Comparing and debugging firewall rule tables

L. Lu, R. Safavi-Naini, J. Horton and W. Susilo

Abstract: Firewalls are one of the essential components of secure networks. However, configuring firewall rule tables for large networks with complex security requirements is a difficult and error prone task. A method of representing firewall rule table that allows comparison of two tables is developed, and an algorithm that determines if two tables are equivalent is provided. (That is the set of packets that are permitted by the two tables are the same.) How such algorithm can assist system administrators to correctly implement organisational policy is discussed. The proposed approach is implemented and the results of the experiments are shown.

1 Introduction

In battle against network attacks, firewalls have played one of the most important roles. A firewall protects networks against malicious attacks by filtering out the unwanted network traffic from the traffic entering the secured network. The filtering decision is made using a table of rules written by the administrator and capturing organisational security policy [1].

Translating a high-level security policy written in a natural language into firewall rule tables, a much lower-level description of the policy, is an error prone task.

One approach for providing higher assurance about correctness of the rule table is to have two independent implementations of the rule table from the original policy, and compare the two tables. If the two tables are 'equivalent', that is permit (and deny) the same set of packets, one concludes that it is very likely that the policy is correctly implemented. On the other hand if the two tables are not equivalent, it is desirable to be able to 'debug' the rule tables, that is, locate rules that have resulted in inconsistency between the tables. Such rules may exist for reasons such as ambiguity in the description of the high-level policy, and can be effectively addressed if the in-equivalence of the two tables can be localised.

A similar question arises when firewall rules need to be adjusted. Adding a new rule, or changing an existing rule, is often required because of policy changes or in response to a discovered security flaw. A new rule is added to a firewall to permit a new set of packets, or stop a set of permitted packets. The new rule may be added at the top of the table, or at an intermediate point. The new set of packets that are permitted (or denied), although not explicitly matched by a single rule in the original firewall table, may have been implicitly covered by the combination of existing rules, so the new rule was not really necessary. (This may indicate incorrect implementation of an existing table, especially if the new rule is positive and specifies packets that have been unintentionally permitted by existing rules.) Therefore it is important to investigate the effect of the addition of new rules by determining the set of packets that are permitted (or denied) 'only' because of the new rule.

In this paper, we give a systematic method of answering questions such as the above. For a rule table T, we construct a new table T' that consists of only positive, non-overlapping rules. We start from the first rule in the table and examine rules one by one, replacing each with what we call the 'effective part' of the rule. The resulting table T' has the property that (1) it permits the same packets as T, and (2) the set of permitted packets is partitioned into disjoint subsets, each corresponding to a distinct rule. Because of these properties, the order of rules in T' is not important anymore and the information is effectively encoded into the structure of rules.

We use this new representation to compare two tables and in particular answer questions such as the equivalence of two tables, or the effect of adding rules to a table: to verify that a table T_1 permits (or denies) the same packets as table T_2, we convert the two tables into their corresponding new forms, T'_1 and T'_2, respectively, and verify if T'_1 is equivalent to T'_2. If the two are not equivalent, the new representation can be used to find rules in the original tables that cause the conflict (Section 4).

When adding a new rule r to table T, an extended table T_r is constructed. The rule is redundant if $T' = T'_r$ where T'_r is representation of T_r in the new form. For non-redundant rules the new form can be used to determine the set of packets that are permitted (or denied) 'only' because of r.

We implemented algorithms that allow the equivalency of two tables to be checked and provide a GUI prototype that can answer questions like the above using the proposed method, and demonstrated in a case study its application for comparing and debugging firewall rule tables.

1.1 Related work

Management tools for developing firewall rule tables have found much attention in recent years. Modelling languages that are more expressive and closer to natural language than firewall rules alleviate the error-prone translation tasks [2–4]. On the other hand, less attention has been paid in particular to the problem of comparing firewall rules and rule tables.

Hazelhurst *et al.* [5, 6] provided an approach to compare firewall rule tables using binary decision diagrams (BDD).

doi:10.1049/iet-ifs:20060171

Paper first received 24th December 2006 and in revised form 19th June 2007

The authors are with Centre for Information Security Research, School of Information Technology and Computer Science, University of Wollongong, Wollongong, NSW, Australia

E-mail: ll97@uow.edu.au

Table 1: Firewall rule table using both negative and positive rules

Rule number	Source IP address range	Destination IP address range	Service	Action
1	192.168.13.10	192.168.13.11	HTTP	drop
2	192.168.13.0/24	any	HTTP	pass

A rule table is converted into a Boolean expression, which can be compactly and uniquely represented by a BDD. The uniqueness of BDD is an important property that allows us to determine the equivalence of rule tables by comparing their BDD representations. Rule tables are represented as Boolean expressions and so Boolean operations can be used to answer questions such as the equivalence of two tables and determining packets that are accepted by one table but not the other.

BDD representation does not preserve the original form and order of rules and so although it can be used to determine equivalence of two tables, it cannot determine the conflicting rules. This is an important advantage of the approach proposed in this paper.

Al-Shaer and Hamed [7, 8] provided a tool to assist in inserting or editing rules in a rule table. An inserted rule is compared sequentially with each rule in the table to ensure that the set of packets it matches is not in the meantime already matched by any preceding rule such that the insertion would not have been necessary, which is usually considered an error. A similar approach is applied to rule edition. However, it is often the case that the packets matched by the new rule, although not explicitly matched by a single existing rule, may have been matched by a combination of existing rules and so the new rule need not be added. The approach proposed in this paper detects such cases and so enhances Al-Shaer and Hamed's work.

The rest of the paper is organised as follows. Section 2 presents related background knowledge on firewall rules and rule tables. In Section 3, we give a formal yet intuitive definition of equivalence between firewall rule tables, as well as some other fundamental definitions. In Section 4, we present a theorem and a set of algorithms to determine whether two rule tables are equivalent, and locating the rules that permit packets denied by the other rule table in the case that inequality holds. In Section 5, we provide a GUI prototype written in Java, along with a complete example of applying the prototype to compare and debug firewall configurations. Section 6 concludes the paper.

2 Firewall background

In the rest of this paper, protocol means transport layer protocol such as TCP or UDP. Let SrcIPAd, DestIPAd, SrcPort, DestPort, Protocol denote source IP address, destination IP address, source port number, destination port number and protocol, respectively. Let $[a, b]$ denote the set of numbers x between a and b; that is, $a \leq x \leq b$. Then SrcIPAd, DestIPAd $\in [1,\ 2^{32} - 1]$, SrcPort, DestPort $\in [1,\ 2^{16} - 1]$ and Protocol $\in$ {TCP, UDP}. We also use $[X]$ to denote a specific range of numbers that can be assigned to element X, for example, [IP] represents the range of legal IP addresses. We assume a packet can be represented in the following form

$$(\text{SrcIPAd},\ \text{DestIPAd},\ \text{SrcPort},\ \text{DestPort},\ \text{Protocol})$$

Let $\mathcal{P}$ denote the set of all packets. It can be seen that $\mathcal{P}$ is a finite set of size $|\mathcal{P}| = 2^{32} \times 2^{32} \times (2^{16} - 1) \times (2^{16} - 1) \times 2$.

2.1 Firewall rules

A firewall filters packets based on fields defined in the network and/or transport layer, including source IP address ([SrcIPAd]), destination IP address ([DestIPAd]) and service (Srv).

[SrcIPAd] and [DestIPAd] specify an acceptable range of source/destination IP addresses, respectively. Srv is the combination of protocol, source port number and destination port number. Note that the source port number is rarely significant, and commonly any source port is acceptable. Each rule is associated with an action (Act) field, the value of which is either 'pass' or 'drop', indicating whether a packet is passed or dropped when it is matched by this rule. Rules with the pass/drop action are referred to as positive/negative rules, respectively.

There may also be other relevant fields, such as the rule number (rn) that identifies the order of rules. In general, a firewall rule can be presented as a 4-tuple ([SrcIPAd], [DestIPAd], Srv, Act).

A rule r specifies a subset $\mathcal{P}_r \subseteq \mathcal{P}$ of packets that it matches. A packet p is matched by a rule r (and so $p \in \mathcal{P}_r$) if $\text{SrcIPAd}_p \in [\text{SrcIPAd}_r]$, $\text{DestIPAd}_p \in [\text{DestIPAd}_r]$ and the combination of $(\text{Protocol}_p, \text{SrcPort}_p, \text{DestPort}_p) = \text{Srv}_r$.

When a packet arrives at the firewall, it is mapped into the form of firewall rules and then matched against the firewall rule table. The first rule that matches the packet will determine the action on the packet. For example, if a packet p is matched by two rules r and r' where r' precedes r in the order, then r' will be applied to p but r will not. If p is matched by multiple rules preceding r, then which rule will be applied to p depends on the order of the matching rules. If no preceding rule matches p, then r will be applied to p.

We use $\{\mathcal{R}\}$ to denote the set of packets to which rule R is applied. These are the packets for which R is the first rule matched in the rule table. In general, $\{\mathcal{R}\} \subseteq \mathcal{P}_R$ – that is, the set of packets to which R is applied is a subset of the set of packets that R matches.

2.2 Firewall rule table

A firewall rule table, T, is an ordered set of rules, and specifies a set of packets $\mathcal{P}_T$ that are passed by the rule table. We use $(R_1, R_2, \ldots, R_n)$ to denote a rule table consisting of rules $R_1, R_2, \ldots, R_n$ in order.

Let Match$(p,\ T)$ denote a function that takes a packet p and a rule table T and returns the first rule that matches p. This rule, denoted by $r_{p,T}$, is the rule that will be applied to the packet p. There can be more rules that match p but only $r_{p,T}$ will be applied to p.

A firewall defaults to accept or drop packets not matched explicitly. In most situations, firewalls use 'drop packet' as the default action so that only packets that are explicitly permitted are passed by the firewall [9], and this is the assumed default action in the following discussion. That being said, if $r_{p,T} = \phi$ where ϕ denotes the empty set, T will drop p. As a result, we need not use negative rules explicitly to drop prohibited packets. Explicit positive rules are needed to accept legitimate packets. Negative rules provide readability and compactness for firewall table by denying packets that are permitted by subsequent positive rules. For example, consider the case that connections from a large block of network addresses are permitted with the exception of a small range of addresses in the middle of the larger block. This can be implemented more compactly using a negative rule denying traffic from the small range of addresses followed by a positive rule that permits traffic from the larger block.

2.3 Example

Consider the firewall rule table shown in Table 1. R_1 drops packets from 192.168.13.10 and to 192.168.13.11 for the HTTP service so that R_2 can be written in a more compact and readable form.

The rule table presented in Table 1 can be written with only positive rules as shown in Table 2. It is seen that Table 1 is more compact and readable (allowing packets from 192.168.13.0/24 to go anywhere with the exception that packets from 192.168.13.10 are not allowed to reach 192.168.13.11).

This example shows that the negative rules can be used to drop a subset of packets matched by subsequent positive rules, and could make the rule table more compact and readable. A negative rule r will not affect the set of permitted packets if $\mathcal{P}_r$ does not have non-empty intersection with the sets $\mathcal{P}_{R_1}, \ldots, \mathcal{P}_{R_i}$ where $R_1, \ldots, R_i$ are positive rules following r. That is, a negative rule r will not affect $\mathcal{P}_T$ if $\forall R_j$, $\mathcal{P}_{R_j} \cap \mathcal{P}_r = \phi$, where R_j are positive rules following r.

3 Preliminaries

Definitions: Two firewall rule tables T_1 and T_2 are equivalent if $\mathcal{P}_{T_1} = \mathcal{P}_{T_2}$.

We assume that the firewalls drop all packets by default, and there is a 'deny all packets' at the end of the firewall rule table. We only need to consider packets that can pass the firewall. This is because $\mathcal{P}$, the set of all packets, is a finite set, if $\mathcal{P}_{T_1} = \mathcal{P}_{T_2}$, then T_1 and T_2 will also drop the same set of packets since $\mathcal{P} - \mathcal{P}_{T_1} = \mathcal{P} - \mathcal{P}_{T_2}$.

A positive rule specifies a set of packets that match the rule. However, the set of packets that will be permitted by the rule will also depend on other rules and their order. Consider a table with rule list $(R_1, \ldots, R_n)$. A packet p that matches a rule R_i in the table will be in one of the following categories.

1. p is matched by R_i and at least one rule R_j where $j < i$. In this case, p will be dropped or passed depending on R_j being negative or positive, respectively. In both cases, R_i will not be applied.
2. p is matched only by R_i but not any rule R_j where $j < i$. In this case, R_i will be applied to p and Match$(p, T) = R_i$.

In the following, we show how a positive rule can be divided into sub-rules to capture above cases. A rule R' is called a sub-rule of a rule R if $\mathcal{P}_{R'} \subset \mathcal{P}_R$.

Consider a firewall table with rule list $T = (R_1, \ldots, R_n)$.

Definition 2: For a rule R_i,

1. R'_i is the redundant part of R_i with respect to table T if R'_i is a sub-rule of R_i, and for some positive rule $R_j \in T$, where $j < i$, $\mathcal{P}_{R'_i} \subseteq \mathcal{P}_{R_j}$. In this case, R'_i will never be applied and $\{\mathcal{R}'_i\} = \phi$.
2. R''_i is the shadowed part of R_i with respect to table T if R''_i is a sub-rule of R_i, and for some negative rule $R_j \in T$, where $j < i$, $\mathcal{P}_{R''_i} \subseteq \mathcal{P}_{R_j}$. In this case, R''_i will never be applied and $\{\mathcal{R}_i'\} = \phi$.
3. R'''_i is the effective part of R_i with respect to table T if R'''_i is a sub-rule of R_i, and matches packets that are not matched by any rule $R_j \in T$, where $j < i$. In other words, $\mathcal{P}_{R'''} = \mathcal{P}_R - (\mathcal{P}_{R'} \cup \mathcal{P}_{R''})$.

If R is written as ([SrcIPAd], [DestIPAd], Srv, pass), then R', R'' and R''' can be written as, respectively, ([SrcIPAd1], [DestIPAd1], Srv, pass), ([SrcIPAd2], [DestIPAd2], Srv, pass) and ([SrcIPAd3], [DestIPA3], Srv, pass), where [SrcIPAd1] $\cup$ [SrcIPAd2] $\cup$ [SrcIPAd3] = [SrcIPAd] and [DestIPAd1] $\cup$ [DestIPAd2] $\cup$ [DestIPA3] = [DestIPAd]. This implies that $\mathcal{P}_R = \mathcal{P}_{R'} \cup \mathcal{P}_{R''} \cup \mathcal{P}_{R'''}$. The Venn diagram in Fig. 1 illustrates the relation between R, R', R'', and R'''.

Al-Shaer and Hamed [7] also used the concepts of shadowed and redundant rule in firewall rule tables. Our definition differs in that a rule R_i is divided into three sub-rules and the redundant/shadowed relation is defined between one of the sub-rules and a preceding rule R_j where $j < i$, as opposed to between R_i itself and R_j.

A packet p that is matched by R''' implies that p will not be passed or dropped by any preceding rule in the rule table and so R will be applied to p. A packet p that is matched by R' or R'' will be passed or dropped by a preceding rule in the rule table and so R will not be applied to it. We have the following properties for R', R'' and R'''.

A_1 A packet p that matches a rule table T is matched by the effective part of exactly one rule R. That is, $p \in \mathcal{P}_T \Rightarrow (\exists R \in T$ such that $p \in \mathcal{P}_{R'''}) \wedge (\forall r \in T$ we have $r \neq R \Rightarrow p \notin \mathcal{P}_{r'''})$.
A_2 Effective parts of rules are positive rules and are associated with non-overlapping subsets of packets. That is, $(\forall R_1, R_2 \in T, R_1 \neq R_2 \Rightarrow \mathcal{P}_{R_1'''} \cap \mathcal{P}_{R_2'''} = \phi)$.
A_3 It is possible to have $\mathcal{P}_{R'} \cap \mathcal{P}_{R''} \neq \phi$. However, $\mathcal{P}_{R'} \cap \mathcal{P}_{R'''} = \phi$ and $\mathcal{P}_{R''} \cap \mathcal{P}_{R'''} = \phi$.
A_4 If $p \in \mathcal{P}_{R'''}$, then Match$(p, T) = R$.
A_5 $\forall R \in T$, $\mathcal{P}_{R'''} \subseteq \mathcal{P}_T$.
A_6 $\forall R \in T$, $\mathcal{P}_{R'''} = \{\mathcal{R}\} = \{\mathcal{R}'''\}$ and $\{\mathcal{R}'\} = \{\mathcal{R}''\} = \phi$.

Given a firewall rule table $T = (R_1, \ldots, R_n)$, we call $T' = (r_1, \ldots, r_n)$ the effective representation of T, where $r_i = R'''_i$ if R_i is a positive rule in T, otherwise $r_i = \phi$. Slightly abusing notations, if we denote the effective part of a negative rule by ϕ for negative rules do not enlarge the set of allowed packets, then T' can be written as $T' = (R'''_1, \ldots, R'''_n)$.

Table 2: Firewall rule table using positive rules only

Rule number	Source IP address range	Destination IP address range	Service	Action
1	192.168.13.10	0.0.0.0–192.168.13.10	HTTP	pass
2	192.168.13.10	192.168.13.12–255.255.255.255	HTTP	pass
3	192.168.13.0–192.168.13.9	any	HTTP	pass
4	192.168.13.11–193.168.13.255	any	HTTP	pass

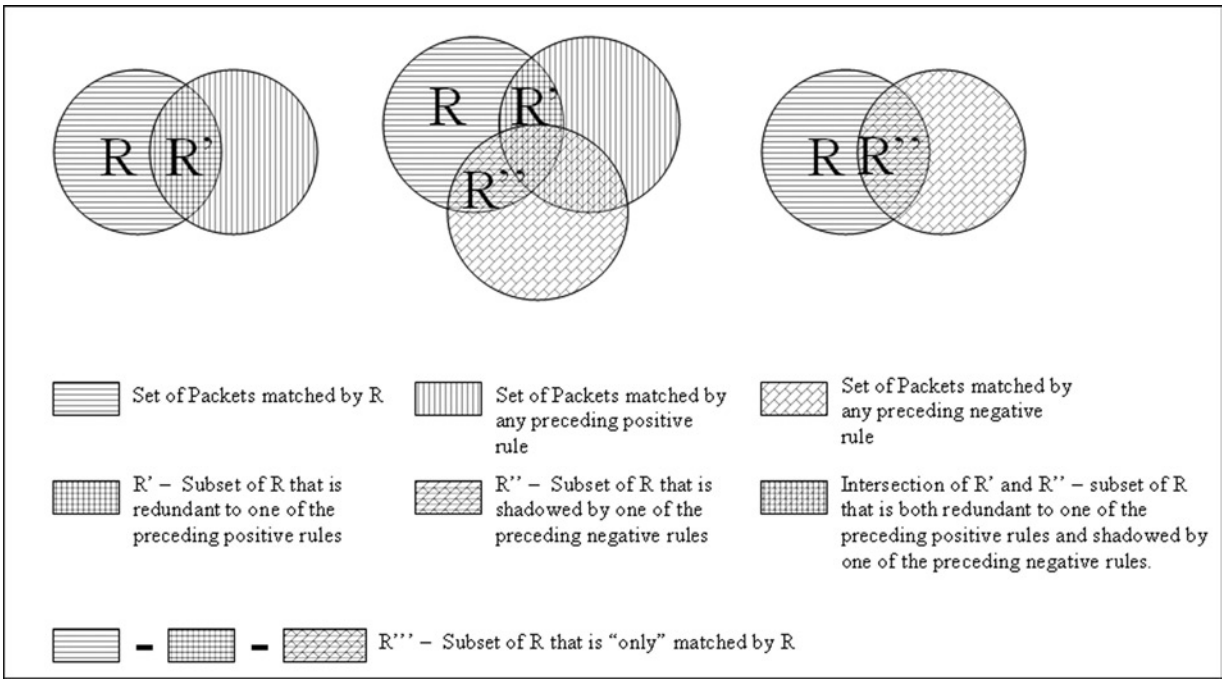

R
R'
R
R'
R''
R
R''
Set of Packets matched by R
Set of Packets matched by any preceding positive rule
Set of Packets matched by any preceding negative rule
R' – Subset of R that is redundant to one of the preceding positive rules
R'' – Subset of R that is shadowed by one of the preceding negative rules
Intersection of R' and R'' – subset of R that is both redundant to one of the preceding positive rules and shadowed by one of the preceding negative rules.
R''' – Subset of R that is "only" matched by R

Fig. 1 *Venn diagram illustrating R', R'' and R'''*
R''' is the set of packet 'only' matched by R

Definition 3: Consider two firewall rule tables T_1 and T_2 and let R denote a rule in T. Assume that both tables are converted into their effective representations. A rule $R \in T_1$ has equivalent rule set in T_2 if there exists a subset $R_1, R_2, \ldots, R_n \in T_2$ such that $\mathcal{P}_{R'''} \subseteq \mathcal{P}_{R_1'''} \cup \mathcal{P}_{R_2'''} \cup \cdots \cup \mathcal{P}_{R_n'''}$ Using property A_1 this implies that $\mathcal{P}_{R'''} \subseteq \mathcal{P}_{T_2}$. The equivalent rule set of a negative rule is defined to be ϕ.

If $R \in T_1$ has an equivalent rule set in T_2, then packets in $P_{R'''}$ are also permitted by the equivalent rule set. If R does not have an equivalent rule set in T_2, then not all packets in $P_{R'''}$ are permitted by T_2.

3.1 Example

We use an example to explain the above concepts in details. Consider the rule table given in Table 3.

For R_5, source IP address range 192.168.13.0/24 = 192.168.13.1–20 ∪ 192.168.13.20–30 ∪ 192.168.13.31–254. Destination IP address range 192.168.13.0/24 = 192.168.13.30–40 ∪ 192.168.13.40–50 ∪ 192.168.13.1–29,51–254.

When packets from 192.168.13.1–20 and to 192.168.13.0/24, or from 192.168.13.0/24 and to 192.168.13.30–40 arrive at the firewall, they will be matched and passed by R_1 and R_3. Although R_5 also matches them, it will not be applied. When packets from 192.168.13.20–30 and to 192.168.13.0/24, or from 192.168.13.0/24 and to 192.168.13.40–50 arrive at the firewall, they will be matched and dropped by R_2 and R_4. Although R_5 also matches them, it will not be applied. As a result, we can divide R_5 as shown in Table 4.

Consider another example as shown in Table 5, it can be seen that the effective parts of rule R_1, R_3 and R_6 are themselves, that is, they have no redundant or shadowed parts. Comparing Tables 3 and 5, it can be seen that R_5 in Table 3 has equivalent rule set in Table 5, which are R_1, R_3 and R_6.

Table 3: Example of rule table

Rule number	Source IP address range	Destination IP address range	Service	Action
1	192.168.13.1–20	any	HTTP	pass
2	192.168.13.20–30	any	HTTP	drop
3	any	192.168.13.30–40	HTTP	pass
4	any	192.168.13.40–50	HTTP	drop
5	192.168.13.0/24	192.168.13.0/24	HTTP	pass

4 Comparing firewall rule tables

In this section, we first prove theorems that specify the condition under which two rule tables are equivalent. We then use these results to give algorithms to determine if two rule tables are equivalent. If the two table are not equivalent, the algorithms will locate rules that are causing the conflict, that is, rules that permit packets that are denied by the second rule table.

Two firewall rule tables T_1 and T_2 are equivalent if $\mathcal{P}_{T_1} \subseteq \mathcal{P}_{T_2}$ and $\mathcal{P}_{T_2} \subseteq \mathcal{P}_{T_1}$. Lemma 1 states that the condition under which packets permitted by T_1 is a subset of packets permitted by T_2, that is, the $\mathcal{P}_{T_1} \subseteq \mathcal{P}_{T_2}$ condition.

Lemma 1: Let $T_1 = (R_1, R_2, \ldots, R_n)$. If all $R_i, i = 1, \ldots, n$ is either negative or has equivalent rule set in T_2, then $\mathcal{P}_{(R_1, R_2, \ldots, R_n)} \subseteq \mathcal{P}_{T_2}$.

Proof: The proof is by induction on n, the number of rules in T_1.

1. For the first rule R_1 in T_1 (situation when $n = 1$).

(a) Assume R_1 is positive.

As R_1 is the first positive rule, it is apparent that $\mathcal{P}_{R_1'} = \mathcal{P}_{R_1''} = \phi$, $\mathcal{P}_{R_1} = \mathcal{P}_{R_1'''}$. Because R_1 has equivalent rule set in T_2, according to the definition of equivalent rule set in Section 3, it can be concluded that $\mathcal{P}_{R_1'''} \subseteq \mathcal{P}_{T_2}$. This implies $\mathcal{P}_{R_1} \subseteq \mathcal{P}_{T_2}$ as $\mathcal{P}_{R_1} = \mathcal{P}_{R_1'''}$.

Table 4: Dividing R_5 into sub-rules

Sub-rule	Source IP address range	Destination IP address range	Service	Action
R'	192.168.13.1–20	192.168.13.0/24	HTTP	pass
	192.168.13.0/24	192.168.13.30–40	HTTP	pass
R''	192.168.13.20–30	192.168.13.0/24	HTTP	pass
	192.168.13.0/24	192.168.13.40–50	HTTP	pass
R'''	192.168.13.31–254	192.168.13.1–29,51–254	HTTP	pass

Table 5: Another example of rule table

Rule number	Source IP address range	Destination IP address range	Service	Action
1	192.168.13.31–254	192.168.13.1–29	HTTP	pass
2	10.1.1.0/24	10.1.2.0/24	HTTP	drop
3	192.168.13.31–100	192.168.13.51–254	HTTP	pass
4	10.1.2.0/24	10.1.1.0/24	SMTP	pass
5	10.1.2.0/24	10.1.1.0/24	HTTP	pass
6	192.168.13.101–254	192.168.13.51–254	HTTP	pass
7	any	any	any	drop

Therefore when $n = 1$, $\mathcal{P}_{(R_1)} \subseteq \mathcal{P}_{T_2}$.

(b) Assume R_1 is negative.

By definition, $\mathcal{P}_{(R_1)} = \phi$, which implies that $\mathcal{P}_{(R_1)} \subseteq \mathcal{P}_{T_2}$.

2. Assume that this theorem is true for $n - 1$ $(n \geq 1)$, that is, we assume that if $R_1, R_2, \ldots, R_{n-1}$ in T_1 all have equivalent rule set in T_2, then $\mathcal{P}_{(R_1,R_2,\ldots,R_{n-1})} \subseteq \mathcal{P}_{T_2}$.

Now we need to prove that the result is valid for n, that is, we need to prove that if $R_1, R_2, \ldots, R_n$ all have equivalent rule set in T_2, then $\mathcal{P}_{(R_1,R_2,\ldots,R_n)} \subseteq \mathcal{P}_{T_2}$.

If R_n is negative, then adding R_n to the rule table does not enlarge the set of packets that are allowed to pass, that is, $\mathcal{P}_{(R_1,R_2,\ldots,R_{n-1})} = \mathcal{P}_{(R_1,R_2,\ldots,R_n)}$. Thus, according to the induction assumption, $\mathcal{P}_{(R_1,R_2,\ldots,R_n)} \subseteq \mathcal{P}_{T_2}$.

If R_n is positive, then

(a) For $\forall p \in \mathcal{P}_{R'_n}$ or $\forall p \in \mathcal{P}_{R''_n}$, if it can pass rule table consisting of $(R_1, R_2, \ldots, R_n)$, then according to Fact 1 of R', R'' and R''', $\exists R'''_k$ $(1 \leq k \leq n-1)$ such that $p \in \mathcal{P}_{R'''_k}$. Because R_k has equivalent rule set in T_2 according to our assumption, then according to the definition of equivalent rule set, $\mathcal{P}_{R'''_k} \subseteq \mathcal{P}_{T_2}$, and hence p can also pass T_2.

(b) For $\forall p \in R'''_n$.

If R_n has equivalent rule set in T_2, then according to the definition of equivalent rule set, $\mathcal{P}_{R'''_n} \subseteq \mathcal{P}_{T_2}$, and hence p can also pass T_2.

Finally, combining (1) and (2), we know that the theorem holds for $\forall n \in N$. □

```
input  : rule $R_i$ — The positive rule whose effective part is to be computed
input  : array of rules $(R_1 \ldots R_{i-1})$ — The $i-1$ preceding rules in the same rule table
         as $R_i$
output: $R'''_i$    Effective part of $R_i$

set $R'_i = R''_i = R'''_i = \phi$
for each rule $R_j$ $(1 \leq j \leq i-1)$ do
    if $R_j$ is positive then
        $R'_i = R'_i \cup (R_j \cap R_i)$
    else
        $R''_i = R''_i \cup (R_j \cap R_i)$
set $R'''_i = R_i - R'_i - R''_i$
return $R'''_i$
```

Fig. 2 *Finding the effective part of a positive rule*

Theorem 1: Two firewall rule tables T_1 and T_2 are equivalent if (1) all positive rules in T_1 have equivalent rule sets in T_2; and (2) all positive rules in T_2 have equivalent rule sets in T_1.

Proof: Let $T_1 = (R_1, R_2, \ldots, R_n)$. Since all of $R_1, R_2, \ldots, R_n$ have equivalent rule sets in T_2, using Lemma 1 we have $\mathcal{P}_{T_1} \subseteq \mathcal{P}_{T_2}$.

Similarly, we can prove that $\mathcal{P}_{T_2} \subseteq \mathcal{P}_{T_1}$.

Hence, using the definition of equivalent firewall tables, we have that T_1 and T_2 are equivalent. □

Theorem 1 gives the condition under which two rule tables are equivalent. However, it is difficult to examine

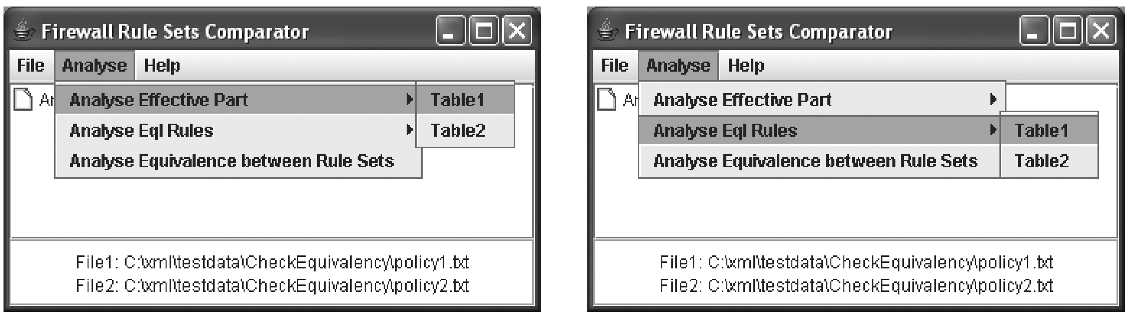
```
input  : Rule table $T_1 = r_1 \ldots r_{n_1}$, and $T_2 = R_1 \ldots R_{n_2}$
output: A boolean value indicating whether $T_1$ and $T_2$ are equivalent, and if not,
        difference between $T_1$ and $T_2$ and rules that are causing the difference

/* Boolean value indicating if $T_1$ and $T_2$ are equivalent                     */
set $equivalence = true$
/* The rules that permit packets denied by the other rule table              */
set $DiffRules_1 = DiffRules_2 = \phi$
/* The set of packets permitted by only one rule table                       */
set $Diff_1 = Diff_2 = \phi$

for each $r_i$ in $T_1$ do
    $r'''_i$ = FindEffectivePart( rule $r_i$, rule[] $r_1 \ldots r_{i-1}$ )
set $T'_1 = (r'''_1, r'''_2, \ldots, r'''_n)$
for each $R_i$ in $T_2$ do
    $R'''_i$ = FindEffectivePart( rule $R_i$, rule[] $R_1 \ldots R_{i-1}$ )
set $T'_2 = (R'''_1, R'''_2, \ldots, R'''_n)$
for each $r'''_i$ in $T'_1$ do
    for each $R'''_i$ in $T'_2$ do
        $r'''_i = r'''_i$ - $R'''_i$
    if $r'''_i \neq \phi$ then
        $equivalence = false$ ;
        $DiffRule_1$.insert($r_i$) ;
        $Diff_1 = Diff_1 \cup tempr'''_i$

for each $R'''_i$ in $T'_2$ do
    for each $r'''_i$ in $T'_1$ do
        $R'''_i = R'''_i$ - $r'''_i$
    if $R'''_i \neq \phi$ then
        $equivalence = false$ ;
        $DiffRule_2$.insert($R_i$) ;
        $Diff_2 = Diff_2 \cup tempR'''_i$
return $equivalence$, $DiffRules_1$, $DiffRules_2$, $Diff_1$, $Diff_2$
```

Fig. 3 *Check the equivalency of two rule tables T_1 and T_2*


Firewall Rule Sets Comparator
File
Analyse
Help
Analyse Effective Part
Analyse Eql Rules
Analyse Equivalence between Rule Sets
Table1
Table2

File1: C:\xml\testdata\CheckEquivalency\policy1.txt
File2: C:\xml\testdata\CheckEquivalency\policy2.txt

Fig. 4 *Prototype*

whether a rule has equivalent rule set in the second table because of implications discussed in Section 2.1.

Theorem 2: A firewall rule table T and its effective representation are equivalent

Proof: Consider the two rule tables $T_1 = (R_1, R_2, \dots, R_n)$ and $T_2 = (r_1, r_2, \dots, r_n)$, where $r_k = R_k'''$ if R_k is positive, otherwise $r_k = \phi$.

1. For each positive R_k, because $\mathcal{P}_{r_k'} = \mathcal{P}_{r_k''} = \phi$ (otherwise $\exists s < k$ such that $\mathcal{P}_{r_k} \cap \mathcal{P}_{r_s} \neq \phi$, that is, $\mathcal{P}_{R_k'''} \cap \mathcal{P}_{R_s'''} \neq \phi$. This cannot be true according to Fact 2 of R', R'' and R'''), it can be seen that $r_k = r_k'''$, and $R_k''' = r_k = r_k'''$. Therefore all positive rules in T_1 have equivalent rule set in T_2.
2. For each positive r_k, because $\mathcal{P}_{r_k'''} \subseteq \mathcal{P}_{r_k} = \mathcal{P}_{R_k'''}$, all positive rules in T_2 also have equivalent rule set in T_1. Thus, according to Theorem 1, T_1 and T_2 are equivalent. □

Theorem 2 can be used to compare two rule tables T_1 and T_2. To compare T_1 and T_2, we need to compare their effective representation T_1' with T_2'. Let $T_1 = (R_1, R_2, \dots, R_n)$ and $T_2 = (r_1, r_2, \dots, r_m)$. Then $T_1''' = (R_1''', R_2''', \dots, R_n''')$ and $T_2''' = (r_1''', r_2''', \dots, r_m''')$. Since $R_i''' \cap R_j''' = \varnothing$ for all i and j, for all rules in T_1, we need to determine if $\mathcal{P}_{R_i'''}$ is covered by a union of $\mathcal{P}_{r_j'''}$. That is if R_i''' has an equivalent rule set in T_2'. Similarly, for all rules in T_2, we need to determine if an equivalent rule set in T_1 exists.

4.1 Algorithms to compare firewall rule tables

In this section, we provide two algorithms where the first one determines the effective part of a rule, and the second one determines if two tables are equivalent and the rules that permit packets denied by the other table if inequality holds.

Fig. 2 finds the effective part of a rule R_i in the rule table $T = (R_1, \dots, R_n)$ where $n \geq i$. Fig. 2 finds R_i' and R_i'' and in the last step obtains R_i''' from R_i' and R_i''. The figure examines every R_j $(1 \leq j \leq i-1)$ and accumulates intersections $R_i \cap R_j$ $(1 \leq j \leq i-1)$ in R_i', and hence the part of R_i which is redundant to at least one preceding positive rule is put into R_i'. According to Definition 2, the resulting R_i' will be the redundant part of R_i. Similarly, the resulting R_i'' will be the shadowed part of R_i. Therefore R_i''' will the effective part of R_i. Time complexity required for is $O(n) \times \beta$, where β denotes time complexity required for computation of $R_j \cap R_i$, which can be reduced to the problem of two-dimensional rectangle intersection. A number of methods for calculation of d-dimensional rectangle intersection is summarised in [10], the best time-complexity for the two-dimensional case being $O(N \log N)$.

The algorithm in Fig. 3 determines if two rule tables T_1 and T_2 are equivalent. First, using Fig. 2, the rules in T_1 and T_2 are converted into their corresponding effective forms. Let $T_1 = (r_1, r_2, \dots, r_n)$ and $T_2 = (R_1, R_2, \dots, R_n)$. We first construct $T_1' = (r_1''', r_2''', \dots, r_n''')$ and $T_2' = (R_1''',$

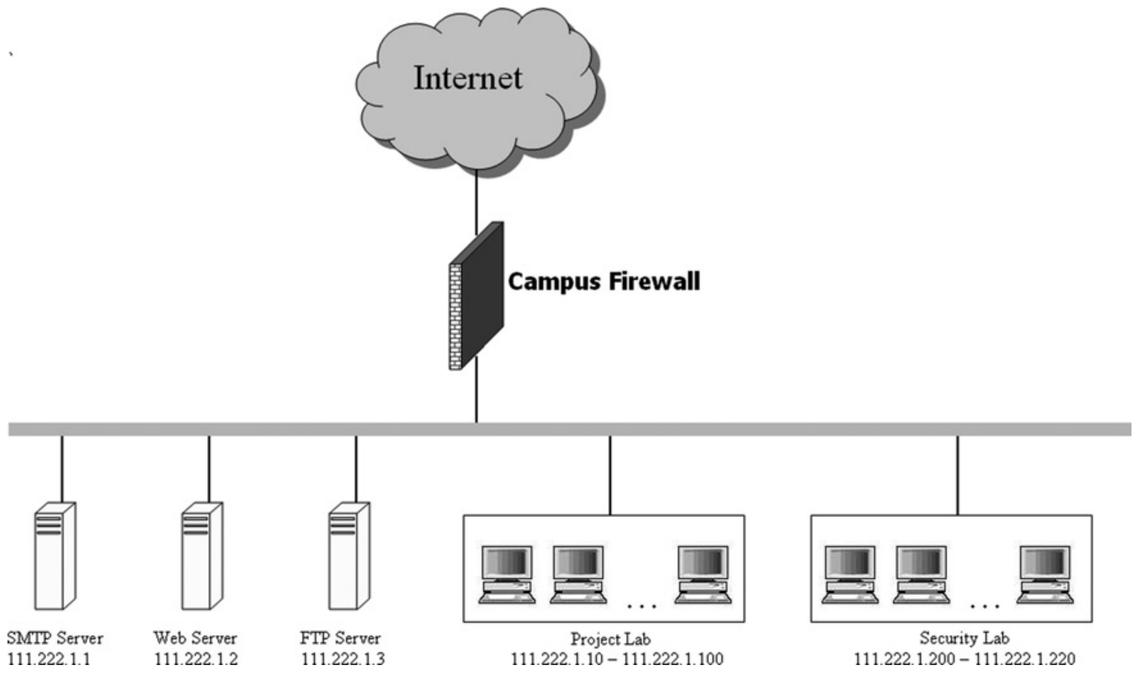


Fig. 5 *Example of network setup*

Table 6: Security policy

Policy description
external hosts can only access the servers
external hosts cannot access any machine in either lab
hosts in the project lab can access everything except the security lab
hosts in the security lab can only access the web server and other hosts in the security lab

R_2'', . . ., R_n''). Then for each rule r_i'' in T_1', we examine if it has an equivalent rule set in T_2'. This step is repeated for all rules in T_2' and if both steps produce a true results, we conclude the two tables are equivalent. Correctness of this conclusion follows directly from Theorem 1 and 2. Time complexity in relation to the number of rules required for the algorithm in Fig. 3 is $O(n^2)$.

If T_1 and T_2 are not equivalent, then the algorithm in Fig. 3 returns the sets of packets permitted only by T_1 or T_2 but not both, and also rules that permit these packets.

5 Implementation and a complete example

5.1 Implemenation

We implemented a GUI prototype of the techniques presented above in Java, which is illustrated in Fig. 4. It can be seen that the prototype can be used to analyse the effective parts of rules in a rule table, equivalent rule set in another rule table, and whether two rule tables are equivalent. We will explain in details how this prototype can be used to compare rule tables and debug firewall configurations with a complete example presented in the following section.

5.2 Complete example

Consider the campus network configuration illustrated in Fig. 5. The network consists of three servers accessible from outside, and two student labs – the project lab and the security lab. The project lab provides a platform for students to carry out daily activities such as browsing Internet or writing assignments. The security lab is for students studying network security to perform experiments such as sniffing or packet spoofing. It is hence desirable to separate traffic of the security lab from that of the project lab.

The security policy is described in Table 6. Description of the policy might be ambiguous and not clear at this stage. This is often an important reason leading to incorrect translation of network policy into firewall rules. We will demonstrate later how the prototype can be used to clarify the ambiguity, and to elicit unstated requirement in policy description.

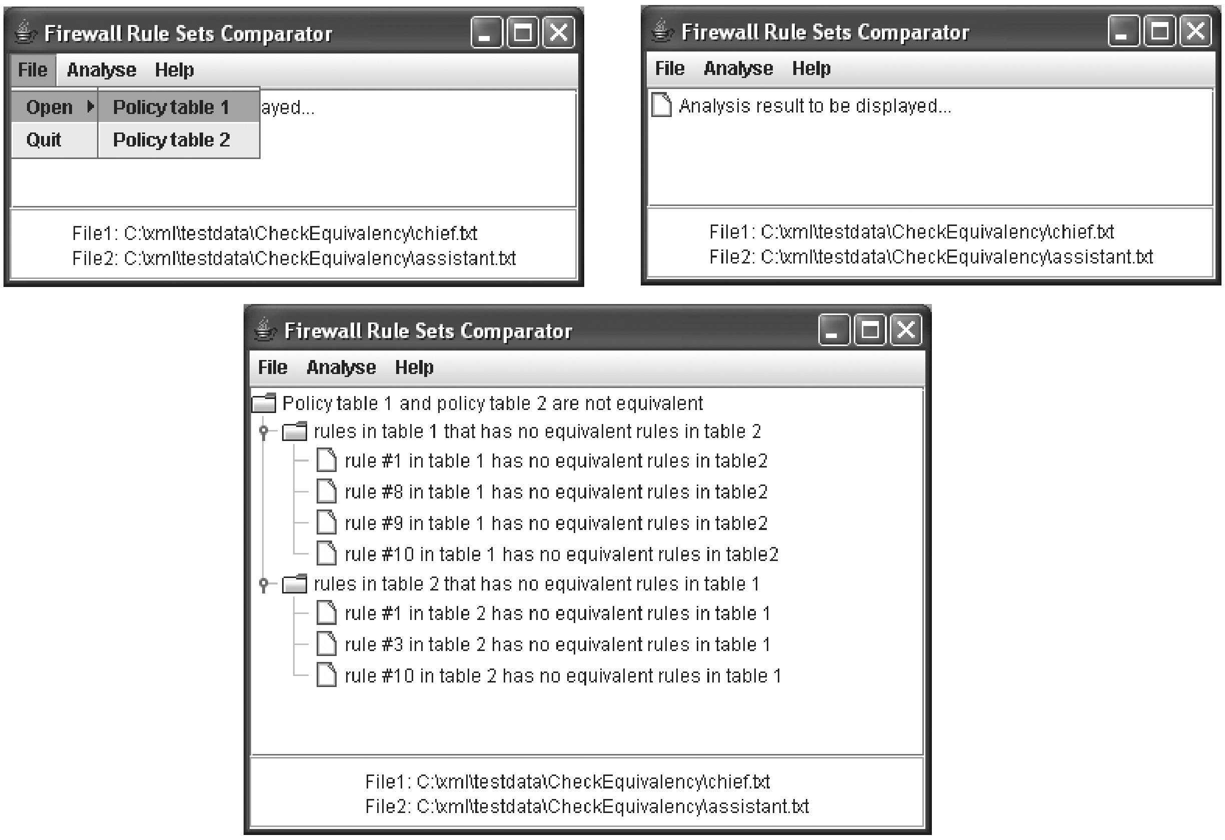
Assume that there are a chief administrator and an associate administrator in the network department. The chief administrator has more experience and follows 'who can access what' to translate the security policy into firewall rule table. On the other hand, the associate administrator

Table 7: Rule table by the chief administrator

Rule number	Source IP address range	Destination IP address range	Service	Action
1	*	*	DNS	pass
2	111.222.1.200–111.222.1.220	111.222.1.1	SMTP	deny
3	*	111.222.1.1	SMTP	pass
4	*	111.222.1.2	HTTP	pass
5	111.222.1.200–111.222.1.220	111.222.1.3	FTP	deny
6	*	111.222.1.3	FTP	pass
7	111.222.1.200–111.222.1.220	111.222.1.10–111.222.1.100	*	deny
8	111.222.1–111.222.1.3	111.222.1.10–111.222.1.100	*	pass
9	111.222.1.10–111.222.1.100	111.222.1.10–111.222.1.100	*	pass
10	111.222.1.200–111.222.1.220	111.222.1.200–111.222.1.220	*	pass

Table 8: Rule table by the assistant administrator

Rule number	Source IP address range	Destination IP address range	Service	Action
1	*	111.222.1.1	SMTP	pass
2	*	111.222.1.2	HTTP	pass
3	*	111.222.1.3	FTP	pass
4	*	111.222.1.10–111.222.1.100	*	deny
5	*	111.222.1.200–111.222.1.220	*	deny
6	111.222.1.10–111.222.1.100	111.222.1.1	SMTP	pass
7	111.222.1.10–111.222.1.100	111.222.1.2	HTTP	pass
8	111.222.1.10–111.222.1.100	111.222.1.3	FTP	pass
9	111.222.1.10–111.222.1.100	111.222.1.200–111.222.1.220	*	deny
10	111.222.1.10–111.222.1.100	*	*	pass
11	111.222.1.200–111.222.1.220	111.222.1.200–111.222.1.220	*	deny
12	111.222.1.200–111.222.1.220	111.222.1.2	HTTP	deny



```
ayed...

File1: C:\xml\testdata\CheckEquivalency\chief.txt
File2: C:\xml\testdata\CheckEquivalency\assistant.txt
```

```
Analysis result to be displayed...

File1: C:\xml\testdata\CheckEquivalency\chief.txt
File2: C:\xml\testdata\CheckEquivalency\assistant.txt
```

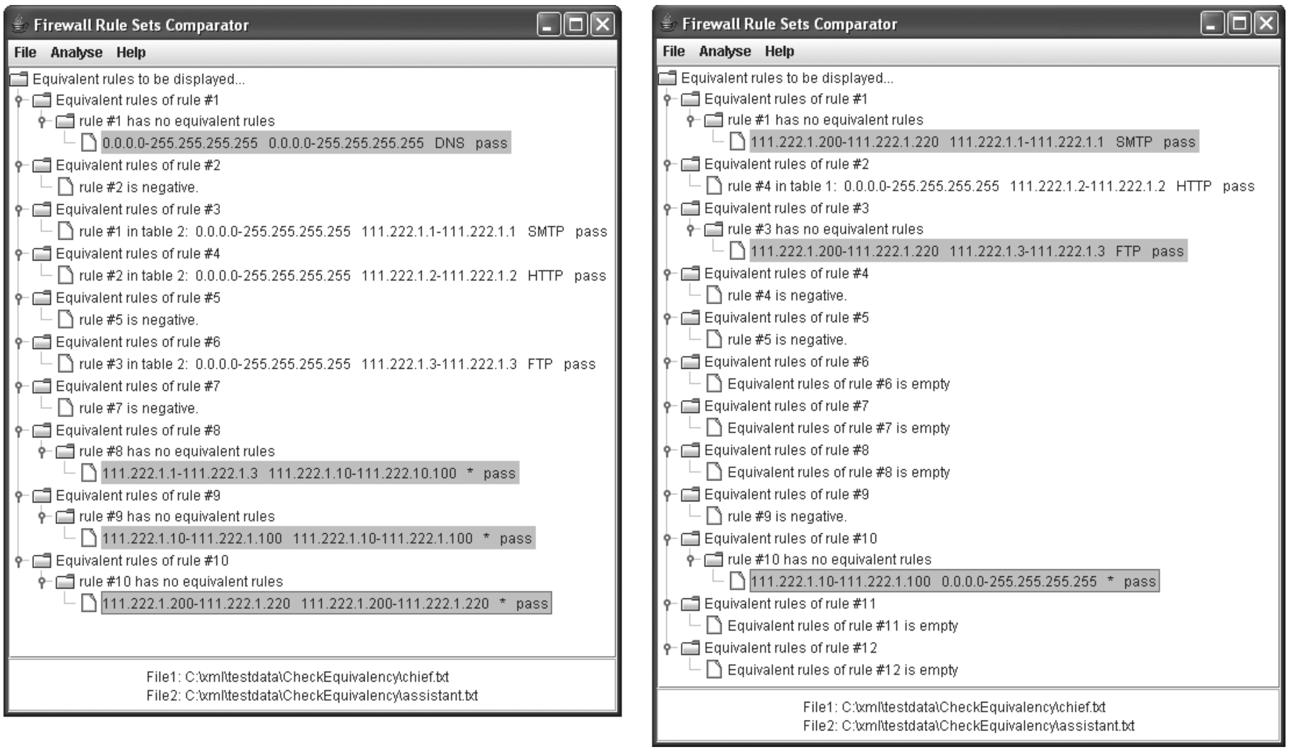
```
Policy table 1 and policy table 2 are not equivalent
  rules in table 1 that has no equivalent rules in table 2
    rule #1 in table 1 has no equivalent rules in table2
    rule #8 in table 1 has no equivalent rules in table2
    rule #9 in table 1 has no equivalent rules in table2
    rule #10 in table 1 has no equivalent rules in table2
  rules in table 2 that has no equivalent rules in table 1
    rule #1 in table 2 has no equivalent rules in table 1
    rule #3 in table 2 has no equivalent rules in table 1
    rule #10 in table 2 has no equivalent rules in table 1

File1: C:\xml\testdata\CheckEquivalency\chief.txt
File2: C:\xml\testdata\CheckEquivalency\assistant.txt
```

Fig. 6 *Rule table compare result*

has less experience and sequentially translates each item in the policy description into corresponding firewall rules. The resulting firewall rule tables are listed in Table 7 and 8, respectively.

Now we need to compare the rule tables to gain a confidence in their correctness, as well as to explore possible ambiguity in policy description and incorrect translation of the policy. To compare the rule tables, we first load them into the prototype, then select 'Analyse' ⇒ 'Analyse Equivalence between Rule Tables' from the menu, as illustrated in Fig. 6. The result shows that the rule tables are not equivalent, because rules 1 and 3 in rule Table 1 and rules 8–10 in rule Table 2 have no equivalent rule set in the other rule table.



```
Equivalent rules to be displayed...
  Equivalent rules of rule #1
    rule #1 has no equivalent rules
      0.0.0.0-255.255.255.255  0.0.0.0-255.255.255.255  DNS  pass
  Equivalent rules of rule #2
    rule #2 is negative.
  Equivalent rules of rule #3
    rule #1 in table 2:  0.0.0.0-255.255.255.255  111.222.1.1-111.222.1.1  SMTP  pass
  Equivalent rules of rule #4
    rule #2 in table 2:  0.0.0.0-255.255.255.255  111.222.1.2-111.222.1.2  HTTP  pass
  Equivalent rules of rule #5
    rule #5 is negative.
  Equivalent rules of rule #6
    rule #3 in table 2:  0.0.0.0-255.255.255.255  111.222.1.3-111.222.1.3  FTP  pass
  Equivalent rules of rule #7
    rule #7 is negative.
  Equivalent rules of rule #8
    rule #8 has no equivalent rules
      111.222.1.1-111.222.1.3  111.222.1.10-111.222.10.100  *  pass
  Equivalent rules of rule #9
    rule #9 has no equivalent rules
      111.222.1.10-111.222.1.100  111.222.1.10-111.222.1.100  *  pass
  Equivalent rules of rule #10
    rule #10 has no equivalent rules
      111.222.1.200-111.222.1.220  111.222.1.200-111.222.1.220  *  pass

File1: C:\xml\testdata\CheckEquivalency\chief.txt
File2: C:\xml\testdata\CheckEquivalency\assistant.txt
```

```
Equivalent rules to be displayed...
  Equivalent rules of rule #1
    rule #1 has no equivalent rules
      111.222.1.200-111.222.1.220  111.222.1.1-111.222.1.1  SMTP  pass
  Equivalent rules of rule #2
    rule #4 in table 1:  0.0.0.0-255.255.255.255  111.222.1.2-111.222.1.2  HTTP  pass
  Equivalent rules of rule #3
    rule #3 has no equivalent rules
      111.222.1.200-111.222.1.220  111.222.1.3-111.222.1.3  FTP  pass
  Equivalent rules of rule #4
    rule #4 is negative.
  Equivalent rules of rule #5
    rule #5 is negative.
  Equivalent rules of rule #6
    Equivalent rules of rule #6 is empty
  Equivalent rules of rule #7
    Equivalent rules of rule #7 is empty
  Equivalent rules of rule #8
    Equivalent rules of rule #8 is empty
  Equivalent rules of rule #9
    rule #9 is negative.
  Equivalent rules of rule #10
    rule #10 has no equivalent rules
      111.222.1.10-111.222.1.100  0.0.0.0-255.255.255.255  *  pass
  Equivalent rules of rule #11
    Equivalent rules of rule #11 is empty
  Equivalent rules of rule #12
    Equivalent rules of rule #12 is empty

File1: C:\xml\testdata\CheckEquivalency\chief.txt
File2: C:\xml\testdata\CheckEquivalency\assistant.txt
```

Fig. 7 *Rule table compare result*

We then select 'Analyse' ⇒ 'Analyse Eql Rules' from the menu to analyse the packets applied to and permitted only by these rules but not by the other rule table. The prototype shows us, for each of the above rules, the set of packets that cannot pass the other rule table, as illustrated in Fig. 7. By a close look at the rule tables and the results in Fig. 7, we revealed a number of problems in the policy description as well as its translation into firewall rules.

1. The policy does not explicitly grant the use of DNS, which in practise is a pre-requisite for using HTTP and FTP. The permission to use DNS should be added explicitly to the policy description to reduce ambiguity.
2. The policy does not mention which hosts the servers can access. The chief administrator grants them access to the project lab, but the associate administrator does not. This conflict should be discussed further and addressed with department managers.
3. The associate network administrator translates each item in the policy description into firewall rules in order. By a mistake in representing 'external hosts' with '*', the negative rules that prohibit external hosts to access the servers also prohibit hosts in both student labs to access themselves. By a similar mistake, he also permits hosts in the security lab to access the SMTP and FTP server.
4. The statement 'hosts in the project lab can access everything' in policy description is ambiguous, as 'everything' can be taken as 'everything in the internal network' or 'everything including both internal and external network'. It can be seen that the chief administrator is more rigorous and take it as 'everything in the internal network', whereas the associate administrator takes the other understanding.

Counter-measures against these problems can be taken effectively after they are revealed. The prototype can therefore assist in clarifying policy ambiguity and debugging firewall configurations by accurately comparing firewall rule tables and locating rules that cause inequality.

6 Conclusions

In this paper, we presented an effective technique for comparing firewall rule tables. Our technique can not only determine whether rule tables represented in different forms are equivalent, but also accurately locate the rules in their original form and order that are causing the inequality. Multiple rule tables written by different administrators to implement the same security policy can be compared to gain an increased level of confidence in their correctness when the rule tables are equivalent; otherwise the rules causing conflicts between rule tables can be accurately located, as can assist in resolving conflicts between rule tables. Our technique can also be used to analyse changes to a rule table, and determine whether desired changes are made correctly by comparing the original rule table and the modified one. We also implemented our technique with a GUI prototype written in Java, and demonstrated a complete example of using the prototype to compare and debug firewall configurations.

7 References

1 Cheswick, W.R., and Bellovin, S.M.: 'Firewalls and internet security, repelling the Wily Hacker' (Addison-Wesley, 1994)
2 Bartal, Y., Mayer, A., Nissim, K., and Wool, A.: 'Firmato: a novel firewallmanagement toolkit'. IEEE Symp. Security and Privacy, 1999
3 Lee, T.K., Yusuf, S., Luk, W., Sloman, M., Lupu, E., and Dulay, N.: 'Compiling policy descriptions into reconfigurable firewall processors'. Systems, Man and Cybernetics, IEEE Int. Conf., 2003
4 Damianou, N., Dulay, N., Lupu, E., and Sloman, M.: 'The ponder policy specification language'. Workshop on Policies for Distributed Systems and Networks, LNCS, Springer, 2001, vol. 1995, pp. 18–38
5 Hazelhurst, S., Attar, A., and Sinnappan, R.: 'Algorithms for improving the dependability of firewall and filter rule lists'. Int. Conf. Dependable Systems and Networks (DSN 2000), 2000
6 Hazehurst, S.: 'Algorithms for analyzing firewall, and router access lists'. Technical report trwitscs-1999, Department of Computer Science, University of the Witwatersrand, South Africa, 1999
7 Al-Shaer, E.S., and Hamed, H.H.: 'Firewall policy advisor for anomaly discovery and rule editing'. IEEE/IFIP 8th Int. Symp. Integrated Network Management, 2004
8 Al-Shaer, E.S., and Hamed, H.H.: 'Discovery of policy anomalies in distributed firewall'. IEEE INFOCOM, Twenty-third Annual Joint Conf. IEEE Computer and Communications Societies, March 2004, vol. 4, pp. 2605–2616
9 Strassberg, K., Gondek, R., and Rollie, G.: 'Firewalls: the complete reference' (McGraw-Hill/Osborne, 2002), pp. 84–88
10 Petty, M.D., and Mukherjee, A.: 'Experimental comparison of d-rectangle intersection algorithms applied to HLA data distribution'. Proc. 1997 Distributed Simulation Symp, Orlando, FL, September 1997, pp. 13–26