World Applied Sciences Journal 7 (Special Issue of Computer & IT): 188-198, 2009
ISSN 1818.4952

Specification, Analysis and Resolution of Anomalies in Firewall Security Policies

Mohsen Rezvani and Ramtin Aryan

Department of Information Technology and Computer Engineering,
Shahrood University of Technology, Shahrood, Iran

Abstract: Firewalls are essential components in network security solutions. Managers have to specify their organizational security policies using low level and order-dependent rules in firewalls. Furthermore, dependency of firewalls to the network topology, frequent changes in network topology and lack of an automatic method for analysis and verification of anomalies in specified security policy lead to inconsistencies and security holes. In this paper we present a formal language for specification of security policy in firewalls. Based on the language, the specified security policy, simple anomalies and total anomalies are translated to propositional logic formulas. Furthermore we have designed and implemented a tool based on theorem proving for detection of the anomalies in the specified policy. In addition, based on the formal model, two algorithms are presented for resolving anomalies in the policy. These algorithms minimize the number of rules without changing the security policy.

Key words: Anomaly . firewall . rule database . resolving anomaly . security policy . total anomaly

INTRODUCTION

Firewalls are the main security devices for securing computer networks. Although using firewalls is a useful method to satisfy security policy of an organization, precise specification of security requirements is vital for accuracy of firewall operations. This may occur due to the notations and the language which is used to define the requirements; which is currently based on an ordered set of rules. Experiments show that the complexity of working with rules, due to their low level of abstraction and conflicts between them, results in miss-configuration of firewalls and potentially security holes for the organization [1].

In the most cases, a firewall administrator should consider the anomalies of the rule database. As rule sets become larger and more complex, administrators cannot prevent all of the anomalies [2]. An automatic analysis of anomalies in a firewall rule database is necessary for improving the semantics of security policy and also discovering the extra rules in the database and finally optimizing the performance of policy matching in firewalls.

This paper presents a security model with an inherent compatibility with overall design of most firewalls. Applying the security model does not force any change in the firewall design. A formal language is designed to express security requirements. Due to common applications of Deontic logic in security models [3], it is used as a basis for specifying the syntax and semantics of the language. Security policy is specified using Deontic formulas.

On the other hands, for discovering and resolving the anomalies in the specified security policy, we present a formal model based on propositional logic. Thus we translate each Deontic formal in specification language to a propositional logic formula. In addition, we present a formal definition of simple and total anomalies in the security policy. The simple anomalies are defined between two security propositions and total anomalies can be defined between one proposition and a set of propositions [4].

For discovering anomalies in a specified security policy, we present a state machine, two detection algorithms and also a tool based on Binary Decision Diagram (BDD) for automatic verification of the detection process. This tool is a proof assistant for propositional logic used for satisfying anomaly theorems based on our formal model.

For Resolving anomalies in a specified security policy, we offer two algorithms. These algorithms lead to decrease the number of rules without changing the policy.

The rest of this paper is organized as follows. Section 0 reviews related works. In section 0, our security policy specification language is presented. Section 0 describes our formal model for specifying anomalies. Section 0 presents the state machine and algorithms for anomaly discovery and resolution. Section 0 presents conclusions and our plans for future work.

Corresponding Author: Mohsen Rezvani, Department of Information Technology and Computer Engineering,
Shahrood University of Technology, Shahrood, Iran

RELATED WORK

We categorized the existing security policy specification methods in [5]. In [6], the security policy is specified formally based on the information flow. A security policy consists of a set of information classes and constraints on flow of information. The constraints are specified by a specific type of logic called branching time temporal logic [7].

In [8], a security policy is specified as a specific case of a regulation. The system to be regulated consists of agents which can execute actions on some objects. Each role is associated with a set of norms (permissions, obligations and prohibitions). An agent can play one or more roles. In this approach, regulation is specified using a logic based on SDL (Standard Deontic Logic).

LaSCO (the Language for Security Constraints on Objects) [9] is a language for specifying policy as a directed graph. The semantics of the language was represented by a first ordered logic.

There are numerous studies on conflict detection and resolution in firewall security policy. This problem can be raised on router, VPN and firewall configuration. We proposed formal modeling approach to specify and verify two specific types of conflict in firewall security policy included covering and consistency [5].

A famous classification of anomalies in firewall rule database is proposed in [2]. They presented four category of anomalies contained shadow, correlation, generalization and redundant. They used a set theory based model for defining the anomalies. All the anomalies are defined between two rules and not cover two set of rules.

A model based on relational algebra and raining 2D-Box presented in [10]. Using complex operators in relation algebra can increase the time complexity of detection algorithm. The main parameter for anomaly detection in [11] is the order of rules in database. In this research, a technique for resolving anomalies is proposed. A high level language for specifying security policy in firewall based on expert systems is presented in [12]. Anomaly detection in [13, 14] has proposed based on set theory and regular expressions. In this work, total anomalies are not considered.

Authors in [15-17] presented different classification for possible policy conflicts. For example, classification for anomalies in [15] is based on Coq [18]. Total anomalies are not considered in these works.

A logical framework for anomaly detection presented in [19]. In [20] an ACLA framework for detection and resolution of anomalies proposed. A decision tree based solution presented in [21] for analyzing firewall anomalies. FIREMAN tool is developed for static analysis of conflicts in firewall [18]. This tool is designed based on binary decision diagrams. Total anomalies are not considered in this tool. In [22] a scheme for conflict resolution is presented based on the idea of adding resolve filters.

Firmato and Fang [23, 24] are a set of firewall management and analysis tools that interact with administrator on queries about firewall rules. The tools specify security policy in stateful packet filtering independent of network topology.

SECURITY POLICY SPECIFICATION LANGUAGE

Main requirements in selecting a method for security policy specification in firewalls are:

- Separating the security policy from the network topology.
- A high level language for defining security policy considering all functionalities of all firewalls.
- Automatic generation of firewall rule bases using the specified security policy.
- A method for verification of security policy.

Based on the above requirements, we propose a method for security policy specification in firewalls. A language based on the Deontic logic is the core of our method. The language supports separation of the security policy from the network topology as well as automatic generation of an existing firewall rules.

Syntax of the language: The syntax of our proposed security policy specification language covers two parts:

- Security policy specification
- Network topology specification

At first, security policy is specified regardless of network topology and then network topology is specified. A security policy in firewalls defined as a hexad (R, Op, S, T, P, A) such that:

- R is a finite set of roles.
- Op is a set of relations on roles. For example, inheritance is a relation. If role x inherits from role y, then all hosts playing role x will do play role x.
- S is a finite set of services in network.
- T is a finite set of time periods.
- P is a finite set of primitive propositions. The propositions specify connections. On the other hand, each proposition introduces a set of connections between two roles. For example, a proposition can be $x \xrightarrow{s_t} y$; where x and y are roles, s is a service and t is a time period. This proposition represents all connections from role x to role y with service s at time period t. Operator $\rightarrow$ specifies direction of connections. Other operators for this concept are $\leftarrow$ and $\leftrightarrow$.
- A is a sequence of security propositions; each proposition is a Deontic logic formula. The propositions are main part of security policy specifications.

We presented more operators for supporting masquerading, content security, authentication and logging in the formal language [5].

Network topology is specified as a set of tuples $\langle r, I \rangle$ where:

- r is an element from R; defined in the security policy specification.
- I is a range of IP addresses. Those hosts with IP addresses in I, play role r.

Semantics of the language: To simplify the language semantics, we assume the functionality of the firewall is restricted to a stateful packet filter, saving the state of all active connections. Such a firewall can simply be modeled as a state transition machine. Variables corresponding to the states are vectors saving information of active connections. Each connection can be represented by a state transition machine [25].

An active connection C can be specified as a triple $(\Sigma_C, \tau_C, I_{0C})$, where:

- Σ_C is a finite set of states for a connection
- τ_C is a transition relation $\Sigma_C \rightarrow \Sigma_C$
- I_{0C} is an initial state of a connection

A firewall can be specified as a composition of active connections. Accordingly, the firewall F is a state transition machine $(\Sigma_F, \tau_F, I_{0F})$ such that:

$$F = \| C_i \qquad (1 \le i \le n)$$

$$\Sigma_F = \prod \Sigma_{C_i} \qquad (1 \le i \le n)$$

$$\tau_F = \bigcup \tau_{C_i} \qquad (1 \le i \le n)$$

$$I_{0F} = \bigcup I_{0C_i} \qquad (1 \le i \le n)$$

As security policy in firewalls is specified at the lowest level based on connections, semantics of the language can use only the first level of the state transition machine. To represent the semantics, first we define the matching function ρ, followed by defining how to apply security policy via a single state. Finally applying security policy by a state transition machine is defined.

Assume that the set of truth-values is B = {Accept, Reject, NoMatch}. Function π maps a primitive proposition to the set of its matching connections. The matching function ρ returns result of matching a connection with a security proposition. Matching a connection c with security proposition π denoted as $\rho(c, \phi, \pi)$ and is defined as:

- If $\phi = pX$ and $c \in \pi(X)$, then the result is Accept, otherwise is NoMatch.
- If $\phi = FX$ and $c \in \pi(X)$, then the result is Reject, otherwise is NoMatch.

A state s of the state transition machine $(\Sigma_F, \tau_F, I_{0F})$ satisfies security policy SP, if and only if for each connection c in s, there exists a security proposition π in SP such that $\rho(c, \phi, \pi)$ = Accept and there not exist any security proposition π in SP such that $\rho(c, \phi, \pi)$ = Reject. A state transition machine $(\Sigma_F, \tau_F, I_{0F})$ satisfies SP, if and only if all states in the machine satisfy the policy.

The semantics described so far can be extended to cover all features provided by the language including application gateways, time and NAT. We used the semantics to show that specifications accepted by the language are applicable and consistent [5].

FORMAL SPECIFICATION OF ANOMALIES

Based on the syntax of our language, security propositions are translated into propositional logic formulas. Then a theorem is defined for each anomaly in the formulas. Each theorem is also a propositional logic formula. For proving the existence of each anomaly in policy, we prove validity of the appropriate theorem using a proof assistant.

Reference [2] defines anomalies in four categories that include Shadowing, Correlation, Generalization and Redundancy. It classifies different anomalies that may exist between two filtering rules in one firewall and then describes a technique for discovering these anomalies.

In this section we represent the anomalies defined in [2] based on our model for specifying security policy. Also we define new anomalies that may exist among more than two security propositions. These anomalies may happen between two subset of security proposition set in firewalls. Thus we defined total version of anomalies defined in [2].

Translation of security propositions: Consider the complication of automatic proof in Deontic logic formulas [26]; we translate security propositions into propositional logic formulas. The result of this translation is used for generating theorems.

Considering the security policy for a stateful packet filtering firewall, the key components in security propositions are contained roles, service, direction and action.

In order to translate a security proposition to a logical formula, we must translate each attribute of it and then conjunct the translated parts. Since the attributes of a security proposition have different meanings, translation needs different conjunction operators.

The idea is representing numbers as a bit sequence. For example, a port number in TCP protocol is a number between 0 and 65535; just a sequence of bits. A number of Boolean variables and expressions are introduced to represent the information in security propositions. Each field is assigned to a number between 0 and n-1. The number can be represented in $m = \log_2^n$ bits and so we introduce m variables $v_0 \ldots v_m$ to encode the field.

According to the syntax of the proposed language, there are two roles in the security proposition. Each role is assigned to a set of IP addresses. Each IP address includes 32 bits. We introduce 32 variables of the form $s_0 \ldots s_{31}$ for the first role (remind the first role) and 32 variables of the form $d_0 \ldots d_{31}$ for the second role (remind it). A range of addresses can be translated using the disjunction operator.

For example, when the IP address of the first role is 192.168.20.1, it can be represented in the binary form 11000000.10101000.00010100.00000001 and can be translated in the following form.

$$s_{31}s_{30}s'_{29}s'_{28}s'_{27}s'_{26}s'_{25}s'_{24}s_{23}s'_{22}s_{21}s'_{20}s_{19}s'_{18}s'_{17}s'_{16}$$
$$s'_{15}s'_{14}s'_{13}s_{12}s'_{11}s_{10}s'_{9}s'_{8}s'_{7}s'_{6}s'_{5}s'_{4}s'_{3}s'_{2}s'_{1}s_{0}$$

```
1. Translate s into s[0] … s[31]
2. Translate m into m[0] … m[31]
3. result = true
4. for  i = 0  to 31 do
5.     if m[i] =1  then
6.       Result = result and s[i]
7.     if m[i] =0 and s[i] =1 then
8.       Result =false; return
9. end of for
```

Algorithm 1: Translating an IP address with mask

In many cases, the address range is defined using masks. The Algorithm 1 is used for translating the range address s [0…31] with mask m [0…31].

The service component contains the protocol type and the port number. Protocol type can be either TCP or UDP. One variable named l is introduced for translating the protocol type. Thus the values of TCP and UDP are represented with l and l′. Port numbers can be specified using 16 Boolean variables of the form $p_0,\ldots,p_{15}$.

The direction component is translated into a single bit presented by the Boolean variable c. The component has three values and is translated as follows:

- For value $\rightarrow$, this is translated to c.
- For value $\leftarrow$, this is translated to c.
- For value $\leftrightarrow$, this is translated to TRUE.

Using the described method, the security proposition can be presented by a Boolean expression. The expression is defined by conjunction of its components. Equation (1) shows the general form in translation of a security proposition in firewalls.

$$\text{Condition} \Rightarrow \text{Action} \qquad (1)$$

where C is the condition part and A is the action part of a translated security proposition. The condition part is defined as conjunct Boolean expressions of translating roles, service and direction of the security proposition. Therefore we can represent the last equation in form of equation (2). In this equation, each X_i specifies an attribute in the security proposition.

$$C_{X_1} \wedge C_{X_2} \wedge \ldots \wedge C_{X_n} \Rightarrow A \qquad (2)$$

In the simplified case of a packet filtering firewall, a bit named g in introduced for specifying the action part. For a security proposition in the form PX, action part will be considered as g and for a security proposition in the form FX, the action part will be considered as g′.

The following sub-sections describe using of this method for defining the theorems and verifying anomalies in security policy.

Shadow anomaly: Proposition P_1 is shadowed by previous proposition P_2 if proposition P_2 matches all the connections that match the proposition P_1 and the two propositions have different filtering actions [2].

Formally, Proposition P_1 is shadowed by proposition P_2 if the following condition holds:

$$\begin{aligned} &P_1 : C_1 \Rightarrow A_1 \\ &P_2 : C_2 \Rightarrow A_2 \end{aligned} \tag{3}$$

$$\exists P_1, P_2 \in SP \bullet \neg(A_1 \Leftrightarrow A_2) \wedge (C_1 \Rightarrow C_2)$$

Equation (3) shows that we must verify the correctness of the equation for each two propositions in the security policy (SP). In this equation, P_1 and P_2 are two different proposition and SP is security policy.

Total shadow anomaly: Proposition P is totally shadowed by a set of previous propositions if the previous propositions match all the connections that match the proposition P_1 and the proposition P_1 has different filtering action rather than the previous propositions [4].

Formally, proposition P_n is shadowed by propositions P_1 P_k if the following condition holds:

$$\begin{aligned} &P_1 : C_1 \Rightarrow A_1 \\ &P_2 : C_2 \Rightarrow A_2 \\ &\vdots \\ &P_n : C_n \Rightarrow A_n \end{aligned} \tag{4}$$

$$\begin{aligned} \exists P_1, P_2, \ldots, P_k, P_n \in SP \bullet \neg(A_n \Leftrightarrow A_1) \\ \wedge (A_1 \Leftrightarrow A_2) \wedge \ldots \wedge (A_{k-1} \Leftrightarrow A_k) \\ \wedge \left[C_n \Rightarrow (C_1 \vee C_2 \ldots \vee C_k)\right] \end{aligned}$$

In the equation (4), propositions $P_1 \ldots P_k$ are in the upper order than proposition P_n. Also proposition P_n is totally shadowed by propositions $P_1 \ldots P_k$ if the correctness of equation (4) is proved.

Correlation anomaly: Two security propositions in the security policy are correlated if they have different filtering actions and the first proposition matches some connections that match the second proposition and also the second proposition matches some connections that match the first proposition [2].

Formally, proposition P_1 and proposition P_2 have a correlation anomaly if the following condition holds:

$$\begin{aligned} &P_1 : C_1 \Rightarrow A_1 \\ &P_2 : C_2 \Rightarrow A_2 \end{aligned} \tag{5}$$

$$\begin{aligned} \exists P_1, P_2 \in SP \bullet \neg(A_1 \Leftrightarrow A_2) \\ \wedge \left[\neg(C_1 \Rightarrow C_2) \wedge \neg(C_2 \Rightarrow C_1) \wedge (C_1 \vee C_2)\right] \end{aligned}$$

By using the equation (5), we could find all correlation anomalies between propositions in security policy. In [2], set theory is used for formal definition of anomalies and it couldn't detect the correlation anomaly in some special cases.

For correlation anomaly, it is not required to define total anomaly, because all cases for correlation anomaly can be detected by definition of the anomaly between two rules.

Generalization anomaly: Proposition P_1 is a generalization of a preceding proposition P_2 if they have different actions and if the Proposition P_2 can match all the packets that match the proposition P_1 [2].

Formally, proposition P_1 is a generalization of proposition P_2 if the following condition holds:

$$\begin{aligned} &P_1 : C_1 \Rightarrow A_1 \\ &P_2 : C_2 \Rightarrow A_2 \end{aligned} \tag{6}$$

$$\exists P_1, P_2 \in SP \bullet \neg(A_1 \Leftrightarrow A_2) \wedge (C_1 \Rightarrow C_2)$$

Equation (6) shows that we must verify the correctness of the equation for each two propositions in the security policy. Generalization is often used to exclude a specific part of the traffic from a general security proposition therefore, it is only considered an anomaly warning and we can't remove the anomaly from the security policy.

Total generalization anomaly: Proposition P_1 is a total generalization of a set of further propositions if the further propositions match all the connections that match the proposition P_1 and the proposition P_1 has different filtering action rather than the further propositions [4].

Formally, proposition P_n is total generalization of a set of propositions $P_1 \ldots P_k$ if the following condition holds:

$$\begin{aligned} &P_1 : C_1 \Rightarrow A_1 \\ &P_2 : C_2 \Rightarrow A_2 \\ &\vdots \\ &P_n : C_n \Rightarrow A_n \end{aligned} \tag{7}$$

$$\exists P_1, P_2, \ldots, P_k, P_n \in SP \bullet \neg(A_n \Leftrightarrow A_1) \wedge (A_1 \Leftrightarrow A_2) \wedge \ldots \wedge (A_{k-1} \Leftrightarrow A_k) \wedge [(C_1 \vee C_2 \ldots \vee C_k) \Rightarrow C_n]$$

In the equation (7), propositions $P_1 \ldots P_k$ are in the lower order than proposition P_n. Also proposition P_n is total generalization of propositions $P_1 \ldots P_k$, if the correctness of equation (7) is proved.

Redundancy anomaly: Proposition P_1 is redundant to proposition P_2 if they have same actions and if the proposition P_2 can match all the connections that match the proposition P_1 [2].

Formally, proposition P_1 is redundant to proposition P_2 if the following condition holds:

$$\begin{aligned} &P_1 : C_1 \Rightarrow A_1 \\ &P_2 : C_2 \Rightarrow A_2 \end{aligned} \quad (8)$$

$$\exists P_1, P_2 \in SP \bullet (A_1 \Leftrightarrow A_2) \wedge (C_2 \Rightarrow C_1)$$

Equation (8) shows that we must verify the correctness of the equation for each two propositions in the security policy. Although redundancy is sometimes preferred, we consider it an error in the firewall security policy because a redundant proposition increases unnecessary overhead to the matching process.

Total redundancy anomaly: Proposition P_1 is a total redundant of a set of propositions if the set of propositions match all the connections that match the proposition P_1 and the proposition P_1 and the set of propositions have the same filtering action [4].

Formally, proposition P_n is a total redundant of a set of propositions $P_1 \ldots P_k$ if the following condition holds:

$$\begin{aligned} &P_1 : C_1 \Rightarrow A_1 \\ &P_2 : C_2 \Rightarrow A_2 \\ &\vdots \\ &P_n : C_n \Rightarrow A_n \end{aligned} \quad (9)$$

$$\exists P_1, P_2, \ldots, P_k, P_n \in SP \bullet (A_n \Leftrightarrow A_1) \wedge (A_1 \Leftrightarrow A_2) \wedge \ldots \wedge (A_{k-1} \Leftrightarrow A_k) \wedge [C_n \Rightarrow (C_1 \vee C_2 \ldots \vee C_k)]$$

In the equation (9), proposition P_n is total redundant of propositions $P_1 \ldots P_k$, if the correctness of the equation is proved.

ANOMALY DISCOVERY AND RESOLUTION

In this section, we present two algorithms for detecting simple and total anomalies. These algorithms are based on the theorems specified in section 0. Also, we provide two algorithms for resolving these anomalies.

Detection algorithms: For detection of each anomaly, we must prove the correctness of the appropriate theorem. For proving the simple anomalies, each couple of propositions is selected. A simple anomaly is discovered as long as the appropriate theorem for the couple of propositions is satisfied. For proving a total anomaly, we will select a proposition and a set of propositions as input for appropriate theorem. Since the anomaly theorems are specified with logical formulas, we design a proof assistant to prove these theorems. This proof assistant is designed for satisfying a propositional logic formula.

A successful idea for proving propositional formulas that comes from semantics of the logic is that of binary decision diagrams, or BDDs [27, 28]. We might say that they are a recent invention, as the originator of BDDs as we know them today was Randall E. Bryant in 1986 [26]. In our proof assistant, we use the BDD idea.

Figure 1 shows the state machine for anomaly detection in a security policy based on our definition for each anomaly. This state machine can satisfy detection of total anomalies. The state machine is started for two propositions. For example, we apply this machine for policy insertion time and analyze anomalies for new security proposition with all propositions in the policy. At the start state of the machine, two actions are compared. We must search for redundancy anomaly as long as the actions of two propositions are equivalent; otherwise we check three anomalies shadow, generalization and correlation. The order of detection of the three anomalies is important for optimization of analysis process.

When the two propositions have the same action, we check the equation (8) for analyzing redundant anomaly. In this situation, if two propositions are not redundant, they have not any anomaly and the state machine changes to "No Anomaly" state. If the two propositions have different actions, we must check the equations (3), (6) and (5).

We run the detection process in policy insertion process. In this time, the detection algorithms will be invoked for new propositions and all propositions in

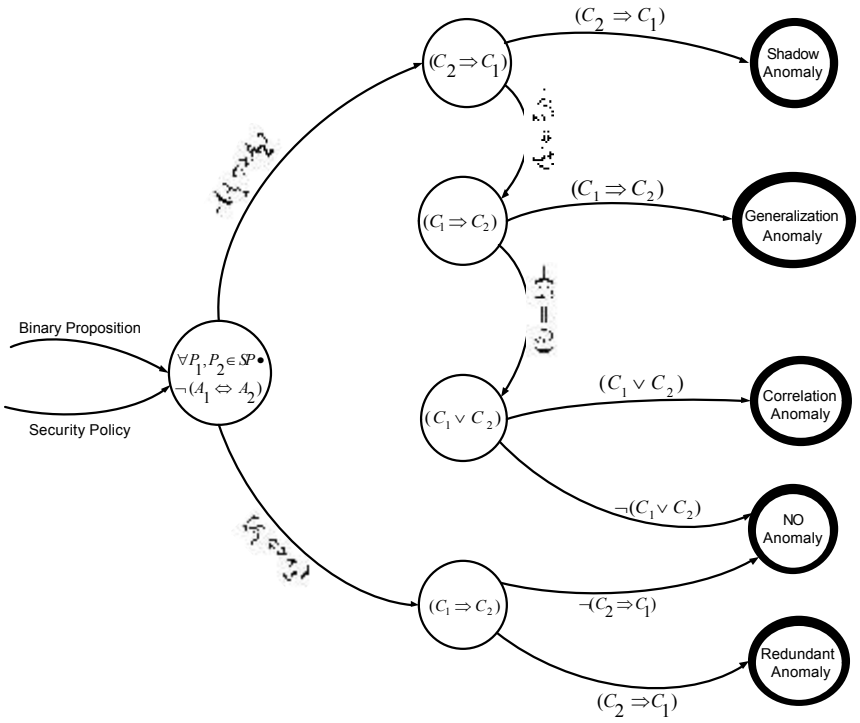

Fig. 1: State machine for discovering anomalies

database and also for new proposition and all subset of propositions in database. Thus we must create logical formulas and use BDD for satisfying them.

Based on state machine in Fig. 1, we present two algorithms for analyzing simple and total anomalies. Algorithm 2 discovers simple anomalies. The inputs of the algorithm are the new proposition P_x and the order of the proposition in database. In the first part of the algorithm, we translate the new proposition P_x to create two parts C_x and A_x (line 2). We search for all propositions in database and check the anomalies formula based on the state machine (Fig. 1). The function BDD_Satisfy is used in two algorithms for satisfying a logical formula. Thus the input of the function is a formula in propositional logic and the output of the function is a Boolean value (TRUE or FALSE).

Algorithm 3 uses the new proposition P_x and its order in security policy to discover any total anomalies. This algorithm has two main steps for checking higher order and lower order propositions. In the first step, the new proposition and a set of lower order propositions are checked for total generalization and total redundancy anomalies (lines 4-15). In the second step of Algorithm 3, the new proposition and higher order propositions are checked for total shadow and total redundancy anomalies (lines 17-28).

```
1.DetectionSimpleAnomaly (Px, order)
2.  Px translated to Cx, Ax
3.  for each Pi in SP
4.    Pi translated to Ci, Ai
5.    if orderi > order then
6.      C1=Ci, C2=Cx, A1=Ai, A2=Ax
7.    else
8.      C1=Cx, C2=Ci, A1=Ax, A2=Ai
9.    if not BDD_Satisfy (A1 ? A2) then
10.     if BDD_Satisfy (C2 ? C1) then
11.         print "Shadow Anomaly (Px, Pi)", Return
12.     if BDD_Satisfy (C1 ? C2) then
13.         print "Generalization Anomaly (Px, Pi)", Return
14.     if BDD_Satisfy (C1 ? C2) then
15.         print "Correlation Anomaly (Px, Pi)", Return
16.   if BDD_Satisfy (C2 ? C1) then
17.     print "Redundancy Anomaly (Px, Pi)", Return
18. print "There is no Simple Anomaly with existent propositions."
19. end DetectionSimpleAnomaly
```

Algorithm 2: Simple anomalies discovery

Resolving algorithms: Regarding the main reason of analysis of anomalies in firewall propositions, the resolving algorithms are presented for optimizing security policy. We propose a new scheme for conflict resolution, which is based on the idea of removing resolve security propositions. Therefore, we resolve the anomalies that lead to decrease the number of propositions in security policy.

```
1.  DetectionTotalAnomaly (??, order)
2.      translate ?? to ??, ??
3.      totalR=False; R=Ø; totalG=False; G=Ø;
4.      for each Rule ?? with lower priority than order in RDB
5.          translated ?? to ??, ??
6.          if not BDD_Satisfy (?? ? ??) then
7.              totalG = totalG ? ??
8.              ?? = ??? ???
9.              if BDD_Satisfy (??????? ? ??)
10.                 print "Total Generalization Anomaly (??, ??)", Return
11.         if BDD_Satisfy (Ax ? Ai) then
12.             totalR = totalR ? ??
13.             ?? = ??? ???
14.             if BDD_Satisfy (??????? ? ??)
15.                 print "Total Redundancy Anomaly (??, ??)", Return
16.     totalR=False; ?? = Ø; totalS=False; ?? = Ø;
17.     for each Rule ?? with higher priority than order in RDB
18.         translate ?? to ??, ??
19.         if not BDD_Satisfy (?? ? ??) then
20.             totalS = totalS ? ??
21.             ?? = ??? ???
22.             if BDD_Satisfy (??????? ? ??)
23.                 print "Total Shadow Anomaly (??, ??)", Return
24.         if BDD_Satisfy (?? ? ??) then
25.             totalR = totalR ? ??
26.             ?? = ??? ???
27.             if BDD_Satisfy (??????? ? ??)
28.                 print "Total Redundancy Anomaly (??, ??)", Return
29.     print "There is no Total Anomaly with existent rules."
30.     end DetectionTotalAnomaly
```

Algorithm 3: Total anomalies discovery

Due to optimization purpose for resolving anomalies, we did not consider correlation and generalization conflicts. We only issue a warning for detecting these anomalies. In contrast, resolving each shadow or redundant anomalies lead to remove a proposition without changing in security policy.

We present two algorithms for resolving simple and total anomalies. Algorithm 4 resolves simple anomalies. The input of the algorithm is the security policy SP. In this algorithm, each two propositions in the security policy are checked for simple shadow and simple redundant anomalies. If one of the anomalies is discovered, the lower proposition in the policy is removed, because the proposition cannot be matched. For discovering the anomalies, we use the formal method in the last section. Therefore, a part of state machine in Fig. 1 is used in Algorithm 3 (lines 10-15).

Algorithm 5 uses the security policy SP to resolve total shadow and total redundant anomalies. This algorithm has two main steps for checking these two total anomalies. In the first step, the selected proposition and a set of lower order propositions are checked for total redundancy anomalies (lines 3-8). In

```
1.ResolveSimpleAnomaly (SP)
2.  for each Pj in SP
3.      Pj translated to Cj, Aj
4.      for each Pi in SP except Pj
5.          Pi translated to C1, Ai
6.          if orderi > orderj then
7.              C1=Ci, C2=Cj, A1=Ai, A2=Aj
8.          else
9.              C1=Cj, C2=Ci, A1=Aj, A2=Ai
10.         if not BDD_Satisfy (A1 ? A2) then
11.             if BDD_Satisfy (C2 ? C1) then
12.                 Remove (R2), Return
13.         if BDD_Satisfy (A1 ? A2) then
14.             if BDD_Satisfy (C2 ? C1) then
15.                 Remove (P2), Return
16. end ResolveSimpleAnomaly
```

Algorithm 4: Resolving simple anomalies

the second step of Algorithm 5, the selected proposition and higher order propositions are checked for total shadow and total redundancy anomalies (lines 9-18). If one of the total anomalies is discovered, the selected proposition in security policy is removed, because the proposition cannot be matched.

Experimental results: We implemented our algorithms using C#. Our experiments were carried out on a desktop PC running Windows Vista with 512M memory and Intel Pentium PIV 2.8 GHz processor.

In order to obtain performance of our firewall anomaly discovery algorithms, ten sets of firewall security propositions are generated. The number of propositions in each set was various from 100 to 100000 propositions. We used the presented discovery algorithms to analyze each security policy. In each test case, we measured the processing time needed to produce the anomaly analysis report.

Table 1 shows the summary of our experimental results. Figure 2 shows the results when running the single anomaly discovery algorithm. Furthermore the results of performance evaluation of total anomaly discovery algorithm are shown in Fig. 3. The increase in the processing time as the rule database size increases is due to the fact that the complexity of our algorithm is dependent on the number of propositions in the security policy of the firewall. Our results indicate that in the worst case, the single anomaly detection process takes 15-2562 ms of processing time to analyze a security policy of 100-100000 propositions and single anomaly detection process takes 16-2453 ms of processing time to analyze a security policy of 100-100000 propositions. Compare to the other works,

Table 1: Processing time (milisecond) for simple and total anomaly discovery

	Simple anomaly				Total anomaly		
Propos. No.	Shadow	Correlation	Generalization	Redundancy	Shadow	Generalization	Redundancy
101	15	15	15	15	16	16	16
201	15	15	15	15	16	16	16
501	15	16	16	16	16	16	16
1001	16	16	16	16	16	15	15
2001	31	32	31	31	31	32	31
5001	78	96	94	78	78	78	78
10001	187	172	184	172	157	156	135
20001	453	454	438	422	359	312	375
50001	1078	1125	1109	1312	1047	1063	1000
100001	2562	2250	2156	2313	2078	2000	2453

we got a lowest time to analyze the anomalies in firewall policies and the more the number of the propositions, the more evident it take effects. Moreover, the algorithms proposed in this paper are not limited to the number of anomalies in the security policy of firewalls. The results in Table 1 show that the processing time of the discovery algorithm for simple anomalies is very close to the performance of discovery algorithm for total anomalies in a firewall containing an equivalent number of security propositions.

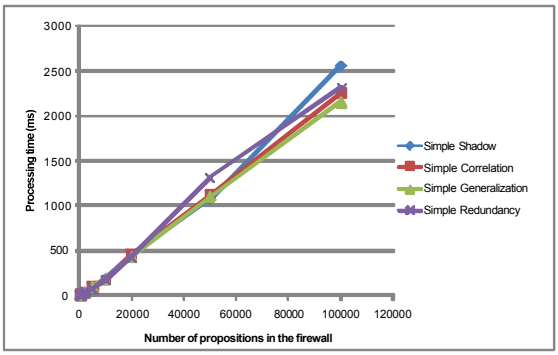

Fig. 2: Processing time for simple anomaly discovery

CONCLUSION AND FUTURE WORKS

In this paper, we have presented a novel technique for specification and verification of anomalies in security policy in firewalls. First, we have provided a formal and high level language for specifying security policy in firewalls. In this language, the security policy is specified based on Deontic logic and is contained of security propositions. Second, we have proposed a framework for translating the specified security policy to propositional logic formulas. This formal framework is used for specifying simple and total anomalies in the security policy of firewalls. Third, we have developed a tool based on BDD for discovering anomalies in a security policy. This analysis process is based on verification of anomaly theorems have been specified in our model. A state machine and two algorithms for detection process have been presented. Our detection technique is useful and practical for firewall administrators. Furthermore, based on our formal model for anomaly specification, we have provided two algorithms for removing simple and total anomalies that lead to decrease the number of rules without changing the security policy.

Our experimental results indicate that in the worst case, the anomaly detection process takes 15-2500 ms of processing time to analyze a security policy of 100-100000 propositions. Moreover, the algorithms proposed in this paper are not limited to the number of anomalies in the security policy of firewalls. The results of our performance evaluation show that the processing time of the discovery algorithm for simple anomalies is very close to the performance of discovery algorithm for total anomalies in a firewall containing an equivalent number of security propositions.

Our future research plan includes integrating the anomaly detection and resolving techniques to our research on developing an optimized structural framework for rule matching in firewalls. In addition we are interested in extending our techniques to support the security policy in distributed firewalls.

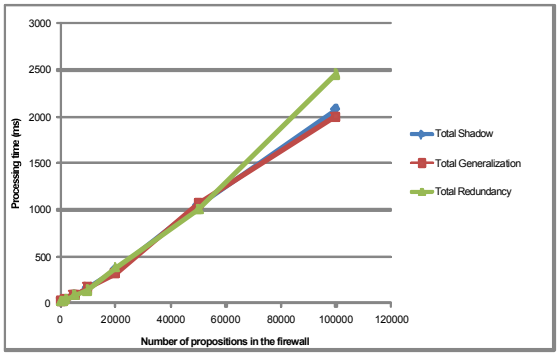

Fig. 3: Processing time for total anomaly discovery

```
1.ResolveTotalAnomaly (SP)
2.  for each Pj in SP
3.      Pj translated to Cj, Aj
4.      for each Pi in SP with priority lower than Pj
5.        Pi translated to Ci, Ai
6.      if BDD_Satisfy (Aj ? Ai) and BDD_Satisfy (? ?=1 (?? ? ??+1)) then
7.        if BDD_Satisfy (? ?=1 (?? ? ??) then
8.          Remove(Pj (, Return
9.      for each Pi in SP with priority higher than Pj
10.       Pi translated to Ci, Ai
11.     if not BDD_Satisfy (Aj ? Ai) and BDD_Satisfy (? ?-1 (?? ? ??-1)) then
12.       if BDD_Satisfy (? ?=1 (?? ? ??) then
13.         for each ?? ?? ? ?=1 ??
14.             if BDD_Satisfy(? 5 ?=1 (?? ? ??) then
15.               Remove(??, ??, ??), Return
16.     if BDD_Satisfy (Ax ? Ai) and BDD_Satisfy (? ?-1 (?? ? ??-1)) then
17.       if BDD_Satisfy (? ?=1 (?? ? ??) then
18.         Remove(Pj), Return
19. end ResolveTotalAnomaly
```

Algorithm 5: Resolving total anomalies

REFERENCES

1. Cheswick, R.W., M.S. Bellovin and D.A. Rubin, 2003. Firewalls and Internet Security, 2nd Edn. Addison Wesley Professional.
2. Al-Shaer, E. and H. Hamed, 2006. Taxonomy of conflicts policies. IEEE Communications Magazine, Vol: 44 (3).
3. Ortalo, R., 1996. Using deontic logic for security policy specification. Tech. Report 96380, LAAS-CNRS Toulouse, France.
4. Rezvani, M. and R. Arian, 2009. Analyzing anomalies in firewall security rules. Proceedings of the 2009 International Conference on Information Security and Privacy (ISP-09), Orlando, FL, USA.
5. Rezvani, M., 2001. High Level Security Policy Specification in Firewalls, Master's Thesis. Dept. of Computer Engineering, Sharif University of Technology, In Persian.
6. Peri, R.V., 1996. Specification and Verification of Security Policies. Ph.D Thesis, The University of Virginia, School of Engineering and Applied Science.
7. Alur, R. and T.A. Henzinger, 1999. Computer-Aided Verification: An Introduction to Model Building and Model Checking for Concurrent Systems, Draft.
8. Cholvy, L. and F. Cuppens, 1997. Analyzing consistency of security policies. Proceedings of the IEEE Symposium on Security and Privacy, pp: 103-112.
9. Hoagland, J., 2000. Specifying and Implementing Security Policy using LaSCO, the Language for Security Constraints on Objects. Ph.D Thesis, The University of California Davis, Department of Computer Science.
10. Zhang, C., M. Winslett and A.C. Gu nter, 2007. On the safety and efficiency of firewall policy deployment. Proceedings of IEEE Symposium on Security and Privacy.
11. Eronen, P. and J. Zitting, 2001. An expert system for analyzing firewall rules. Proceedings of the 6th Nordic Workshop on Secure IT-Systems (NordSec 2001).
12. Pornavalai, C. and T. Chomsiri, 2004. Firewall policy analyzing by relational algebra. Proceedings of the 2004 International Technical Conference on Circuits/Systems. Computers and Communications (ITC-CSCC 2004).
13. Cuppens, F., N.C. Boulahia and J.G. Alfaro, 2005. Detection and removal of firewall misconfiguration. Proceedings of the 2005 IASTED International Conference on Communication, Network and Information Security, Phoenix, USA, pp: 154-162.
14. Chow, E.C., G.K. Godavari and J. Xie, 2001. Content switch rules and their conflict detection. Proceeding of the PDCAT 2001, Taipei, Taiwan.
15. Capretta, V., B. Stepien, A. Felty and S. Matwin, 2007. Formal correctness of conflict detection for firewalls. Proceedings of the 2007 ACM Workshop on Formal Methods in Security Engineering, pp: 22-30.
16. Mankai, M. and L. Logrippo, 2005. Access control policies: modeling and validation. Proceedings of the 5th NOTERE Conference (Nouvelles Technologies de la Re´partition), Gatineau, Canada, pp: 85-91.

17. Baboescu, F. and G. Varghese, 2002. Fast and scalable conflict detection for packet classifiers. Proceedings of the Tenth IEEE International Conference on Network Protocols, pp: 270-279.
18. Yuan, L., J. Mai, Z. Su, H. Chen, C. Chuah and P. Mohapatra, 2006. FIREMAN: A toolkit for FIREwall Modeling and Analysis. Proceedings of IEEE Symposium on Security and Privacy, pp: 199-213.
19. Arosha, B., K. Kakas, A.L. Emil and C.R. Alessandra, 2006. Using argumentation logic for firewall policy specification and analysis. Proceedings of the 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM), Springer Verlag.
20. Qian, J., S. Hinrichs and K. Nahrstedt, 2001. ACLA: A Framework for Access Control List (ACL) Analysis and Optimization. Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security Issues of the New Century, pp: 4.
21. Liu, A. and M. Gouda, 2005. Complete redundancy detection in firewalls. Proceedings of the 19th Annual IFIP Conference on Data and Applications Security.
22. Adiseshu, H., S. Suri and G. Parulkar, 2000. Detecting and Resolving Packet Filter Conflict. Proceedings of IEEE Infocom.
23. Bartal, Y., A.J. Mayer, K. Nissim and A. Wool, 1999. Firmato: A novel firewall management toolkit. Proceedings of the 20th IEEE Symposium on Security and Privacy.
24. Mayer, A., A. Wool and E. Ziskind. 2000. Fang: A firewall analysis engine. Proceedings of IEEE Symposium on Security and Privacy.
25. Stevens, W.R., 1994. TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, Reading, MA, USA.
26. Goubault-Larrecq, J. and I. Mackie, 1997. Proof Theory and Automated Deduction, Volume 6 of Applied Logic Series, Kluwer Academic Publishers, Dordrecht.
27. Hazelhurst, S., 1999. Algorithms for analyzing firewall and router access lists. University of Witwatersrand, Johannesburg, South Africa, Tech. Rep. TR-Wits-CS-1999-5.
28. Bryant, R., 1991. On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication. IEEE Transactions on Computers, pp: 205-213.