A NOVEL ALGORITHM FOR DETECTING CONFLICTS IN FIREWALL RULES

Amjad Gawanmeh[1] *and Sofiène Tahar*[2]

[1]Department of Electrical and Computer Engineering,
Khalifa University of Science, Technology and Research, Sharjah, UAE
amjad.gawanmeh@kustar.ac.ae
[2]Department of Electrical and Computer Engineering,
Concordia University, Montreal, Québec, Canada
tahar@ece.concordia.ca

ABSTRACT

Firewalls are widely adopted for protecting private networks by filtering out undesired network traffic in and out of secured networks. Therefore, they play an important role in the security of communication systems. The verification of firewalls is a great challenge because of the dynamic characteristics of their operation, their configuration is highly error prone, and finally, they are considered the first defense to secure networks against attacks and unauthorized access. In this paper, we present a formal model for firewalls rulebase and a novel algorithm for detecting and identifying conflicts in firewalls rulebase. Our algorithm is based on calculating the conflict set of firewall configurations using the domain restriction. We show that the algorithm terminates, then we apply it on a firewall rulebase example.

***Index Terms*—** Fireall security, Formal model, Formal verification, Rulebase conflict

1. INTRODUCTION

Firewalls [1] are part of network security that were designed to enable secure connections between private and outside networks. The growing complexity of networks made them indispensable to control information flow within a network, and they are widely adopted technologies for protecting private networks. Therefore, firewalls have been the frontier defense for secure networks against attacks and unauthorized traffic by filtering out unwanted network traffic coming into or going from the secured network. Testing and verification of firewalls is a great challenge because of the dynamic characteristics of their operation, their configuration is highly error prone, and finally, they are considered the first defense to secure networks against attacks and unauthorized access. In addition, firewalls can be used extensively before it turns out that they are vulnerable to attacks, even though they receive intensive analysis, and are thought to be correct. Most firewall operations depend on an existing sequence of rules, which is intentionally made dynamic in order to eliminate certain denial of service (DoS) attacks. Therefore, it is essential to detect conflicting rules in firewalls configurations, and at the same time be able to decide if they conform to the security requirement of the firewall.

Formal methods [2] are based on using mathematical reasoning to verify that design specifications comprehend certain design requirements. Formal methods have been successfully used for the precise analysis of a variety of hardware and software systems [3]. In this paper we extend our previous work in [4] by proposing a formal model for firewalls rulebase and a novel algorithm for the verification of firewalls rulebase that can efficiently detect conflicts in firewall rules. In [4], we presented the domain restriction method implemented in Event-B [5], while in this work, the algorithm is based on formally calculating the *conflict set* for a given firewall rulebase. In addition, the algorithm can verify rulebase consistency, and identify conflicting rules, if they exist. The formal model is also used in order to prove three properties for the algorithm: termination, correctness, and soundless.

The rest of the paper is organized as follows: Section 2 provides a brief literature review on verification of firewall related properties. In Section 3, we present the formal model and domain restriction method. Section 4 presents our algorithm for detecting conflicts in firewall rules illustrated on a firewall rulebase example. Finally, Section 5 concludes the paper with future work hints.

2. RELATED WORK

In this section we discuss related work on using formal methods for verification of firewalls and their configurations. Abbes *et al.* [6] proposed a method to detect overlaps between packet filters within one firewall, they classify rules based on the conditions of each filtering rule to separate non-overlapping rules. Ben Youssef *et al.* [7] proposed a method for checking whether a firewall reacts correctly with respect to a security policy given in high level declarative language. The method is implemented in satisfiability solver modulo theories (SAT solver). These works focus on validating firewall rulebase with regard to their security policy, while our focus will be on the consistency of the rulebase.

In another approach, Brucker *et al.* [8] presented a case study to model firewalls and their policies in higher-order logic (HOL) throughout a set of derived theories for simplifying policies. Matoušek *et al.* [9] introduced a formal method approach for verification of security constraints on networks with dynamic routing protocols in use. We believe a formal model that captures the details of firewall rulebases is necessary prior to the definition of an effecting algorithm in this particular case. Acharya and Gouda [10] proposed a verification algorithm that takes a firewall *F* and a property *R*, and determines whether or not the firewall *F* satisfies the property *R*. Kotenko and Polubelova [11] used Promela for detecting anomalies in the specification of the security policy of computer networks with model checking, the method is implemented in the SPIN model checker.

Liu [12] verified in his work whether a firewall policy satisfies a given property. The method is based on showing that a property about rules does not conflict with any rule defined by a decision path of the firewall decision diagram. Jeffery and Samak [13] used SAT solvers for the model analysis of reachability and cyclicity properties of interest in firewall policies. The model for network configurations is based on a single firewall model and is shown to be efficient compared to BDD based approaches. The use of model checking has the problem of state space explosion, specially for a large number of firewall rules.

Most of above the approaches only check for conflicts between rules which is obtained by inspecting certain fields in the policy, rather than checking them in the rulebase, in addition, it considers only firewall policies at high level of abstraction. They also ignored the dynamic update of these rules, and did not consider the sequence at which these rules are inspected. Therefore the verification of firewalls rulebase, considering their dynamic sequence, is essential and have not been yet explored thoroughly.

3. A FORMAL MODEL FOR FIREWALL RULES

Firewalls inspect a set of rules in order to filter traffic based on protocol type, port used, or source and destination IP addresses. Firewall actions are either to accept, or deny incoming and outgoing traffic. We assume a finite domain containing the possible network addresse pairs in a firewall rulebase $\langle s, d \rangle$, i.e., source and destination, where either s or d can be empty. Let $\mathcal{N}$ be the set of possible network address pairs for packets incoming to and outgoing from a network such that $\langle s, d \rangle \in \mathcal{N}$. We define two sets based on $\mathcal{N}$, the first, $\mathcal{N}_s$, is for source addresses, and the second, $\mathcal{N}_d$, is for destination addresses. $\mathcal{N}$ is an abstract set that represents that actual network addresses, which can be further refined to represent protocol type or port numbers in a network address. Let $\mathcal{A}$ be the abstract set of all possible actions a firewall can perform, this set can be defined as follows: $\mathcal{A} = \{accept,\ deny\}$. We define every firewall rule to be a mapping relation from an address pair in $\mathcal{N}$ into an action in $\mathcal{A}$, formally, $r = n \mapsto a$, where $n \in \mathcal{N}$, $a \in \mathcal{A}$ and $\mapsto$ is a mapping relation from addresses to actions that maps one element in $\mathcal{N}$ to an element in $\mathcal{A}$. n may contain either source or destination or both. We use $source(r)$ and $dest(r)$ to denote the address that appears in rule r. We use $\mathfrak{R}$ to denote the set of all possible firewall rules such that $\mathfrak{R} = \mathcal{N} \times \mathcal{A}$.

A firewall rulebase, $\mathcal{R}$, is a finite set of rules: $\{r_1, r_2, \ldots r_n\}$, such that $\mathcal{R} \subset \mathfrak{R}$. A firewall is configured so that $\mathcal{R}$ is inspected in an arbitrary order. A firewall configuration, $\mathcal{F}$, is an ordered sequence of rules in the form: $\mathcal{F} = r_1, r_2, \ldots, r_n$ such that $r_1 \in \mathcal{R} \wedge r_2 \in \mathcal{R} \wedge \ldots r_n \in \mathcal{R}$. We use $\mathcal{R}(\mathcal{F})$ to denote $\mathcal{R}$ for a given firewall configuration $\mathcal{F}$.

For a given firewall rulebase with a set of rules, a packet with a source address, a destination address, or both, is checked by inspecting the rules in sequence. The basic principle of firewalls operation states that the order in which rules are inspected should not affect the result. $\mathcal{F}'$ is an arbitrary firewall configuration for $\mathcal{F}$ if every rule in $\mathcal{F}$ is also in $\mathcal{F}'$ and every rule in $\mathcal{F}'$ is also in $\mathcal{F}$, $\forall r_i \cdot r_i \in \mathcal{R}(\mathcal{F}) \Rightarrow r_i \in \mathcal{R}(\mathcal{F}') \wedge r_i \in \mathcal{R}(\mathcal{F}') \Rightarrow r_i \in \mathcal{R}(\mathcal{F})$, and there are at least two rules that are inspected in different order: $\mathcal{F} \neq \mathcal{F}'$

Conflict set, $\mathcal{CS}$, is a set of all rules such that, (1) $\forall r \cdot r \in \mathcal{CS} \Rightarrow r \in \mathcal{R}(\mathcal{F})$, and consequently $r \in \mathcal{R}(\mathcal{F}')$, (2) $\exists n \cdot n \in \mathcal{N} \wedge n = \ < source(r), dest(r) >$, and (3) $\exists r' \cdot r' \in \mathcal{R}(\mathcal{F}) \wedge r' \neq r \wedge n = \ < source(r'), dest(r') >$.

A given firewall configuration is considered consistent if there are no conflicts in its rules. The inconsistency occurs when two different rules match the packet being inspected, and each rule gives a different action. The consistency property is denoted as ϕ. If a given firewall configuration is consistent then we can write: $\phi \models \mathcal{F}$.

The domain of firewall rules, $\mathcal{D}$, is defined as:
$\mathcal{D}(\mathcal{F}) = \{n | n \in \mathcal{N} \wedge \exists a, r \cdot (a \in \mathcal{A} \wedge r \in \mathcal{R}(\mathcal{F}) \wedge r = n \mapsto a)\}$

Furthermore, two domains can be defined for source and destination addresses, $\mathcal{N}_s$ and $\mathcal{N}_d$, respectively, as:
$\mathcal{D}_s(\mathcal{F}) =$
$\{s | n = \langle s, d \rangle \wedge n \in \mathcal{N} \wedge s \in \mathcal{N}_s \wedge d \in \mathcal{N}_d \wedge \exists a, r \cdot (a \in \mathcal{A} \wedge r \in \mathcal{R}(\mathcal{F}) \wedge r = n \mapsto a)\}$
$\mathcal{D}_d(\mathcal{F}) =$
$\{d | n = \langle s, d \rangle \wedge n \in \mathcal{N} \wedge s \in \mathcal{N}_s \wedge d \in \mathcal{N}_d \wedge \exists a, r \cdot (a \in \mathcal{A} \wedge r \in \mathcal{R}(\mathcal{F}) \wedge r = n \mapsto a)\}$

Similarly, the configuration co-domain, $\mathcal{C}$, defined as:
$\mathcal{C}(\mathcal{F}) = \{a | a \in \mathcal{A} \wedge \exists n, r \cdot (n \in \mathcal{N} \wedge r \in \mathcal{R}(\mathcal{F}) \wedge r = n \mapsto a)\}$

Domain restriction is applied on firewall rulebase in order to obtain a subset of $\mathcal{R}(\mathcal{F})$. The operators $\lhd$ and $\rhd$ are used to represent domain restriction and co-domain restriction over a set of firewall rules, respectively. First, we formally define domain restriction based on a set of network address pairs, then we refine ir further for source and destination addresses.

Domain restriction is defined using the operator $\lhd$ over a given set of network addresses, N, where $N \in \mathcal{P}(\mathcal{D}(\mathcal{F}))$, and a set of firewall rules $\mathcal{R}(\mathcal{F})$ as follows:

$$N \lhd \mathcal{R}(\mathcal{F}) = \{n \mapsto a | n \in N \wedge a \in \mathcal{A} \wedge \exists r \cdot (r \in \mathcal{R}(\mathcal{F}) \wedge r = n \mapsto a)\}$$

Domain restriction of firewall configurations network source and destination addresses, N_s and N_d, where $N_s \in \mathcal{P}(\mathcal{D}_s(\mathcal{F}))$ and $N_d \in \mathcal{P}(\mathcal{D}_d(\mathcal{F}))$, and $N \in \mathcal{P}(\mathcal{D}(\mathcal{F}))$, is defined respectively as:

$$N_s \lhd \mathcal{R}(\mathcal{F}) = \{n \mapsto a | n \in N \wedge a \in \mathcal{A} \wedge \exists r \cdot (r \in \mathcal{R}(\mathcal{F}) \wedge \exists d \cdot (d \in \mathcal{D}_d \wedge n = \langle s, d \rangle \wedge r = n \mapsto a))\}$$

$$\mathcal{N}_d \lhd \mathcal{R}(\mathcal{F}) = \{n \mapsto a | n \in N \wedge a \in \mathcal{A} \wedge \exists r \cdot (r \in \mathcal{R}(\mathcal{F}) \wedge \exists s \cdot (s \in \mathcal{D}_s \wedge n = \langle s, d \rangle \wedge r = n \mapsto a))\}$$

Co-domain restriction is defined for a chosen set of actions $A \in \mathcal{P}(\mathcal{A})$, the operator $\rhd$ is used to represent this operation, which is formally defined as follows:

$$A \rhd \mathcal{R}(\mathcal{F}) = \{n \mapsto a | n \in \mathcal{N} \wedge a \in \mathcal{A} \wedge \exists r \cdot (r \in \mathcal{R}(\mathcal{F}) \wedge r = n \mapsto a))\}$$

4. DETECTING CONFLICTS IN FIREWALL RULES

Domain restriction operation is closed under $\mathcal{N}$, $\mathcal{N}_s$, and $\mathcal{N}_d$. In addition $\mathcal{N} \lhd \mathcal{R}(\mathcal{F})$, $\mathcal{N}_s \lhd \mathcal{R}(\mathcal{F})$, and $\mathcal{N}_d \lhd \mathcal{R}(\mathcal{F})$ obtain the same set, namely, $\mathcal{R}(\mathcal{F})$. Similarly, co-domain restriction is closed under $\mathcal{A}$, this property helps in the definition of the properties of the algorithm we propose here. We divide our algorithm into two steps, the first one, as shown below, is used to model consistency of firewall configurations based on conflict set:

Algorithm 1 Detecting Conflicts in Firewall Configurations

```
Input: Firewall Configuration $\mathcal{F}$
Output: Consistency of $\mathcal{F}$ : $\phi \models \mathcal{F}$
Calculate Conflict Set for $\mathcal{F}$: $\mathcal{CS}$ using Algorithm 2
if $\mathcal{CS} == \emptyset$ then
    $\phi \vdash \mathcal{F}$
else
    $\phi \nvdash \mathcal{F}$
end if
```

The second one is used to calculate the conflict set, $\mathcal{CS}$, of a given firewall configuration $\mathcal{F}$, or alternatively, $\mathcal{R}(\mathcal{F})$. The algorithm works by inspecting all rules in $\mathcal{R}(\mathcal{F})$ and obtaining source and destinations addresses for every rule, then two simple sets of actions $A_a = \{accept\}$ and $A_d = \{deny\}$ are defined. Then we calculate the set of rules that occur in the domain of this network address for source, R_s, and another for destination, R_d, by applying domain restriction method: $R_s = \mathcal{N}_s \lhd \mathcal{R}$ and $R_d = \mathcal{N}_d \lhd \mathcal{R}$. Next, co-domain restriction operator applied on R_s and R_d in order to calculate two sets of rules using co-domain restriction for A_a, called R_{sa} and R_{da}, and two for A_d, called R_{sd} and R_{dd}. Where, $R_{sa} = A_a \rhd R_s$, $R_{sd} = A_d \rhd R_s$, $R_{da} = A_a \rhd R_d$, and $R_{dd} = A_d \rhd R_d$. Finally, we check for existing conflicts for source and destination and update the conflict set. This operation is repeated for all rules. Algorithm 2 below:

Algorithm 2 Calculating Conflict Set

```
Input: Firewall Configuration $\mathcal{F}$
Output: Conflict Set: $\mathcal{CS}$
Initialize:
  $\mathcal{CS} = \emptyset$
  $A_a = \{accept\}$
  $A_d = \{deny\}$
REPEAT
• Chose rule $r_i \in \mathcal{R}(\mathcal{F})$ and obtain network addresses for $r_i$
      $n_s = source(r_i)$
      $n_d = dest(r_i)$
• Apply domain restriction:
      $R_s = n_s \lhd \mathcal{R}(\mathcal{F})$ ;  Source address
      $R_d = n_d \lhd \mathcal{R}(\mathcal{F})$ ;  Destination address
• Apply co-domain restriction:
      $R_{sa} = A_a \rhd R_s$  ;  *Accept* action
      $R_{da} = A_a \rhd R_d$
      $R_{sd} = A_d \rhd R_s$  ;  *Deny* action
      $R_{dd} = A_d \rhd R_d$
• Check for conflicts for $r_i$ :
if $R_{sa} \neq \emptyset \wedge R_{sd} \neq \emptyset$ then
    $CS = CS \bigcup \{r_i\}$
else
    No conflict for source address
end if
if $R_{da} \neq \emptyset \wedge R_{dd} \neq \emptyset$ then
    $CS = CS \bigcup \{r_i\}$
else
    No conflict for destination address
end if
  $\mathcal{R}(\mathcal{F}) = \mathcal{R}(\mathcal{F}) - \{r_i\}$
UNTIL $\mathcal{R}(\mathcal{F}) == \emptyset$
END
```

The algorithm has n^2 complexity, since domain restriction operators have a complexity of n, where n represents the number of rules. It is essential to show that the algorithm, once applied on a finite number of rules will terminate. Termination is constructed by observing the behavior of each step in the algorithm, and showing that one dependence is solved in every iteration, the space is finite, and it is decreasing in every iteration, starting with a set of rules, in every iteration of the algorithm this set will be reduced in $\mathcal{R}(\mathcal{F}) = \mathcal{R}(\mathcal{F}) - \{r_i\}$, and eventually, $\mathcal{R}'(\mathcal{F})$ becomes $\emptyset$, which is the precondition for algorithm termination.

Firewall Rulebase Example. An example of a firewall rulebase is given below, where the firewall, once a packet is received, checks its corresponding chain of rules and decides if the packet must be dropped or allowed to pass.

```
r1 = If source IP address = 10.*.*.*, DENY
r2 = If source IP address = 192.168.*.*,ACCEPT
r3 = If source IP address = 0.0.0.0, DENY
r4 = If source IP address = 10.1.*.* to 10.3.*.*,
     DENY
r5 = If source IP address = 60.40.*.*, ACCEPT
r6 = If source IP address = 1.2.3.4, DENY
r7 = If destination IP address = 60.47.3.9 AND
     destination port=80 OR 443, ACCEPT
r8 = If destination IP address = 60.47.3.*
     AND destination port= 21, ACCEPT
DENY ALL default rule
```

For the above firewall, $\mathcal{F} = r_1, r_2, \ldots, r_8$, and the set of rules $\mathcal{R}(\mathcal{F}) = \{r_1, r_2, \ldots, r_8\}$. Applying the algorithm for the first iteration, $r_i = r_1$, $n_s = 10.*.*.*$, and n_d is empty. Applying domain restriction, $R_s = \{r_1, r_4\}$, $R_d = \emptyset$, then, applying co-domain restriction, $R_{sa} = \emptyset$, $R_{sd} = \{r_1, r_4\}$, $R_{da} = \emptyset$, and $R_{dd} = \emptyset$, and therefore $\mathcal{CS} = \emptyset$, then $\mathcal{R}(\mathcal{F}) = \{r_2, \ldots, r_8\}$. Repeat the above procedure for all rules, $\mathcal{CS}$ will remain empty, which indicates that there is no conflict in the above rulebase. Considering an additional rule, r_9:

```
r9 = If source IP address = 10.40.*.*, ACCEPT
```

we apply the algorithm again, then $\mathcal{R}(\mathcal{F}) = \{r_1, r_2, \ldots, r_9\}$, then starting with $r_i = r_1$, we obtain $n_s = 10.*.*.*$, and $n_d = null$, then $R_s = \{r_1, r_4, r_9\}$, $R_d = \emptyset$, next step, $R_{sa} = \{r_9\}$, $R_{sd} = \{r_1, r_4\}$, $R_{da} = \emptyset$, and $R_{dd} = \emptyset$. Since $R_{sa} \neq \emptyset \wedge R_{sd} \neq \emptyset$ then, $\mathcal{CS} = \{r_1\}$, which indicates that this rule has conflict with another one in the rulebase.

This algorithm can be applied dynamically while updating the rulebase. In this case, the complexity of the algorithm is reduced to linear for a newly added rule. The algorithm can verify consistency of firewall rulebase, and in addition, identify all the rules with conflicts if they occur.

5. CONCLUSION

Firewall configuration and the maintenance of their rulebase is highly error prone, therefore, the verification of their correctness is essential. In this paper, we present an algorithm for detecting conflicts in firewall configurations rules that can be used to verify the consistency of these configurations taking in consideration their dynamic operation. The algorithm is based on a formal model that utilizes the domain restriction method we presented in our previous work [4]. Compared to [4], this work presents an implementation independent algorithm based on a formal model for firewalls rulebase. The algorithm is supported with an intuitive proof of termination.

This method can model firewall configuration rules at the network address level of abstraction, which we believe is the major domain where most conflicts happen to be in firewalls rulebase. However, the method can still be modified to support detecting conflicts in rules at the protocol and ports level of abstraction. This requires modifying the formal model and hence, the above algorithm. Yet the conflict set based method can still be applied with the same level of complexity. The algorithm can be implemented in any existing SAT solver. Alternatively, it can be integrated into an existing theorem prover that supports first-order set theory operations such as Event-B or HOL theorem provers.

6. REFERENCES

[1] D. Chapman and E. Zwicky, *Building Internet Firewalls*, Orielly & Associates Inc., 2000.

[2] J.R. Abrial, "Faultless Systems: Yes We Can!," *IEEE Computer Journal*, vol. 42, no. 9, pp. 30–36, 2009.

[3] P. Boca and J.P. Bowen J. Siddiqi, *Formal Methods: State of the Art and New Directions*, Springer-Verlag London Limited, 2010.

[4] A. Gawanmeh and S. Tahar, "Modeling and Verification of Firewall Configurations Using Domain Restriction Method," in *IEEE International Conference on Internet Technology and Secured Transactions*. pp. 642–647, IEEE Computer Society Press, 2011.

[5] J.R. Abrial, *Modelling in Event-B: System and Software Engineering*, Cambbridge University Press, 2009.

[6] T. Abbes, A. Bouhoula, and M. Rusinowitch, "An Inference System for Detecting Firewall Filtering Rules Aanomalies," in *ACM Symposium on Applied computing*. pp. 2122–2128, ACM press, 2008.

[7] N. Ben Youssef, A. Bouhoula, and F. Jacquemard, "Automatic Verification of Conformance of Firewall Configurations to Security Policies," in *IEEE Symposium on Computers and Communications*. pp. 526–531, IEEE Computer Society Press, 2009.

[8] A. Brucker, L. Brügger, and B. Wolff, "Model-Based Firewall Conformance Testing," in *Testing of Software and Communicating Systems*. vol. 5047 of *LNCS*, pp. 103–118, Springer-Verlag, 2008.

[9] P. Matoušek, J. Ráb, O. Ryšavý, and M. Švéda, "A Formal Model for Network-Wide Security Analysis," in *IEEE International Conference on Engineering of Computer Based Systems*. pp. 171–181, IEEE Computer Society Press, 2008.

[10] H. Acharya and M. Gouda, "Projection and Division: Linear-Space Verification of Firewalls," in *IEEE International Conference on Distributed Computing Systems*. pp. 736–743, IEEE Computer Society Press, 2010.

[11] I. Kotenko and O. Polubelova, "Verification of Security Policy Filtering Rules by Model Checking," in *IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*. pp. 706–710, IEEE Computer Society Press, 2011.

[12] A.X. Liu, "Formal Verification of Firewall Policies," in *IEEE International Conference on Communications*. pp. 1494–1498, IEEE Computer Society Press, 2008.

[13] A. Jeffrey and T. Samak, "Model Checking Firewall Policy Configurations," in *IEEE Symposium on Policies for Distributed Systems and Networks*. pp. 60–67, IEEE Computer Society Press, 2009.