are still deemed as Listed-Rule firewalls and are time consuming when performing linear and sequential rule searching.

Our research in this paper presents several limitations of Listed-Rule firewall and proposes a new type of firewall, which has different mechanisms. We use a tree-shape and hierarchical rule set. Although the phrase 'hierarchical rule set' [10] has appeared in the manuals of CSS (Cisco Services Switch), those rules are relevant to load-balancing devices in a network rather than on the firewall and merely describe which content (e.g., an html file) is accessible by visitors to the Web site. Alex Liu and Mohamed Gouda proposed 'Diverse Firewall Design' [11] using tree structure rules translated from a rule list to discover and eradicate some of the rule conflictions. However, their work was still based on the traditional firewall design.

1.2. Firewall on cloud environment

Dimitrios Zissis et al. [12] addressed cloud computing security and focused on cryptography. Seoksoo Kima et al. [13] proposed an enterprise security management system with reinforced internal security. However, the work presented in both [12,13] did not focus on the firewall directly.

A firewall can be implemented on cloud environment using hardware or software. If a software firewall (e.g., IPTABLES installed on Guest OS) is applied, the firewall position is as shown in Fig. 1(a) [14]. On the other hand, if a hardware firewall is applied, the firewall position differs from what is shown in Fig. 1(a) [15] and the firewall is not in the hypervisor (i.e., a piece of computer software, firmware or hardware that creates and runs virtual machines) [16] although the levels of network security remain the same.

The firewall positioning in Fig. 1(a) has its own benefits because it does not consume much resource due to only a single firewall computer required. However, this positioning still faces security problems because a virtual machine (VM) behind the firewall may be attacked by other VMs situated in the same domain. Therefore, to upgrade the security level, one proposal is to reposition the firewalls as shown in Fig. 1(b) [14]. However, this positioning consumes much more resource (e.g., disk space, RAM, and hypervisor's CPU). To resolve this problem, it is proposed that the firewalls are repositioned as shown in Fig. 1(c) [14] and this proposal consumes less resource but requests more rules in the firewall than the one shown in Fig. 1(a). The added rules are the ones for preventing the attacks between VMs. To overcome the problems, we will present our Tree-Rule firewalls to support the third model (Fig. 1(c)) so that the firewall will consume less resource, process rapidly, and show no rule conflicts.

1.3. Paper organization

The aim of this paper is to identify the limitations of Listed-Rule firewall on a large network containing a number of computers, e.g., a cloud network. In Section 2, we will present that Listed-Rule firewall causes rule conflicts, which will be increased by the number of rules. We will confirm these limitations through mathematical proofs and present our results. In Section 3, we will propose a new firewall model called the Tree-Rule firewall. We will implement and perform our newly designed the Tree-Rule firewall on the Linux platform. The experimental results will be shown in Section 4. We will test our proposed Tree-Rule firewall using a regular network and under a cloud environment to compare its performance efficiency under such scenarios. This paper will be concluded in Section 5.

2. Limitations of the Listed-Rule firewall

In this section, we will illustrate that the existing firewalls (Listed-Rule firewalls) have five critical limitations which lead to security problem, speed problem, and 'difficult to use' problem. These limitations are that

1. 'shadowed rule' (i.e., *a rule that can never be matched by any packet because the packet must have matched with other rules above*) can lead to security and speed problem;
2. swapping position between rules changes the firewall policy and hence causes a security problem;
3. 'redundant rules' can cause speed problem;
4. firewall administrators have to locate 'bigger rules' only after 'smaller rules', and this results in a 'difficult to use' problem; and
5. sequential rule searching can lead to a speed problem.

In the following, we design a model to reveal the five limitations of the Listed-Rule firewall. This newly designed model is called the '2D-Box Model' [17] for explaining a matching between packets and firewall rules as shown in Fig. 2. Suppose that there are two source IP addresses ('*a*' and '*b*'), two destination IP addresses ('*x*' and '*y*'), and two port numbers ('1' and '2') in the system. We do not include other attributes (such as source ports and protocol types) for easier understanding.

In Fig. 2, SIP stands for the set of Source IP addresses of a rule entry (corresponding to a rule order number), DIP stands for the set of Destination IP addresses, and DPT stands for the set of Destination Port numbers.

Referring to the 2D-Box Model (see Fig. 2), the incoming packets will be matched with Rule-1 (the 1st rule in the list) first. In this case, Rule-1 will 'accept' two packets. The remaining packets will continue falling down to Rule-2 that has a 'deny' action. After that, the remaining packets will continue falling down to other rules below until they reach the last rule or match with some rules.

Given a rule entry with its SIP, DIP and DPT, let

$$\begin{aligned}\text{SIP} \times \text{DIP} \times \text{DPT} = \{(i, j, k) | i = &\text{ source IP address},\\ j = &\text{ destination IP address and}\\ k = &\text{ destination port number}\}.\end{aligned} \tag{1}$$

Note that the action for either 'Accept' or 'Deny' is excluded from the above representation, where "×" is an operator for computing the 'Cartesian product' [18]. The result of the Cartesian products represented in Eq. (1) is called the *relation* corresponding to the rule entry [18]. For example (see Fig. 2), for Rule-1, we have the relation (denoted by R_1),

$$R_1 = a \times y \times \text{any} = \{(a, y, 1), (a, y, 2)\},$$

and for Rule-4, we have the relation (denoted by R_4),

$$R_4 = \text{any} \times \text{any} \times 1 = \{(a, x, 1), (a, y, 1), (b, x, 1), (b, y, 1)\}.$$

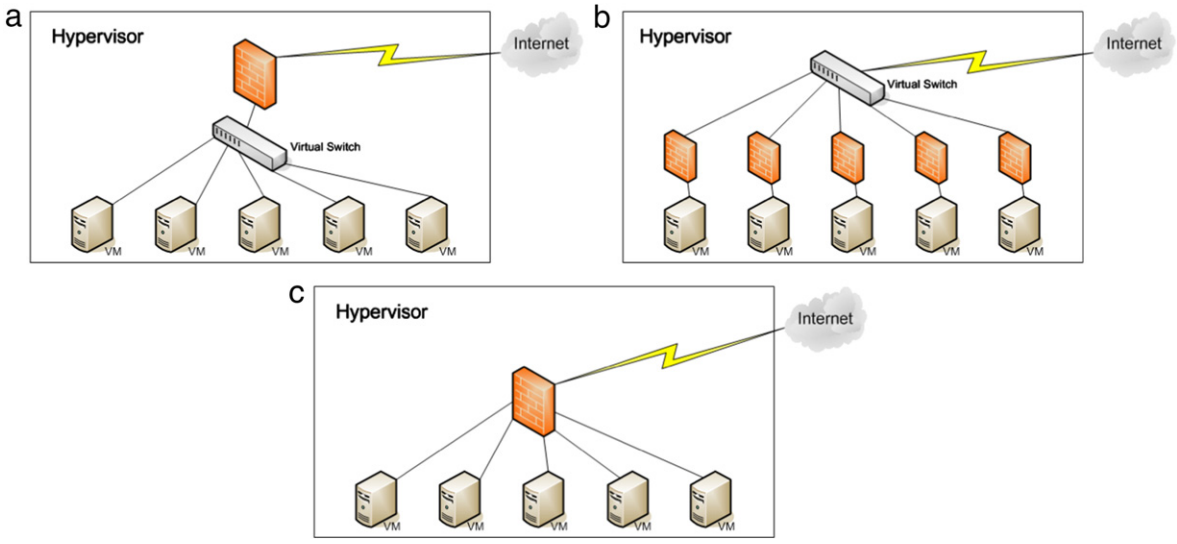

Fig. 1. Firewall models in cloud environment [14].

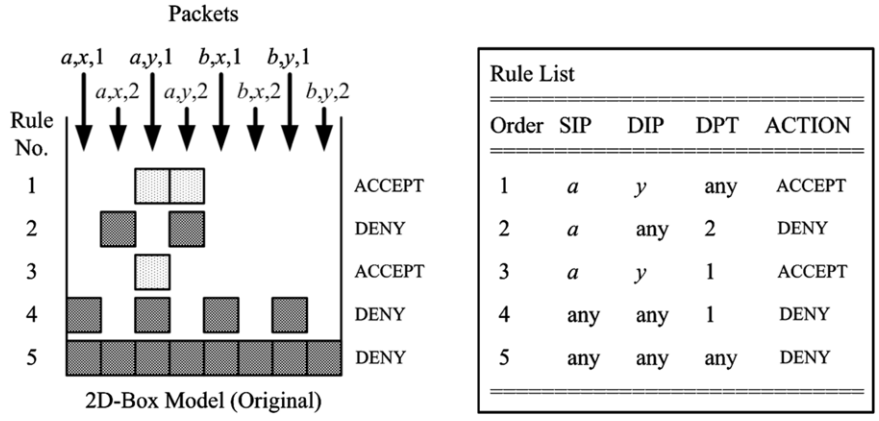

Fig. 2. The 2D-box model (left) and the rules of a Listed-Rule firewall (right).

In general, for a given rule entry order number i, let Rule-i denote Rule i, and R_i denote the relation corresponding to Rule-i (without including any action).

For example, suppose that Rule-x is:

Rule No.	Source IP	Dest IP	Dest Port	Action
x	10.1.1.1	20.2.2.0/30	80–81	Accept

then R_x is

{(10.1.1.1, 20.2.2.0, 80),
(10.1.1.1, 20.2.2.1, 80),
(10.1.1.1, 20.2.2.2, 80),
(10.1.1.1, 20.2.2.3, 80),
(10.1.1.1, 20.2.2.0, 81),
(10.1.1.1, 20.2.2.1, 81),
(10.1.1.1, 20.2.2.2, 81),
(10.1.1.1, 20.2.2.3, 81)}.

We can apply the 2D-Box Model to the firewall rules. For example, we can define a range of IP addresses = {0.0.0.0–255.255.255.255}, or a range of port numbers = {0–65535} in IPv4 as illustrated in the above examples. Moreover, the 2D-Box Model can also be extended and applied on IPv6.

2.1. Limitations on the shadowed rule

Recall that a 'shadowed rule' is a rule that can never be matched by any packet because the packet should have been matched with one of the rules before this rule. For example, Rule-4 in Table 1 is a shadowed rule because any packet that matches this rule has already matched Rule-3 in order of precedence. Another example is Rule-3 on the right side of Fig. 2. This limitation can cause security and speed problems.

Security problems are likely to be occurred, especially in an enterprise network that has a large number of rules in the firewall. For example, suppose that a new worm is sending packets to attack the network. After this attack is detected, the firewall administrator will add a new firewall rule for protection against such an attack. If this added rule is shadowed by old rules above, which allow attacking packets to go through, then the security problem is definitely occurred.

Speed problem can occur because many shadowed rules can waste the firewall processing time on these useless rules. Because most of packets will be matched with the last rule (i.e., the rule that denies all packets), the shadowed rules will have to be processed for matching the packets before the last rule. This can generate low throughput to the firewall.

In the following, we prove that shadowed rules are not necessary for firewall and can be deleted without any change to firewall policy.

Theorem 1. *If Rule-i (i.e, the i-th rule) in a Listed-Rule firewall is a shadowed rule, then we can remove Rule-i without any changes to firewall rule policy.*

Proof. Suppose that p matching a packet is an entry of the relation (i.e, a set represented by the Cartesian product defined in Eq. (1)) corresponding to Rule-i. Then, p should have also been an entry of the relation corresponding to one of the rules above Rule-i, according to the definition of a shadowed rule described at the beginning of this section. Therefore, we have

$$p \in R_{i-1} \cup R_{i-2} \cup \cdots \cup R_1.$$

This implies that

$$R_i - (R_{i-1} \cup R_{i-2} \cup \cdots \cup R_1) = \phi.$$

Therefore,

$$R_i \cup R_{i-1} \cup \cdots \cup R_1 = R_{i-1} \cup R_{i-2} \cup \cdots \cup R_1.$$

This concludes that

$$R_{i+1} - (R_i \cup R_{i-1} \cup \cdots \cup R_1) = R_{i+1} - (R_{i-1} \cup R_{i-2} \cup \cdots \cup R_1),$$

i.e., deleting Rule-i or not will not affect the rules after Rule-i and hence does not change the rule policy of the firewall. ■

2.2. Limitation about swapping position between rules

Swapping the positions of two rules on a Listed-Rule firewall can cause policy changes on the firewall if the two rules have different actions, and both of them can be matched with the same packet. For example, swapping between Rule-7 and Rule-8 (see Table 1) will change the action on the packet (with Source IP = 10.3.3.9, Destination IP = 20.3.3.9, Destination Port = 80) from being accepted to being denied.

Changing the packet action from being accepted to being denied can also be a security problem. For example, if packets that send/receive between clients and servers are blocked, it can be deemed as another security problem because of the lack of Availability (ready to use). Moreover, security problems are likely to occur if dangerous packets that must be denied are accepted because of the rule swapping. The above limitations are further described in the following theorems.

Theorem 2. *Let Rule-x and Rule-y ($y > x$) be two rule entries on a Listed-Rule firewall and have different actions, and K be*

$$R_{x-1} \cup R_{x-2} \cup \cdots \cup R_1.$$

Then swapping the positions of Rule-x and Rule-y will cause rule policy changes on the firewall if

$$(R_x \cap R_y) - K \neq \phi.$$

Proof. Note that

$$(R_x \cap R_y) - K \neq \phi.$$

Thus, we have that

$$(R_x \cap R_y) - K = (R_x - K) \cap (R_y - K) \neq \phi. \quad (2)$$

Eq. (2) indicates that there is a *relation* entry, p, that will fall into both $R_x - K$ and $R_y - K$. Let P be a packet that matches p, then before swapping the positions of Rule-x and Rule-y,

$$p \in R_x - K, \quad (3)$$

and P is taken the action defined for Rule-x. After swapping the positions of Rule-x and Rule-y, Rule-y becomes the first rule after Rule-$(x-1)$,

$$p \in R_y - K, \quad (4)$$

and P is taken the action defined for Rule-y. By the assumption of this theorem, the actions taken before and after swapping the positions of Rule-x and Rule-y (corresponding to Eqs. (3) and (4) respectively) are different, so the swapping operation changes the rule policy of the firewall. ■

2.3. Limitation about redundant rules

A redundant rule is a rule that is redundant to (or has been implied in) another rule below it with the same action. For example, Rule-8 in Table 1 is redundant to Rule-9. As another example, Rule-4 in Fig. 2 is redundant to Rule-5. If we remove a redundant rule, a firewall policy should not change. Redundant rules can cause a speed problem because many redundant rules can waste the firewall processing time. We prove below that redundant rules are not necessary and can be deleted without any change to firewall policy.

Theorem 3. *Suppose Rule-i, Rule-$(i+1)$, Rule-$(i+2)$, …, Rule-$(i+n)$ on a firewall have the same action (where 'n' is a positive integer). If*

$$R_i \subset R_{i+1} \cup R_{i+2} \cup \cdots \cup R_{i+n},$$

then removing Rule-i will not make any change of policy on the firewall.

Proof. Let

$$R_A = R_{i+1} \cup R_{i+2} \cup \cdots \cup R_{i+n}.$$

Before removing Rule-i, let

$$p \in R_i - (R_{i-1} \cup R_{i-2} \cup \cdots \cup R_1).$$

Note that $R_i \subset R_A$, so

$$R_i - (R_{i-1} \cup R_{i-2} \cup \cdots \cup R_1) \subset R_i \subset R_A.$$

Therefore, after removing Rule-i,

$$p \in R_{i+1} \cup R_{i+2} \cup \cdots \cup R_{i+n},$$

and packets that used to match p will fall down to match with Rule-$(i+1)$, Rule-$(i+2)$, …, or Rule-$(i+n)$ in order. Because all of Rule-i, Rule-$(i+1)$, …, and Rule-$(i+n)$ have the same action, there are no changes to the policy after removing Rule-i. ■

2.4. Limitation of rule design

In the rule design process of a Listed-Rule firewall, firewall administrators have to put 'bigger rules' after 'smaller rules'.

Note that

- a 'bigger rule' is a rule that can be mapped into a 'bigger relation', where
- a 'bigger relation' is a relation that is bigger than some other relations (i.e., a superset of other relations).

An example of bigger rule is the last rule in a Listed-Rule firewall (e.g., Rule-18 in Table 2), which is bigger than every other rule above. We cannot move this rule to the first position because it can shadow all other rules if we do.

As another example of bigger rule, if we want to prevent normal users from attacking the servers in a DMZ (see Fig. 3) but allow admin people to manage servers through port 22, we have to prevent access of normal users using Rule-14 (in Table 2) but allow access of admin people using Rule-1 and Rule-2. As a result, Rule-14 is a bigger rule compared with Rule-1 (and Rule-2), and we have to locate Rule-14 after Rule-1 (and Rule-2). With this limitation, it is difficult to design rules on a Listed-Rule firewall because rule positions are not independent. Moreover, in a Listed-Rule firewall, the biggest rule (e.g., Rule-18 in Table 2) cannot be moved upward to other positions. This also causes a speed problem because the packets that only match this biggest rule (i.e., the last rule) will have to go through all other rules first to find any possible matching.

2.5. Limitation from sequential computation

Rule computation for packet decision on a Listed-Rule firewall is a sequential process. Consequently, it may cause a speed problem.

Table 2
An example of rules on a medium size network.

No.	Source_IP	Dest_IP	Dest_Port	Action
1	200.1.2.99	200.1.1.3	22	Accept
2	200.1.2.99	200.1.1.4	22	Accept
3	200.1.2.*	200.1.1.2	22	Accept
4	200.1.2.*	200.1.1.5	22	Accept
5	200.1.2.*	200.1.1.3	110	Accept
6	200.1.2.*	200.1.1.3	143	Accept
7	200.1.2.*	200.1.1.5	3306	Accept
8	*	200.1.1.1	22	Accept
9	*	200.1.1.1	80	Accept
10	*	200.1.1.2	80	Accept
11	*	200.1.1.2	443	Accept
12	*	200.1.1.3	25	Accept
13	*	200.1.1.4	53	Accept
14	200.1.2.*	200.1.1.*	*	Deny
15	200.1.2.*	*	*	Accept
16	200.1.1.3	*	25	Accept
17	200.1.1.4	*	53	Accept
18	*	*	*	Deny

Especially, a firewall that has a large number of rules may work with a slow speed. The time used for rule computation would depend on the number of rules for the firewall.

Let the number of rules in a Listed-Rule firewall be N. Then, the number of rules (on average) that will be compared with a packet is $N/2$, and the time to compute each packet's matching is

$t \in O_{(N)}$.

In the next section, we will demonstrate a Tree-Rule firewall and show that this new type of firewall leads to more efficient computation.

3. Design and implementation of the Tree-Rule firewall

This section presents the design and implementation of the newly proposed Tree-Rule firewall. The basic design [17] is presented and illustrated in Section 3.1. We will analyze the benefits of the proposed firewall. In Section 3.2, we will improve the basic design and analyze the time complexity. The implementation of Tree-Rule firewall is shown in Section 3.3.

3.1. Basic design

The design of our proposed 'Tree-Rule firewall' is shown in Fig. 4. This design can avoid the five limitations of Listed-Rule firewall. In this subsection, we will explain the advantages of Tree-Rule firewall including:

- no shadowed rules,
- no need of rule swapping because all rules will be sorted automatically,
- no redundant rules,
- ease of rule design (with independent rule paths), and
- high speed for packet access decision.

The Tree-Rule firewall is a new kind of firewall in which the rules are presented in a tree form (see Fig. 4) instead of a list of lines (i.e., rules). This firewall will read attribute information from packet header and compare packet's first attribute with the data in the root nodes in the rule tree. After that, the firewall will check packet's other attributes in order by searching only on relevant nodes at the corresponding levels. As a result, the packet will be decided quickly with a specific action. For example, as shown in Fig. 4, when packets arrive at the Tree-Rule firewall, the firewall will consider Dest IP (destination IP address), Dest Port (destination port), and Source IP respectively in order until packets' access decisions are made by predefined actions.

Actually, an attribute within the root node can be Source IP, Destination Port, or any attribute suitable to work with the firewall rules. Users can select attributes that they want for each column before creating tree rules. For example, if we focus on the protection for servers inside our network, we should select Destination IP to be the root node. This is because we can easily imagine that targeted servers are important sources of information and their IP addresses are most significant to block any misused information from them. On the other hand, if we focus on permissions for users, we should specify Source IP to be the root node to allow where (destination IP addresses) they (Source IP addresses) want to go. We have created a Graphic User Interface (GUI), a rule editor, where users can specify attributes for each column easily. In this paper, we use Destination IP for the root node.

As we can see in Fig. 4, the Tree-Rule firewall has no security problem because the users do not need to swap rule positions. Also, the Tree-Rule firewall has no rule numbers. Instead, we call each path of the tree a 'Rule Path'. Data in each node will be sorted in the ascending order.

Rule designers are not necessary to have any skills. They need only basic concepts for Tree-Rule firewall design. This means that Tree-Rule firewall's rules are easy to design.

Moreover, the rules (or rule paths) that will be matched by almost all packets (such as the rule on the bottom path that shows 'Else → Else → Else → Deny') in Fig. 4 will take the same length of time to make packet access decision (Accept or Deny) as other rules (or rule paths).

3.1.1. Time complexity of the basic design

With regard to performance, the time complexity for making a packet access decision in a Listed-Rule firewall is $O(N)$, where N is the number of rules. The time complexity in a Tree-Rule firewall is in the order of $\log(N)$. For example, for the Listed-Rule firewall rules shown in Table 2, if we assume that the chances for rules to be matched by packets are equal. Then, it will take $(18/2) \times 3 \times C = 27C$ (where C is the time interval that is used for comparing between 'one attribute of packet header' and 'one attribute of rule'). On the other hand, the Tree-Rule firewall in Fig. 4 (where Dest IP has 6 lines, Dest Port has 4 lines (on average), and Source IP has 2 lines (on average)) will take a time less than $C \times \text{Log } 8 + C \times \text{Log } 4 + C \times \text{Log } 2 = C \times (3 + 2 + 1) = 6C$. Note that all 'Log' values above are of base 2.

We use an enterprise network as another example to compare the computation complexity using a Listed-Rule firewall and a Tree-Rule firewall. Assume that the enterprise network consists of approximately 100 servers, and each server opens about 20 ports and has approximately 5 groups of users requesting access to the DMZ and the Internet. Then, there are approximately $100 \times 20 \times 5 = 10{,}000$ rules on the Listed-Rule firewall on the network to control the access to the DMZs and the Internet. Therefore, on average, it will take about $(10{,}000/2) \times 3 \times C = 15{,}000C$ to make the access decision for each packet and require $10{,}000 \times 3 \times C = 30{,}000C$ in the worst case. In contrast, using a Tree-Rule firewall, it takes less than $C \times \text{Log } 128 + C \times \text{Log } 32 + C \times \text{Log } 8 = C \times (7 + 5 + 3) = 15C$ to make the access decision for any packet in the worst case.

Note that, in the above description,

128 is the number of destination IP addresses rounded up from 100,
32 is the number of destination Ports rounded up from 20, and
8 is the number of source IP addresses rounded up from 5.

As we can see, the above time complexity comparison between a Tree-Rule firewall and a Listed-Rule firewall is similar to the comparison between 'Binary Search Tree' [19] and 'Linear Search' [20] in an Array.

3.1.2. Additional benefits of the basic design

On the 'security' aspect, the Listed-Rule firewall on enterprise networks (having many rules) is likely to encounter with

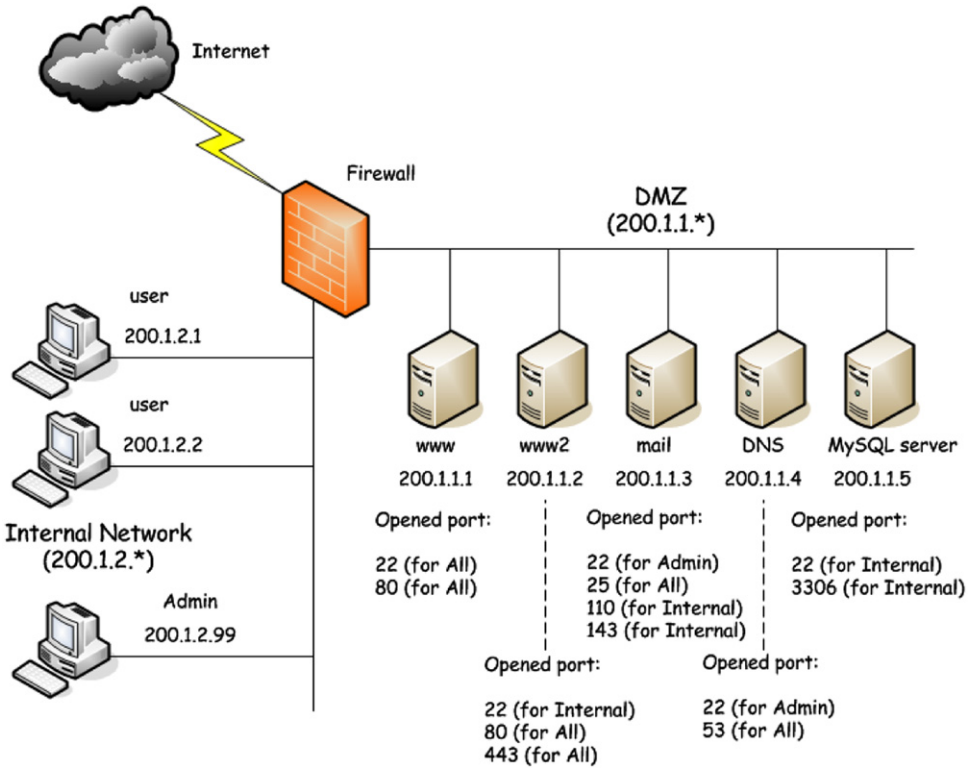

Fig. 3. A medium size network with a DMZ.

confliction of rules. Such conflictions include shadowed and redundant rules. In contrast, the Tree-Rule firewall creates no rule conflictions because the Tree-Rule firewall can never contain any shadowed rules or redundant rules.

On the 'easy to use' aspect, it is very difficult to design rules for a Listed-Rule firewall following the policy of an organization. With many lines of rules, it is difficult to check and test how each rule works. On the contrary, a Tree-Rule firewall can be designed easily because every rule sentence (rule path) of Tree-Rule firewall takes a separate path.

3.2. Improvement of basic design

In most cases, many computers need to be protected with the same policy. In the basic design of Tree-Rule firewall (see Fig. 4), the root node (column of 'Dest IP') has the number of lines equal to the number of user's computers. Each line is linked to some sub-trees with repeated (the same) data. To get rid of the above-mentioned problems and to improve the basic design, the 'Single IP Address' design (as shown in Fig. 4) is replaced by 'IP Address Range' design (see Fig. 5). In corresponding to the changes, those single destination ports (i.e., Dest Ports) shown in Fig. 4 are replaced by the port ranges shown in Fig. 5.

In Fig. 5, it can be seen that the IP Addresses between 200.1.2.1–200.1.2.254 apply the same policies that allow 200.1.1.1–200.1.1.254 to remotely access to port number 22. Using the improved design, 253 lines can be saved on the root node, and the memory spaces that were previously requested in the basic design for the 253 sets of the repeated data (sub-trees) can now be saved.

3.2.1. Time complexity of improved design

We have shown that the improved design can save memory spaces. We now show that the rule searching time increases only a bit because of the change from a single number design to a range design. It is found that the searching time within the node slightly increased as shown below.

$t \in O(\log_2 N)$ is changed to $t \in O(1 + \log_2 N)$

as shown in Fig. 6.

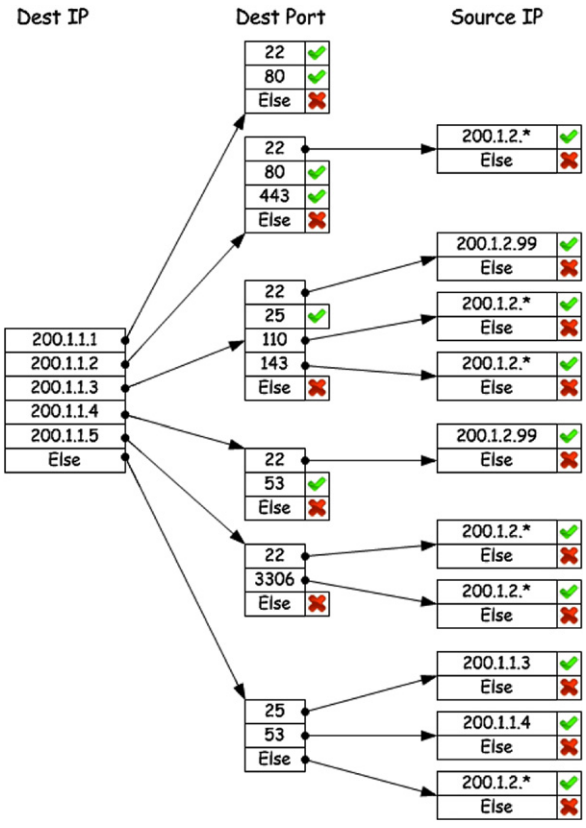

Fig. 4. A basic Tree-Rule firewall structure.

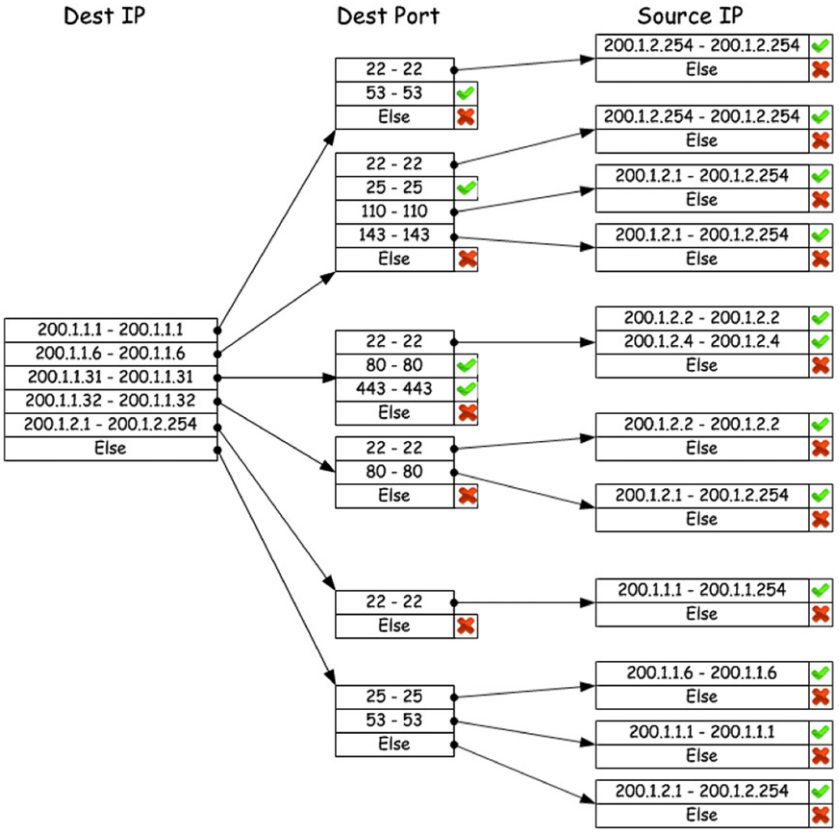

Fig. 5. Improved design of a Tree-Rule firewall using IP address ranges and port ranges.

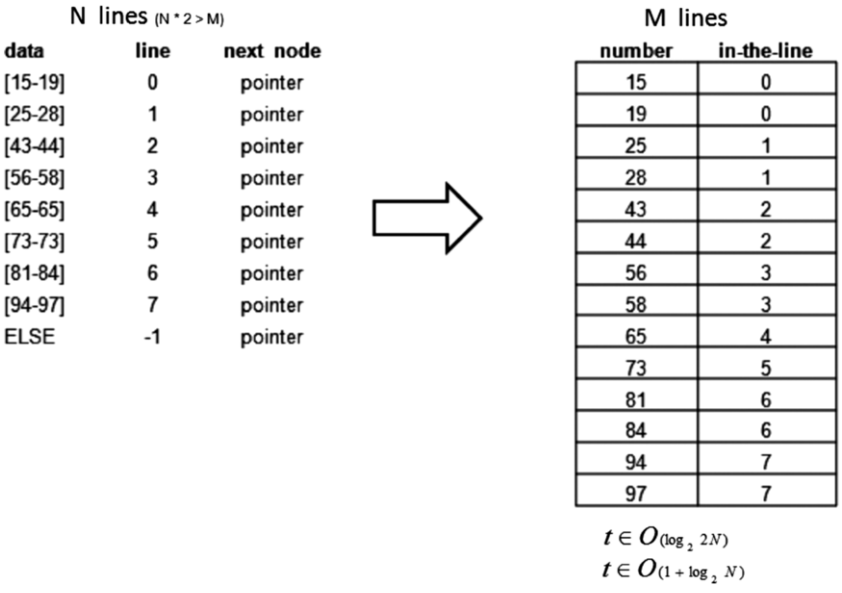

Fig. 6. Slightly increased 't' when a range of number is applied.

As shown in Fig. 6, a new attribute called 'in-the-line' is added to the data structure of a node to help search the data using a Binary Search algorithm (in array). The searching results as shown on the right part of Fig. 6 can be that

1. the number is certainly found (e.g., searching for number 25),
2. the nuumber is not found but still on the range (e.g., searching for number 17), and
3. the number is not found and not on the range (e.g., searching for number 20).

In the first two cases, the firewall will check 'Dest Port' (see Fig. 5) based on the link indicated in the 'in-the-line' (see Fig. 6). For example, when searching for number 17, the firewall will check the 'Dest Port' on link 0. To search for numbers within a node, we need only use a regular Binary Search algorithm, e.g., the algorithm presented on http://www.cosc.canterbury.ac.nz/mukundan/dsal/BSearch.html, and utilize the 'first', 'mid' and 'last' parameters to see that the searching result belongs to which of those three cases listed above.

Regarding time consuming in a node, it can be concluded that

$t \in O(1 + \log_2 N)$

where N is the row number within a node (see the left figure of Fig. 6). The data searching on the nodes will not be more than 3 times per packet and they will only occur in the root node, 'Dest Port' node and 'Source IP' node. If the searching times of the three nodes are t_1, t_2, and t_3, respectively, the access decision time per packet will be $t = t_1 + t_2 + t_3$. If the numbers of the lines within the three columns are N_1, N_2, and N_3, respectively, the decision time per packet will be estimated as follows:

$t \approx k_1(1 + \log_2 N_1) + k_2(1 + \log_2 N_2) + k_3(1 + \log_2 N_3)$

where k_1, k_2, and k_3 are all constants.

3.3. Implementation

The firewall implementation is conducted on Cent OS Linux with the Tree-Rule firewall functioned on Netfilter (see Fig. 7). Similar to IPTABLES, we focus on the network firewall that verifies the packet forwarding between the network interfaces. Our algorithm, NF_IP_FORWARD, includes three programs as follows.

1. Core Firewall. It is written with C language on Linux in order to detect the packets and make a decision on tree rule regulation for whether the packets should be accepted or dropped. This program runs on the Kernel Space with a file type of ".ko".
2. Rule Sender. This is written with C language on Linux in order to receive tree rules from GUI (running on the user's Windows XP/7/8). Rule Sender would send the tree rules to Core Firewall through 'procfs Virtual File System', a specific memory for the data exchange between the regular software and the software functioning on Kernel. This Rule Sender runs on the User Space (not the Kernel Space).
3. GUI. It is written with C# language on Windows in order to communicate with users so that each user could create a graphical tree rule. After creating and editing a tree rule, the user can either save or send/apply the rule to the firewall so that the rule can be functioned. GUI will communicate with the 'Rule Sender' on the firewall.

To install and run a Tree-Rule firewall, users do not need to uninstall the IPTABLES software from the system but run both of them together. Before running the Tree-Rule firewall, users must launch a command 'service iptables stop'. Meanwhile, running the Tree-Rule firewall can be commanded with 'insmod CoreFirewall.ko' and './RuleSender', respectively.

4. Experimental results

The firewall is tested on both regular networks (LANs) and cloud environment. The results are presented in Sections 4.1 and 4.2, respectively.

4.1. Testing in LAN

We conduct the performance test to compare between Tree-Rule firewall and IPTABLES (a free firewall popularly used on Linux) on the same computer and OS.

For testing, we use three computers (Intel 2.8 GHz CPU with 4GB RAM) according to their roles as the Firewall, the Target, and the Attacker. The Firewall has two network cards (100 mbps speed) to be directly connected with the target and the attacker. The firewall computer is located in the middle. Testing is conducted to assess the CPU load and throughput when the number of the rules is increased. At an early stage, we add a similar number of firewall rules that users normally used, and the number is between 100 and

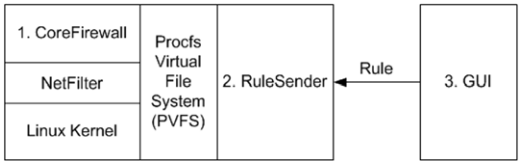

Fig. 7. Implementation of Tree-Rule firewall.

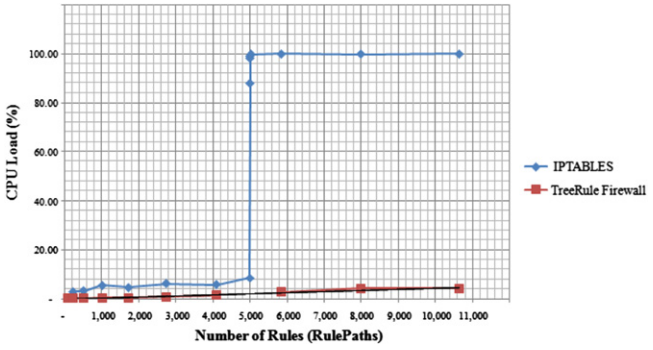

Fig. 8. CPU load of firewalls.

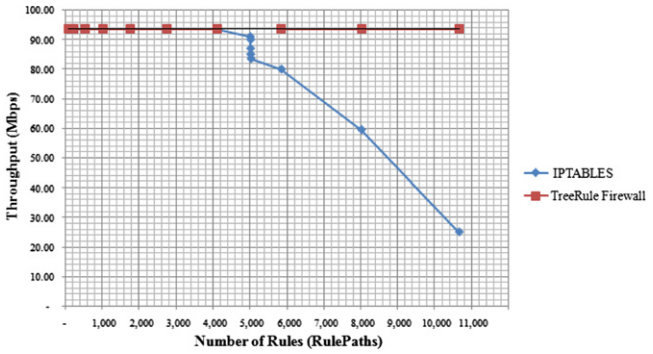

Fig. 9. Throughput of firewalls.

200 rules or rule paths. In the Tree-Rule firewall, the number of the rule paths is calculated based on the total number of the actions as illustrated in Fig. 5. We generate the rules by randomizing the Source IP addresses, Destination IP Addresses and Destination Ports, and define the actions as 'Accept' to allow all packets passing through the firewall (in order to obtain accurate throughput). Besides, we boot the Attacker computer with Back Track 5 R3 and generate the packets with hping3 command to create the packets with randomized IP addresses and ports before sending them to the Target as many as possible with the maximum rapidity by using '–flood' parameter. The testing results are shown in Figs. 8 and 9.

From the results shown above, it is found that using 200–4000 rules or rule paths requires a small space of CPU (3%–5% for IPTABLES and 0.4%–1.0% for Tree-Rule firewall); meanwhile, the throughput using either firewall is close to 100% (93.6% for both IPTABLES and Tree-Rule firewalls). Then, the rule number is constantly enlarged. We discover that if the rule number is increased to more than 5000 rules, the computer with IPTABLES needs more space on the CPU and the CPU usage is close to 100%, whereas the computer with Tree-Rule firewall requires only up to 4.3% of CPU. In terms of throughput, if there are more than 5000 rules, the throughput of the computer with IPTABLES will significantly decrease. However, the throughput of the computer with the Tree-Rule firewall remains regularly high.

Table 3
Throughput comparison between the Tree-Rule firewall and IPTABLES on ESXi and Hyper-V.

No. of rules	On ESXi		On Hyper-V	
	Tree-Rule firewall	IPTABLES	Tree-Rule firewall	IPTABLES
8	96.45	99.13	93.12	98.59
64	95.47	98.43	92.14	93.19
216	96.33	95.61	89.04	85.72
512	95.16	89.62	87.22	75.20
1000	96.58	80.34	84.79	62.66
1728	93.54	71.33	84.49	46.48
2744	92.30	59.82	87.70	35.61
4096	94.02	47.11	88.12	24.85

4.2. Testing on cloud environment

The most popular four hypervisors are ESXi (from VMware), KVM, Hyper-V (from Microsoft) and Xen Server [21–28]. In this research, we pay attention to ESXi and Hyper-V only because Hyper-V is easy to use (it is a Microsoft Windows based server) and ESXi is very reliable (noting that VMware company is the pioneer of Virtual Machines). The Tree-Rule firewall is tested on both operating systems. Typically, ESXi has its own firewall built in but Hyper-V has not. However, there is a firewall called the '5Nine vFirewall' which has been developed to suit functioning on Hyper-V and is recently popular. Both the ESXi firewall and 5Nine vFirewall are Listed-Rule firewalls. In this subsection, we compare the CPU loads and throughputs between the Tree-Rule firewall and IPTABLES on ESXi and Hyper-V respectively. We also indicate the disadvantages of ESXi's own firewall and '5Nine vFirewall' on Hyper-V compared with the Tree-Rule firewall.

4.2.1. Testing on ESXi

From the test, it is found that ESXi firewall protects only ESXi itself (i.e., the hypervisor) but is unable to protect the internal virtual machines. Therefore, to keep the internal virtual machines protected, users require additional firewalls.

Moreover, we test both the Tree-Rule firewall and IPTABLES on ESXi with 2^3, 4^3, 6^3, . . ., and 16^3 rules (i.e., 8, 64, 216, . . ., and 4096 rules) respectively. The results showing the throughput of these firewalls are demonstrated in Table 3 and Fig. 10.

4.2.2. Testing on Hyper-V

Recall that the 5Nine vFirewall is a firewall developed for functioning on Hyper-V. The benefit of the 5Nine vFirewall, compared with ESXi, is that the 5Nine vFirewall can protect the virtual machines from the external attacks and prevent the attacking between the virtual machines [29,30]. However, one disadvantage of vFirewall is that it is required to install agents inside the virtual machines and this is not feasible for the virtual machines with Linux OS since they do not have an agent. Another disadvantage of vFirewall is the instability. For example, with a huge number of rules, the firewall will set a virtual machine into a PUASE state.

We also test both the Tree-Rule firewall and IPTABLES on Hyper-V with 2^3, 4^3, 6^3, . . ., and 16^3 rules (i.e., 8, 64, 216, . . ., and 4096 rules) respectively. The results showing the throughput of these firewalls are also demonstrated in Table 3 and Fig. 10.

4.2.3. Testing analysis

In Table 3, the percentage for the throughput is obtained by dividing the actual throughput (in Mbps) by the maximum throughput (i.e., the forwarding rate without applying any firewall), and then multiplying the result by 100. In our experiments, the maximum throughput of the firewall computer is 615 Mbps when no firewall rules are applied. This number depends on CPU clock speed, OS version, quality of hardware, NICs, etc.

We only compare the throughput of IPTABLES and Tree-Rule firewall because thousands of IPTABLES rules can be created using shell scripts, and the rules for a Tree-Rule firewall under a cloud environment can easily be produced by modifying our source code written for the GUI of Tree-Rule firewall creation (see Section 3.1). Because the proposed Tree-Rule firewall has three attributes (i.e., Source IP, Dest IP and Dest Port), its rules can be easily produced using a three layer programming loop.

On the other hand, it is not easy to create thousands of 5Nine vFirewall rules or ESXi's rules using the corresponding GUI. As a result, we are not able to test the speeds of the ESXi firewall and 5Nine vFirewall when the sizes of their rule lists are large. Nevertheless, we have investigated and found that when the positions of rules are swapped, these rules create all types of the rule conflicts that we have mentioned in Section 2, because these two firewalls are Listed-Rule firewalls.

From the three types of the firewall positioning (as previously mentioned in Section 1.2), the conclusion and analysis are made as follows.

A. Positioning a single firewall to prevent the external attacks (Fig. 1(a))
   - The ESXi firewall is capable, but only protects the ESXi itself, not the internal virtual machines.
   - The 5Nine vFirewall is capable and can protect the internal virtual machines.
   - The Tree-Rule firewall and IPTABLES are capable. If there are too many virtual machines, the number of the rules will increase and the IPTABLES will encounter some problems (some Cloud Service Providers may contain more than 40,000 of the internal virtual machines). Besides, with too many rules, the rule creation/modification may simply cause redundant rules and conflict rules, which mostly occur with the firewall employing a listed rule (e.g., IPTABLES, Cisco ACL and other firewalls recently available).

B. Positioning the firewalls based on the number of VMs or Cloud networks/Company networks (Fig. 1(b))
   - The ESXi firewall is incapable since it is unable to protect the internal virtual machines.
   - The 5Nine vFirewall is capable and gives the same results as the topology does. It needs neither installing additional software nor rearranging the virtual network within the hypervisor. Although it is flexible, it can only run on Microsoft platforms.
   - The Tree-Rule firewall and IPTABLES are capable. Compared with the 5Nine vFirewall, these two firewalls require much more disk spaces and RAMs, since they need to be installed on every position of virtual machines (or cloud networks/company networks), which is time consuming. Moreover, the virtual network inside the hypervisor needs to be rearranged. Differently, when comparing between the Tree-Rule firewall and IPTABLES, both of them require equal disk spaces and RAMs since they consume less resource compared to the size of their OS (Linux).

C. Positioning a single firewall with various interfaces (Fig. 1(c))
   - The ESXi firewall is incapable.
   - The 5Nine vFirewall is capable for this topology, with neither additional software nor rearranging the virtual network inside the hypervisor. This firewall is flexible since it can communicate directly with the hypervisor. However, it can only work on Microsoft platforms.

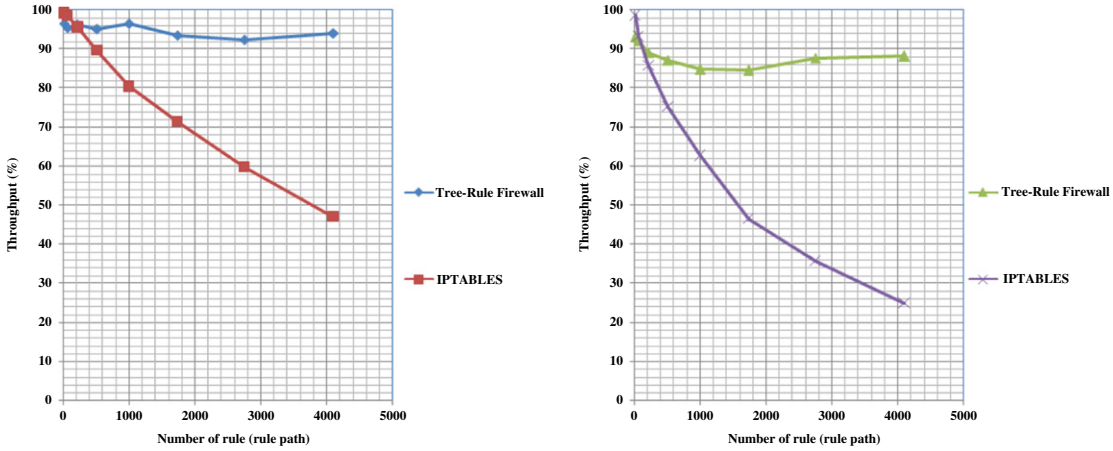

Fig. 10. Throughput of firewalls experimented on ESXi (left) and Hyper-V (right).

Table 4
Capability comparison among the firewalls on cloud environment.

	ESXi firewall	5Nine vFirewall	IPTABLES	Tree-Rule firewall
Can protect hypervisor	Yes	Yes	Yes	Yes
Can protect internal VMs	No	Yes	Yes	Yes
Can prevent attacking between VMs	No	Yes	Yes	Yes
Stable	Yes	No	Yes	Yes
Fast packet decision	No	No	No	Yes
Support more than 5000 rules	No	No	No	Yes
No shadowed/redundant/confliction rules	No	No	No	Yes
No down time for adding new VMs	No	Yes (only on Hyper-V)	No	No

- The Tree-Rule firewall and IPTABLES are capable, and the down time during an appending of the virtual NIC (Network Interface Card) and the virtual network inside the hypervisor need to be rearranged. Moreover, if there are too many virtual NICs (because of many virtual machines), the number of the rules will increase and the IPTABLES may encounter problems. With a huge number of rules, the rule creation/modification may simply cause redundant rules and conflict rules.

Based on Sections 4.2.1–4.2.3, the additional conclusion is shown in Table 4.

The 5Nine vFirewall is flexible with the increase of VMs (since it directly communicates with the hypervisor via API), and both the Tree-Rule firewall and IPTABLES need the installation of an additional instance (Guest OS) or more virtual NICs to protect the newly added virtual machines. However, the 5Nine vFirewall has four disadvantages. First, the 5Nine vFirewall can only be used on Microsoft platforms. Second, users have to install its agents on all VM clients with Windows XP SP3 (or higher version) Operating System. Third, the 5Nine vFirewall is difficult to manage rules since the user must select a VM (virtual machine) to manage its rule. The rules that protect each VM are on different pages. On the contrary, rules of Tree-Rule firewall and IPTABLES are on the same page (see Fig. 1(c) for the case of using various virtual NICs to connect with their VMs). Last, the rules of the 5Nine vFirewall possibly cause redundancy and confliction (as its rules are based on the Listed-rule model). Redundancies and conflictions, however, can never occur in any Tree-Rule firewall.

5. Conclusion and future work

In this study, we aim to identify the limitations of the currently used firewalls (Listed-Rule firewalls) and have found five disadvantages including: (1) possibility of shadowed rule that causes the problematic network security and functional speed; (2) rule switching that changes the meaning of the rules entailing the problematic network security; (3) possibility of redundant rules that entails the problematic speed; (4) designing that needs to place the bigger rule after smaller rules, which cause the designing difficulty; and (5) sequential rule processing that causes the problematic speed.

Furthermore, we have proposed the Tree-Rule firewall that demonstrates none of the above-mentioned limitations. The Tree-Rule firewall utilizes rules in a tree data structure, and forwarding decision of an input packet based on tree rules will follow the tree structure so that the decision on the packet becomes faster. The Tree-Rule firewall has been tested and compared with IPTABLES on LAN and we have found that the Tree-Rule firewall gives better performance.

Moreover, the Tree-Rule firewall has been tested on a cloud environment and we have found it more suitable than the Listed-Rule firewall for a cloud network, which is a large network that requires a number of computers and large rule size, and found it rapid for forward decision on packets. We have made a capability comparison among the proposed Tree-Rule firewall, the IPTABLES and two state-of-the-art firewalls under a cloud environment. The advantages of using the Tree-Rule firewall have been demonstrated.

In the next step of our study, we will extend the number of columns in the tree structure to include more than three attributes (e.g., adding Protocol and MAC address columns) and investigate the order of column localization because the speed of the Tree-Rule firewall may depend on the attribute specified for the root node.

We will further develop the Tree-Rule firewall to make it applicable for NAT (Network Address Translation), IPv6 and VPN (Virtual Private Network), and able to communicate directly with a hypervisor (as a component of the hypervisor). Furthermore, the Tree-Rule firewall will be improved to become more flexible than the 5Nine vFirewall with Hyper-V so that more VMs can be added into the hypervisor and new virtual NICs can be added on the firewall itself without any down-time, to provide the optimal benefits for cloud networks.

References

[1] A. Shebanow, R. Perez, C. Howard, The effect of firewall testing types on cloud security policies, International Journal of Strategic Information Technology and Applications 3 (3) (2012) 60–68.
[2] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, M. Rajarajan, A survey of intrusion detection techniques in cloud, Journal of Network and Computer Applications 36 (1) (2013) 42–57.
[3] E. Al-Shaer, H. Hamed, Firewall policy advisor for anomaly detection and rule editing, in: Proceedings of the IEEE/IFIP Integrated Management, IM, 2003, pp. 17–30.
[4] E. Al-Shaer, H. Hamed, R. Boutaba, M. Hasan, Conflict classification and analysis of distributed firewall policies, IEEE Journal on Selected Areas in Communications 23 (10) (2005) 2069–2084.
[5] H. Haded, E. Al-Shaer, Taxonomy of conflicts in network security policies, IEEE Communications Magazine 44 (3) (2006) 134–141.
[6] S. Hazelhusrt, Algorithms for analyzing firewall and router access lists, Technical Report TR-WitsCS-1999, Department of Computer Science, University of the Witwatersrand, 1999.
[7] P. Eronen, J. Zitting, An expert system for analyzing firewall rules, in: Proceedings of the 6th Nordic Workshop on Secure IT-Systems, NordSec, 2001, pp. 100–107.
[8] L. Yuan, J. Mai, Z. Su, FIREMAN: A toolkit for firewall modeling and analysis, in: Proceedings of the 2006 IEEE Symposium on Security and Privacy, 2006, pp. 199–213.
[9] L. Zhao, A. Shimae, H. Nagamochi, Linear-tree rule structure for firewall optimization, in: Proceedings of Communications Internet and Information Technology, 2007, pp. 67–72.
[10] Cisco content services switch basic configuration guide, 2002. http://www.cisco.com/en/US/docs/app_ntwk_services/data_center_app_services/css11500series/v7.10/configuration/basic/guide/basicgd.pdf.
[11] A. Liu, M. Gouda, Diverse firewall design, IEEE Transaction on Parallel and Distributed Systems 19 (9) (2008) 1237–1251.
[12] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Generation Computer Systems 28 (3) (2012) 583–592.
[13] S. Kima, S. Kimb, G. Leea, Structure design and test of enterprise security management system with advanced internal security, Future Generation Computer Systems 25 (3) (2009) 358–363.
[14] Virtual firewall appliances: trust misplaced, 2012. http://blog.cloudpassage.com/2012/01/24/virtual-firewall-appliances-trust-misplaced/.
[15] VMware virtual switch isolation, 2013. http://serverfault.com/questions/186735/vmware-virtual-switch-isolation.
[16] R.P. Goldberg, Architectural Principles for Virtual Computer Systems, Harvard University, 2010, pp. 22–26.
[17] T. Chomsiri, X. He, P. Nanda, Limitation of listed-rule firewall and the design of tree-rule firewall, in: Proceedings of the 5th International Conference on Internet and Distributed Computing Systems, China, 2012, pp. 275–287.
[18] C. Pornavalai, T. Chomsiri, Firewall policy analyzing by relational algebra, in: Proceeding of the 2004 International Technical Conference on Circuits/Systems, Computers and Communications, ITC-CSCC, 2004, pp. 214–219.
[19] Binary search tree, 2013. http://en.wikipedia.org/wiki/Binary_search_tree.
[20] Linear search, 2013. http://en.wikipedia.org/wiki/Linear_search.
[21] Top 10 hypervisors: choosing the best hypervisor technology, 2011. http://searchservervirtualization.techtarget.com/tip/Top-10-hypervisors-Choosing-the-best-hypervisor-technology.
[22] Virtualization wars: VMware vs. Hyper-V vs. XenServer vs. KVM, 2011. http://www.networkworld.com/news/2011/103111-tech-argument-virtualization-252559.html.
[23] M.A. Bamiah, S.N. Brohi, S. Chuprat, Using virtual machine monitors to overcome the challenges of monitoring and managing virtualized cloud infrastructures, in: Proceedings of the fourth International Conference on Machine Vision, ICMV, 2011.
[24] I. Voras, M. Orli, B. Mihaljevi, An early comparison of commercial and open-source cloud platforms for scientific environments, in: Proceedings of 6th KES International Conference, KES-AMSTA, 2012, pp. 164–173.
[25] H. Mousannif, I. Khalil, G. Kotsis, Collaborative learning in the clouds, Information Systems Frontiers 15 (2) (2013) 159–165.
[26] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehörster, A. Brinkmann, Non-intrusive virtualization management using libvirt, in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE, 2010, 574–579.
[27] Initial findings with VMware ESXi and Hyper-V comparison, 2013. http://redmutt.com/?p=228.
[28] Price: VMware (ESXi) vs. Hyper-V (Windows server 2012), 2012. http://www.insideris.com/price-vmware-esxi-vs-hyper-v-windows-server-2012/.
[29] 5nine virtual Firewall 2.1 for Microsoft Hyper-V, 2011. http://www.5nine.com/Docs/vFirewall2_data_sheet.pdf.
[30] 5nine virtual Firewall 2.0 (v-Firewall, Beta), 2009. http://www.5nine.com/Docs/VFW2_QSG_07.pdf.

Xiangjian He is a Professor of Computer Science. He is also the Director of Computer Vision and Recognition Laboratory, the leader of Network Security Research group, and a Deputy Director of Research Centre for Innovation in IT Services and Applications (iNEXT) at the University of Technology, Sydney (UTS). He is an IEEE Senior Member. He has been awarded Internationally Registered Technology Specialist by International Technology Institute (ITI). His research interests are network security, image processing, pattern recognition and computer vision.

Thawatchai Chomsiri is a Ph.D. student at the Faculty of Engineering and Information Technology (FEIT) of the University of Technology, Sydney (UTS), Australia. He is also an Assistant Professor at the Department of Information and Communication Technology in the Faculty of Informatics of the Mahasarakham University, Thailand. Mr. Chomsiri has 17 years of experience in industry, teaching and research. His research interests are computer networking, and computer and network security.

Priyadarsi Nanda is a Senior Lecturer in the School of Computing and Communications, and is a Core Research Member at the Centre for Innovation in IT Services Applications (iNEXT). His research interests are network QoS, network securities, assisted health care using sensor networks, and wireless networks. Dr Nanda has over 23 years of experience in teaching and research.

Zhiyuan Tan is with the Faculty of Engineering and Information Technology (FEIT) at the University of Technology, Sydney (UTS), Australia and a research member of Research Centre for Innovation in IT Services and Applications (iNEXT). He is also a research member of the Information and Communication Technologies (ICT) Centre, Commonwealth Scientific and Industrial Research Organisation (CSIRO). His research interests are network security, pattern recognition, machine learning and P2P overlay network.