6th International Conference on Internet Technology and Secured Transactions, 11-14 December 2011, Abu Dhabi, United Arab Emirates

MODELING AND VERIFICATION OF FIREWALL CONFIGURATIONS USING DOMAIN RESTRICTION METHOD

Amjad Gawanmeh

Khalifa University
Sharjah Campus, PO.Box 573 Sharjah, UAE
amjad.gawanmeh@kustar.ac.ae

Sofiène Tahar

Concordia University
Montreal, Quebec, H3G 1M8 Canada
tahar@ece.concordia.ca

ABSTRACT

Firewalls play an important role in the security of communication systems. They are widely adopted for protecting private networks by filtering out undesired network traffic in and out of the secured network. 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 propose a new approach for modeling and verification of firewall configuration rules using domain restriction method. Our approach is implemented in Event-B formal techniques, where we model firewall configuration rules, and then use invariant checking to verify the consistency of firewall configurations in Event-B theorem proving framework.

1. INTRODUCTION

With the growing complexity of computer networks, security has become a crucial issue. Firewalls are part of network security that were designed to enable secure connections between private networks and outside networks, the growing complexity of networks made them indispensable to control information flow within a network. Therefore, they are widely adopted for protecting private networks by filtering out undesired network traffic in and out of the secured network. Therefore, firewalls are the front defense for secure networks against attacks.

The configuration of a firewall is highly error prone and solutions are needed in order to analyze its correctness. A firewall configuration error can create security holes that will allow undesired traffic to pass into a private network or blocks legitimate traffic and disrupts normal firewall operation, which can lead to undesired consequences. Therefore, the central role of firewalls in the security of the organization information make their management a critical task.

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 thought to be correct. This shows that the informal design of firewall and their configuration is error prone because reasoning about them is difficult. In addition, most firewalls operation depends on the existing sequence of rules, which is intentionally made dynamic in order to eliminate certain 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 verification [1] uses mathematical reasoning to verify that design specifications are correct against certain design requirements. Formal methods have been successfully used for the precise analysis and verification of a verity of systems including protocols, security systems, software and hardware systems.

In this paper we propose a new approach for analyzing firewall configurations based on formal techniques in order to show that they are correctly implemented. Our method for verification of firewall configuration rules is based on calculating domain restricted sets. A formal model for firewall configuration rules is defined, then, domain restriction operations are defined on this model. Finally, these operations are used to formally verify the consistency of the configuration rules. For illustration purpose, we use the Event-B [2] formal method in order to apply our approach on firewall example, where we model firewall rules in Event-b, then define an appropriate invariant to check consistency in firewalls rules.

The rest of the paper is organized as follows. Section 2 discusses related work. In Section 3, we present a formal model for firewall configurations and domain restriction based method for their verification. In Section 4, we present an implementation of the method in Event-B, and illustrate it on a case study. Finally, Section 5 concludes the paper with open issues and future work directions.

978-1-908320-00-1/11/$26.00 ©2011 IEEE

2. RELATED WORK

Many methods have been proposed for the detection of rules conflicts in firewalls configurations. Abbes *et al.* [3] 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.* [4] proposed a method for checking whether a firewall reacts correctly with respect to a security policy given in an high level declarative language, the method is implemented in satisfiability solver modulo theories (SAT solver). These work are limited to the problem conflict avoidance, and do not consider the more general problem of verifying whether a firewall reacts correctly with respect to a given security policy. Brucker *et al.* [5] presented a case study to model firewalls and their policies in higher-order logic (HOL) throughout a set of derived theories for simplifying policies. Cuppens *et al.* [6] proposed an automatic process generating firewall rules from an abstract specification of a security policy.

Liu [7] verifies in his work wither a firewall policy satisfies a given property. The method is based on checking whether a the property rule does not conflict with any rule defined by a decision path of the firewall decision diagram. Yuan *et al.* [8] introduced FIREMAN, a static analysis toolkit for firewall modeling and analysis that treats firewall configurations as specialized programs and applies static analysis techniques to check problems configurations, such as policy violations, inconsistencies, and inefficiencies in firewalls. Jeffery and Samak [9] used SAT solvers for the model analysis of reachability and cyclicity properties of interest in firewall policies. In another work Matoušek *et al.* [10] introduced a formal method approach for verification of security constraints on networks with dynamic routing protocols in use.

In fact, there is good amount of work in the literature on detecting firewall policies conflict, approaches only checks for conflicts between rules which is obtained by inspecting certain fields in the policy, in addition, it considers only firewall policies at high level of abstraction. On the other hand, the policy itself can be wrongly implemented in the firewall configuration in the absence of conflicts, therefore there is a need to check the consistency of firewall configuration with regards to firewall policies. We believe the use of first-order theorem proving will be more efficient in this particular case. Even though fully automatic verification will not be feasible, with some effort the verification can still be conducted within reasonable amount of interaction.

3. DOMAIN-BASED RESTRICTION METHOD

In this section we present our formal model for firewall configurations, where we formally define components and relations in firewalls, then our verification methods based on this model. Firewalls [11] are network elements that controls packets in a secured network based on a set of rules. These rules define the actions performed by the the firewall based on certain and configured filtering conditions. Firewall rules filter traffic based on protocol type, port used, or source and destination IP addresses Firewall actions are either to accept, or deny. The first allows packets to pass through, while the second blocks them. Rules are examined in sequence, the packet is accepted or denied by a specific rules if it matches the required network addressing fields of this rule. Otherwise, the following rule is examined until a matching rules is found. In case no rule is found, a default policy action can be performed.

In order to provide a formal and precise model for the above description, we will use first-order logic that allows reasoning about firewall operations and primitives, while at the same time, can be implemented directly in supporting verification Methods such as Event-b.

We assume a finite domain containing the possible network addresses pairs in a firewall configuration $\langle s, d\rangle$, i.e. source and destination, 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 will be refined in order to represent actual network addresses, it can be refined further to represent protocol type or port numbers in IP network addressing scheme. 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 $\mathcal{A}$. We define $\mathcal{R}$ be the set of firewall rules such that $\mathcal{R} = \mathcal{N} \times \mathcal{A}$, therefore, $r \in \mathcal{R}$. A sequence of rules is a subset of the power set of $\mathcal{R}$, a firewall configuration $\mathcal{F}$ is a finite sequence of rules of the form $r_1, r_2, \ldots r_i$, therefor, we can write: $\mathcal{F} \in \mathcal{P}(\mathcal{N} \times \mathcal{A})$, where $\mathcal{P}$ is the power set.

The domain of firewall configuration rules, $\mathcal{D}$, is defined as follows:

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

Furthermore, two domains can be defined for source and destination addresses, $\mathcal{N}_s$ and $\mathcal{N}_d$, respectively, are defined as:

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

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

Similarly, The configuration co-domain, $\mathcal{D}$, defined as:

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

Domain restriction is applied on firewall configurations in order to obtain a subset of $\mathcal{R}$. The operator $\lhd$ is used to represent domain restriction based on source and destination addresses. First, we formally define domain restriction based on a set of network address pairs, then we refine this definition

further for source and destination addresses.

Domain restriction is defined using the operator $\lhd$ over a set of network addresses, N, where $N \in \mathcal{P}(\mathcal{D}(\mathcal{F}))$, and a set of firewall rules $\mathcal{R}$ as follows:
$N \lhd \mathcal{R}(\mathcal{F}) = \{n \mapsto a | n \in N \land a \in \mathcal{A} \land \exists r \cdot (r \in \mathcal{R} \land r = n \mapsto a)\}$

Domain restriction of firewall configurations for any set of network source and destination addresses,N_s and N_d, where $N_s \in \mathcal{P}(\mathcal{D}_s(\mathcal{F}))$ and $N_d \in \mathcal{P}(\mathcal{D}_d(\mathcal{F}))$, is defined respectively as:
$N_s \lhd \mathcal{R}(\mathcal{F}) = \{n \mapsto a | n \in N \land a \in \mathcal{A} \land \exists r \cdot (r \in \mathcal{R} \land \exists d \cdot (d \in \mathcal{D}_d \land n = \langle s, d \rangle \land r = n \mapsto a))\}$
$N_d \lhd \mathcal{R}(\mathcal{F}) = \{n \mapsto a | n \in N \land a \in \mathcal{A} \land \exists r \cdot (r \in \mathcal{R} \land \exists s \cdot (s \in \mathcal{D}_s \land n = \langle s, d \rangle \land r = n \mapsto a))\}$

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, $\mathcal{R}$.

Co-domain restriction is defined for a chosen set of actions $A_c \in \mathcal{P}(\mathcal{A})$, the operator $\rhd$ is used to represent this operation, which is formally defined as follows:
$A_c \rhd \mathcal{R}(\mathcal{F}) = \{n \mapsto a | n \in N \land a \in \mathcal{A} \land \exists r \cdot (r \in \mathcal{R} \land r = n \mapsto a))\}$

To verify that a given configuration rules, Rn, for a specific network range of addresses N are reselected by the firewall configurations rules $\mathcal{R}$, we define two simple sets of actions $A_a = \{accept\}$ and $A_d = \{deny\}$, then we calculate two sets of rules where this network occurs in their domain, one as source, R_s, and another as destination, R_d, where $R_s = N \lhd \mathcal{R}(\mathcal{F})$ and $R_d = N \lhd \mathcal{R}(\mathcal{F})$. Next, we calculate two sets of rules using co-domain restriction for A_a and A_d by applying co-domain restriction operators on the calculated sets R_s and R_d as follows:

$R_{sa} = A_a \rhd R_s$
$R_{sd} = A_d \rhd R_s$
$R_{da} = A_a \rhd R_d$
$R_{ds} = A_d \rhd R_d$

The configurations is considered consistent if the calculated rules above does not coincide with each others for source addresses and for destination addresses, formally:

$(R_{sa} \cap R_{sd} = \emptyset) \ \land \ (R_{da} \cap R_{dd} = \emptyset)$.

This method is implemented in Event-B framework by providing a model for the firewall configuration rules, then calculating the domain restricted sets. The consistency of the rules is obtained through Event-B invariants.

4. CASE STUDY

4.1. Event-B

Event-B [2] is a formal method for modeling guarded operations. Event-B method provides invariants proofs for state-based systems that are updated by guarded events. Event-B has been shown suitable to perform verification of wide range of systems, but have not been explored for checking properties over firewall configurations. Since Event-B provides a library of operators for set operations in first-order logic, we can use it efficiently for the implementation of our method. On the other hand, the Event-B language [12] allows modeling firewall specifications at different levels of abstraction.

In Event-B [1] [2], the guard is a predicate built on state variables while an action is a generalized substitution that defines a state transition. A guard activates an event when it evaluates to true. A descriptive specification describes what the system does by using a set of variables, constants, properties over constants and invariants which specify properties that the machines state verify.

The correctness of an event-B model is established by proof obligations for the invariants, where each event, including the initialization event, should preserve these invariants. Event-B guards are used to define preconditions that should hold before the event can be executed. The guard and the action of an event defines a relation between variables before the event holds and after. Proof obligations are produced from events in order to state that the invariant condition is preserved. These proof obligations need to be verified in Rodin in order to proof the correctness of the invariants.

Event-B models can be refined with events or variables. We use $R(v, v')$ to represent events that models firewall rules. In the refined model, we will use the same relation, we call it $R_c(v_c, v_c')$ that models the updated set of rules, v_c, in the refined model M_C based on a refined address.

The correctness of firewall policy consistency with regards to the event-B concrete model M_c is achieved through the correctness of the gluing invariant $J(v', v_c')$. Figure 1 below illustrates the link between the abstract and refined model of firewall configurations.

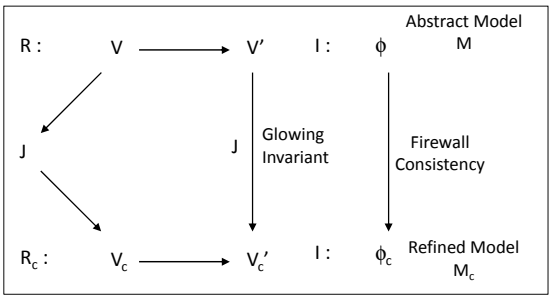

Fig. 1. Modeling Firewalls Two Levels of Abstraction

The abstract and concrete state variables, v and v_c, respectively, are linked together using the gluing invariant $J(v, v_c)$. A_M represents an abstract model for firewalls at the network level, where variable v and invariant $I(v)$ are refined by a concrete model C_M of the firewall at the IP address level, with

[1]This subsection is a description of Event-B method from Event-B references, mainly [2], we follow the same description we presented in a previous work in [13]

variables v_c and gluing invariant $J(v, v_c)$. If $R_A(v, v')$ and $R_C(v_c, v_c')$ are, respectively, the abstract and concrete before-after predicates of the same event, then we obtain the following implication:

$$(I(v) \wedge J(v, v_c) \wedge R_C(v_c, v_c')) \Rightarrow \exists v'.(R_A(v, v') \wedge J(v', v_c'))$$

Under the abstract invariant $I(v)$ and the gluing invariant $J(v, v_c)$, a concrete step $R_C(v_c, v_c')$ can be simulated by an abstract one $R_A(v, v')$ in such a way that the gluing invariant $J(v', v_c')$ is preserved. This leads to the following statement: $I(v) \wedge J(v, v_c) \wedge R_C(v, v_c') \Rightarrow J(v, v_c')$. The gluing invariant, $J(v, v_c)$, is used to represent the consistency of firewall configurations.

Rodin tool [14] is a theorem prover that is designed to run automatically and use a large library of mathematical rules, provided with the system, however, interactive guidance from the user is required for certain proof obligations. We use the Rodin platform in order to define and implement two models for the firewall: an abstract model at the network address level and a refined model at the IP address level. In addition, the consistency of firewalls configurations are defined as Event-B invariants, and then are verified for the refined model by discharging all the proof obligations generated by the tool.

4.2. Verification of Firewall Configurations

An example of a firewall is given in Figure 2, where net_a, net_b, net_c, and net_d, along with their architecture, in addition to the IP addresses are only illustrative. The firewall contains filter rules, when a packet arrives to its entry, the corresponding chain of rules decides if the packet must be dropped or must continue its traversal of the rules. The chain is made up of a list of rules, when inspecting packets chains use the following: the source and destination network, the IP source address, the IP destination address, the protocol, the source port and the destination port.

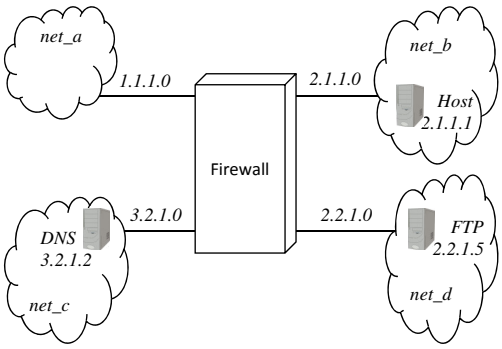

Fig. 2. A Firewall Controlling Traffic in a Network

We consider representing the network a different levels of abstraction, with each refinement of the network, we add more details about the address, and hence, we obtain more concrete firewall rules. In order to illustrate our verification method, we chose a network represented by three zones controlled by a firewall whose initial configuration corresponds to a given set of rules. For the firewall controlled network given in Figure 2, will consider the a security policy, which can be implemented in a firewall configuration that is composed of a set of rules, for instance, consider the following set of rules:

$\mathcal{R}1$: net_a has the right to access net_b.

$\mathcal{R}2$: net_b has the right to access net_c.

$\mathcal{R}3$: net_c has no right to access to net_a.

$\mathcal{R}4$: host in net_b has the right to access DNS and FTP servers.

$\mathcal{R}5$: net_c has no right to access to FTP server in net_d.

The above firewall rules can contain any of the following parameters: Network, IP address, protocol or and port. At the first level of abstraction, we consider network parameters, where we abstract rules to be at the network level. The abstract rules are contained in the concrete ones, an Event-B model is defined for this level of abstraction along with certain invariants to check the consistency of the firewall configuration.

Abstraction of the above set of rules results in an abstract configuration at the network level. Note that this abstraction may result in contradiction in case certain part of one network is granted access to an outer network, and another part of the same network is denied access to that outer network. Abstraction here will definitely result in contradiction between the abstract rule and one of the concrete rules. To overcome this, we consider both rules result in an undefined action. This issue, however, will be resolved at the refined level when we consider the model at the IP address level where both cases can be defined with the appropriate action.

In the first step we will present an abstract model of the above policy at the network level. We define two types *ACTION* and *NET*, the first type define behavior of the firewall on filtered packets: *ACCEPT* or *DENY*. In order to have a complete specification, which is necessary for proof obligations discharge, we assume that an there is always a default last rule that is either *ACCEPT* or *DENY*.

CONTEXT X0
SETS
ACTION NET
CONSTANTS
ACCEPT DENY
net_a net_b net_c net_d
AXIOMS
axm1 : $partition(ACTION, \{ACCEPT\}, \{DENY\},)$
axm2 : $partition(NET, \{net\_a\}, \{net\_b\}, \{net\_c\}, \{net\_d\})$
axm3 : $DENY \neq ACCEPT$
usu
END

MACHINE F0
SEES X0
VARIABLES
RULE
INVARIANTS
inv7 : $RULE \in NET \times NET \rightarrowtail ACTION$

EVENTS

Initialisation

begin

act2 : $RULE :=$
$\{(net\_a \mapsto net\_b) \mapsto ACCEPT,$
$(net\_b \mapsto net\_c) \mapsto ACCEPT,$
$(net\_c \mapsto net\_a) \mapsto DENY,$
$(net\_b \mapsto net\_d) \mapsto ACCEPT,$
$(net\_c \mapsto net\_d) \mapsto DENY\}$

end

END

In this Event-B model, the set *RULE* is defined as a total injection function from the set of $NET \times NET$ into the set *ACTION*. The actual rules of the firewall configurations are defined in the initialization event, where the address of every rules is abstracted into the network address. This may result in merging two or more rules into one single rule.

This model is refined by defining the network address NET over the range of possible IP addresses in the axioms below. In order to make the variants below readable and simple, we assume IP addresses are of class A only, however, the invariants can simply be extended to include other classes by adding the constraints below.

Firewall configuration is defined at this level by mapping every pair of addresses(source and destination) into its possible action throughout the term $RULE$ below. Rules are checked for consistency in Rodin by evaluating the invariant $RuleCheck$ below for the rules map that defines the firewall configurations. This invariant generates a number of proof obligations were discharged using Rodin proof control.

CONTEXT X1

CONSTANTS

RN NET

AXIOMS

axm1 : $RN \in 0..255$

axm2 : $NET \in RN \times RN \times RN \times RN$

END

MACHINE F1

SEES X1

VARIABLES

RULE

INVARIANTS

inv1 : $RULE \in NET \times NET \rightarrowtail ACTION$

RuleCheck : $\forall src1, src2, dst1, dst2 \cdot src1 \in NET \land$
$src2 \in NET \land dst1 \in NET \land dst2 \in NET \land$
$(RULE(src1 \mapsto dst1) = ACCEPT \land$
$RULE(src2 \mapsto dst2) = DENY) \Longrightarrow$
$\neg(src1 = src2 \land dst1 = dst2) \land$
$((NetAdd(src1) = \top) \lor (NetAdd(src1) = \top \land$
$SUBNET(src1) \neq SUBNET(src2))) \land$
$((NetAdd(src2) = F) \lor (NetAdd(src2) = \top \land$
$SUBNET(src1) \neq SUBNET(src2))) \land$
$(NetAdd(dst1) = \top) \lor (NetAdd(dst1) = \top \land$
$SUBNET(dst1) \neq SUBNET(dst2))) \land$
$((NetAdd(dst2) = F) \lor (NetAdd(dst2) = \top \land$
$SUBNET(dst1) \neq SUBNET(dst2)))$

EVENTS

Initialisation

begin

act1 : $RULE := \{$
$(1 \mapsto 1 \mapsto 1 \mapsto 0 \mapsto 2 \mapsto 1 \mapsto 1 \mapsto 0) \mapsto ACCEPT,$
$(2 \mapsto 1 \mapsto 1 \mapsto 0 \mapsto 3 \mapsto 2 \mapsto 1 \mapsto 0) \mapsto ACCEPT,$
$(2 \mapsto 2 \mapsto 1 \mapsto 0 \mapsto 1 \mapsto 1 \mapsto 0 \mapsto 0) \mapsto DENY,$
$(2 \mapsto 1 \mapsto 1 \mapsto 1 \mapsto 3 \mapsto 2 \mapsto 1 \mapsto 2) \mapsto ACCEPT,$
$(2 \mapsto 1 \mapsto 1 \mapsto 1 \mapsto 2 \mapsto 2 \mapsto 1 \mapsto 5) \mapsto ACCEPT,$
$(3 \mapsto 2 \mapsto 1 \mapsto 0 \mapsto 2 \mapsto 2 \mapsto 1 \mapsto 5) \mapsto DENY\}$

end

Event *evt1*

any

$w4\ \ w3\ \ w2\ \ w1$

where

grd1 : $w1 = 0 \land w4 \in 0..255 \land w3 \in 0..255 \land w2 \in 0..255$

then

act1 : $NetAdd(w4 \mapsto w3 \mapsto w2 \mapsto w1) := TRUE$

end

Event *evt2*

any

$w4\ \ w3\ \ w2\ \ w1$

where

grd1 : $w1 = 0..255 \land w4 \in 0..255 \land w3 \in 0..255 \land w2 \in 0..255$

then

act1 : $SUBNET(w4 \mapsto w3 \mapsto w2 \mapsto w1) := w4 \mapsto w3 \mapsto w2$

end

END

Next step is to implement the set primitives and their domain restriction operators in Event-B, these operators are embedded in the platform, therefore, we directly use them to implement our model. The consistency of firewall configurations is defined using the invariant, *inv3*, in Rodin platform as shown below, the tool generates proof obligations that were successfully discharged using Event-B proof control.

MACHINE F2

SEES X2

VARIABLES

n Rs Rd Aa Ad Rsa Rsd Rda Rdd

INVARIANTS

inv1 : $Rs \in RULE \land Rd \in RULE \land Rsa \in RULE \land$
$Rsd \in RULE \land Rda \in RULE \land Rdd \in RULE$

inv2 : $Aa \in ACTION \land Ad \in ACTION$

inv3 : $Rsa \cap Rsd = \emptyset \ \land \ Rda \cap Rdd = \emptyset$

EVENTS

Initialisation

begin

act1 : $Aa := \{ACCEPT\}$

act2 : $Ad := \{DENY\}$

end

Event *evt1*

any

n

where

grd1 : $n \in NET$

then

act1 : $Rs := n \lhd RULE$

act2 : $Rd := n \rhd RULE$

act3 : $Rsa := Aa \rhd Rs$

act4 : $Rsd := Ad \rhd Rs$

act5 : $Rda := Aa \rhd Rd$

act6 : $Rdd := Ad \rhd Rd$

end

END

The results achieved here are important because our method allows modeling the firewall configurations at different levels of abstraction. We presented a high level model at an abstract network address level. This model is further refined in order to include more details about the addresses in firewall rules, while preserving the correctness of the invariants. The verification of a more refined model will be straight forward, and will require a refinement of this model based on addresses by including protocol types or port numbers. This is going to be covered in future work. In order to make this method more appealing and applicable on industrial size firewalls, an interface is required in order to map firewall rules into Event-B data structure model, the semantics of this translation can be deduced using our model, and the interface can provide automatic translation from firewall configuration rules into Event-B syntax. This issue will be addressed in the future work.

5. CONCLUSION

In this paper we present a formal model for firewall configuration rules based on domain restriction. This model is used to formally verify the consistency of the configuration rules in firewalls. We use the Event-B based invariant checking to implement our method to be able to conduct verification on firewall configurations. We illustrate our method on a case study by modeling firewall configurations at the network level of abstraction, then, we refine this model by considering the network at the IP address level.

Firewall configuration rules are embedded in Event-B, the consistency of firewall configurations is defined in Event-B invariants, then Rodin firstorder theorem prover is used to to proof the consistency of this configuration by proving each of the proof obligations automatically.

The advantage of our method is the ability to model firewall configurations at different levels of abstraction. A high level model representing firewall rules at an abstract network address level is used first. This model is further refined by using IP network addresses in firewall rules, while preserving the correctness of the invariants, and hence the consistency of firewall configurations.

As future work, we will provide a formal proof of the correctness of the method by showing the completeness and soundness of the presented model. In addition we intend to use the same method to proof firewall consistency at more refined levels by allowing rules at the protocol and port number levels.

6. REFERENCES

[1] A. Gupta, "Formal Hardware Verification Methods: A Survey," *Formal Methods in System Design*, vol. 1, no. 2-3, pp. 151–238, 1992.

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

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

[4] N. Ben Youssef, A. Bouhoula, and F. Jacquemard, "Automatic verification of conformance of firewall configurations to security policies," in *Symposium on Computers and Communications*. july 2009, pp. 526–531, IEEE Computer Society Press.

[5] A. Brucker, L. Brügger, and B. Wolff, "Model-Based Firewall Conformance Testing," in *Int. Conf. on Testing of Software and Communicating Systems*, Berlin, Heidelberg, 2008, vol. 5047 of *Lecture Notes in Computer Science*, pp. 103–118, Springer-Verlag.

[6] F. Cuppens, N. Cuppens-Boulahia, T. Sans, and A. Miège, "A Formal Approach to Specify and Deploy a Network Security Policy," in *Formal Aspects in Security and Trust*. 2004, vol. 173 of *Lecture Notes in Computer Science*, pp. 203–218, Springer-Verlag.

[7] A.X. Liu, "Formal Verification of Firewall Policies," in *IEEE Int. Conf. on Communications*. May 2008, pp. 1494–1498, IEEE Computer Society Press.

[8] L. Yuan, H. Chen, J. Mai, C. Chuah, Z. Su, and P. Mohapatra, "FIREMAN: a Toolkit for Firewall Modeling and Analysis," in *Symposium on Security and Privacy*. may. 2006, pp. 199–213, IEEE Computer Society Press.

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

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

[11] D. Chapman and E. Zwicky, *Building Internet Firewalls. 2nd Ed.*, Orielly & Associates Inc., 2000.

[12] C. Metayer, J. Abrial, and L. Voisin, "RODIN Deliverable 3.2: Event-B Language," Tech. Rep. Project IST-511599, School of Computing Science, University of Newcastle, UK, 2005.

[13] A. Gawanmeh, L. Jemni Ben Ayed, and S. Tahar, "Event-B based Invariant Checking of Secrecy in Group Key Protocols," in *Local Computer Networks*. October 2008, pp. 950–957, IEEE Computer Society Press.

[14] *Rodin Platform*, "http://www.event-b.org, 2010," .