EPiC Series in Computing

Volume 45, 2017, Pages 68–76

SCSS 2017. The 8th International Symposium on Symbolic Computation in Software Science 2017

Automatic detection and correction of firewall misconfigurations- A formal approach

Amina Saâdaoui, Nihel Ben Youssef Ben Souayeh and Adel Bouhoula

Digital Security Research Unit
Higher School of Communication of Tunis (Sup'Com)
University of Carthage, Tunisia
{amina.saadaoui, nihel.benyoussef, adel.bouhoula}@supcom.tn

Abstract

Firewall has been at the center of intense research in the last decade owing to the increase of malicious attacks on networks. Constant updating of the firewall configuration by modifying, adding and removing rules increases the complexity of the configuration resulting in overlapping and often conflicting filtering rules. As a consequence, the set of filtering rules becomes unreliable and contains multiple misconfigurations creating ambiguity in classification of new traffic, not only affecting the performance of the firewall, but also putting the system in a vulnerable position. Manual management of this problem can be overwhelming and potentially inaccurate. Therefore, there is a need of automated methods to analyze, detect and fix misconfigurations. The objective of our work is to propose (1) a new formal approach to discover effective firewall configurations errors, (2) an optimal and automatic method with the minimum number of operations to correct these misconfigurations in both centralized firewalls and firewalls in a distributed environment and (3) a tool that implements proposed techniques and significantly helps user in discovering and resolving firewall misconfigurations.

1 Introduction

The complexity of networks is constantly increasing, as it is the size and complexity of firewall configurations. These Firewalls are the first line of defense for the enterprise network, they examine the traffic of network against an ordered list of filtered rules, generally, defined by network administrator according to the global security policy. Overtime, the exponential growth in network traffic, services and applications has led to a growth in rule-sets size and a growth in firewall complexity. In a typical enterprise network, a single firewall may be configured with thousands of rules. Moreover, in multi-firewall environment, with hundreds of firewalls, it is difficult to verify detect and correct manually misconfigurations that can arise between different rules.

In our work, we consider the following problem: In a multi-firewall environment network, where each firewall can accumulate hundreds of changes over the years, how can we analyze detect and fix firewalls misconfigurations? To deal with firewall rules analysis problem, many

solutions have been proposed. In [12, 11, 10], detection algorithms presented deal only with pairwise filtering rules. In such a way, some other classes of configuration anomalies could be uncharted. In [3], authors present a model for detecting anomalies in a small network. Authors in [7] present a formal automata-based approach to detect rules anomalies. Abbes et al. present in [1] a method that allows representing filtering rules in a tree data structure and to update it according to the evolution of the firewall configuration. Nevertheless, the drawback of these approaches [3, 7, 1, 17] is that they did not propose a method to correct automatically discovered misconfigurations. Authors in [15] use the concept of Relational Algebra and a 2D box model to identify anomalies. Also, there are research works focusing on creating automated solutions to detect configuration inconsistencies firewalls through formal verification [4], [9]. In [5], Gawanmeh et al. presented a formal model for firewall configuration rules based on domain restriction. This model is used to define an algorithm to formally verify the consistency of the configuration rules in firewalls. This work offers the ability to characterize firewall configurations at different levels of abstraction. In [20] authors present an anomaly identification and resolution approach their approach is similar to the work of Hu et al [13] who propose a framework that facilitates the resolution of anomalies by considering the analysis of relations between all rules in the firewall configuration. The proposed idea to resolve anomalies is based on calculating a risk level that permits users, in some cases, to manually select the appropriate strategies for resolving the conflict. In such a way, the administrator can make wrong choices. FIREMAN [19] is a static analysis toolkit to check anomalies in firewalls. It can only show there are anomalies between one filtering rule and preceding rules without identifying all rules involved in the anomaly. Adao et al. [2] propose Mignis, a declarative policy language to specify a Linux firewall, Netfilter configurations.

Some other approaches take into account the requirements of the security policy and try to verify the firewall configuration with the security policy. For example, Matsumoto and Bouhoula [18] propose a SAT based approach to verify firewall configurations with respect to the security policy. Ben Youssef and Bouhoula [16] propose a formal method for verifying automatically that a firewall configuration is conform to a security policy. But, in the case of the presence of anomalies and non-conformity with the security policy, these two methods does not propose a method to correct the detected anomalies. In this work, we propose a new approach to discover and fix misconfigurations in a multi-firewall environment.

This paper is organized as follows: Section 2 overviews the formal representation of firewall configurations and security policies and details FDD structure. In section 2.4, we articulate our approach to detect and correct firewall misconfigurations and to discover and remove superfluous rules. In section 4, we present the formalism used in our work. Finally, we present our conclusions and discuss our plans for future work.

2 Preliminaries

In what follow, we define, formally, some key notions.

2.1 Firewall configuration

A simple firewall configuration is a finite sequence of filtering rules of the form $FR = (r_i \Rightarrow A_i)_{0<i<N+1}$. These rules are tried in order, up to the first matching one. A filtering rule consists of a precondition r_i which is a region of the packet's space, usually, consisting of source address, destination address, protocol and destination port. Each right member A_i of a rule of FR is an

action defining the behavior of the firewall on filtered packets: $A_i \in \{accept, deny\}$. The last rule of a firewall configuration is called the default policy of the firewall. In fact, each firewall implements a positive or negative policy. If it is positive, the default decision is to deny-all i.e., deny a packet when any configuration rule applies. By contrast, the negative policy will accept-all.

2.2 Security Policy

A security policy SP is presented as a finite unordered set of directives, as showed in the example of the Introduction, defining whether packets are accepted or denied. We consider also two sets, SP_accept and SP_deny where SP_accept consists of packets accepted to pass through the set of directives SP and SP_deny is the subset of denied packets. In this work we suppose that SP is consistent, i,e. $SP\_accept \cap SP\_deny = \emptyset$.

2.3 Firewall Decision Diagram (fdd) of a simple firewall

The firewall decision diagram (fdd) as defined in [6, 8] is an acyclic and directed graph that has the following properties: There is exactly one node in fdd that has no incoming edges. This node is called the root of fdd. The nodes in fdd that have no outgoing edges are called terminal nodes. fdd is the union of direct paths dp_i. The algorithm used to construct an fdd is detailed in [6, 8]. So we have:

$$fdd = \bigcup_{i(i:1 \to m)} dp_i.$$
$$dp_i = dp_i.srce \wedge dp_i.protocol \wedge dp_i.dest \wedge dp_i.port \wedge dp_i.rules \wedge dp_i.action.$$

- $dp_i.src$ is the range of source address represents by the direct path dp_i.
- $dp_i.dst$ is the range of destination address represents by the direct path dp_i.
- $dp_i.port$ is the range of port number represents by the direct path dp_i.
- $dp_i.protocol$ is the range of protocols represented by the direct path dp_i.
- $dp_i.rules$ is the set of rules from the firewall configuration that match the domain of packets represented by this direct path, $dp_i.rules = \{r_{ki}\}_{(k:1 \to l)}$, where r_{1i} is the first rule in the firewall configuration applied on the domain of dp_i. The action of this direct path is the action applied by r_{1i}.
- $dp_i.action$= the action of this direct path dp_i.

We define a variable called $dp_i.field$ which represents different fields of a direct path, where field could be (src, dst, port, protocol). Each $dp_i.field$ (i.e., $dp_i.src$, $dp_i.dst$, $dp_i.port$, $dp_i.protocol$) satisfies the following two conditions for each node in the firewall decision diagram:

- Consistency: $dp_i.field \cap dp_j.field = \emptyset$, $i \neq j$.
- Completeness: $\bigcup_{i \in [1,n]} dp_i.field = *$ ($*$ means total field's domain since default rule is deployed in each firewall).

Fig. 1 shows an fdd of a simple firewall configuration.

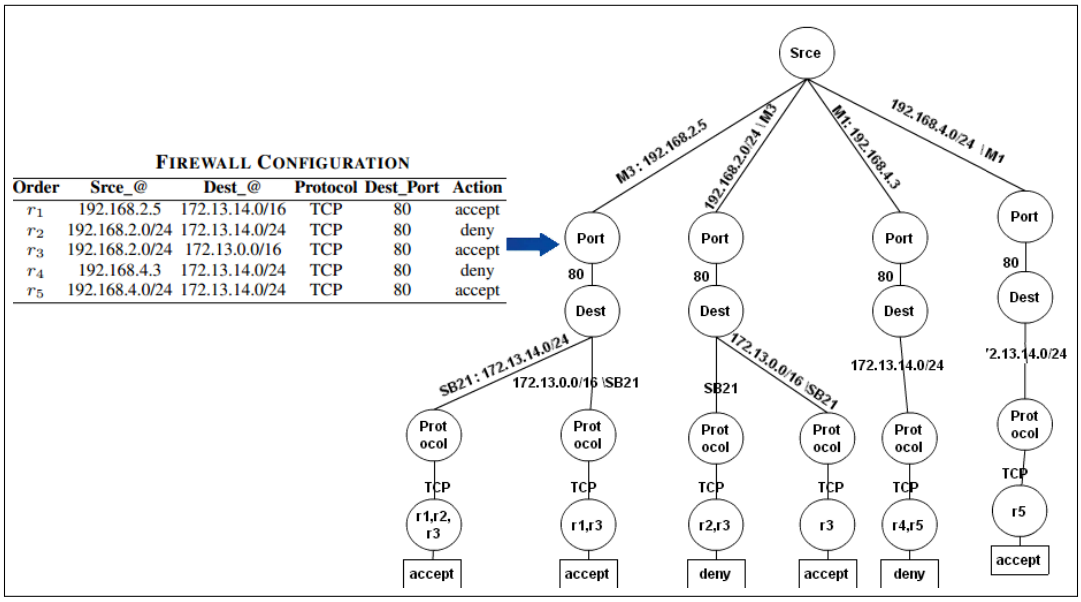

Figure 1: Firewall Decision Diagram- Firewall1

2.4 FDD of a path in a distributed environment

A network path $path_i[src, dst]$ is composed by an ordered set of firewalls through which the traffic flow from the source src to the destination dst. $path_i = \{fc_j, n <= j <= m\}$. Let $Paths$ be the set of all possible paths in our network. $Paths = \{path_i, 1 <= i <= k\}$.

A firewall decision diagram of a path $path_i$ is constructed using the collection of rules of different firewalls fc_j belonging to this path. Therefore, The firewall decision diagram of the set $Paths$ of our network could be represented as follows: $FDD(Paths) = FDD = \bigcup_{\{0<i<N+1\}} fdd_i$, where each fdd_i is the firewall decision diagram of the path $path_i$, so FDD is the union of fdd_i of each path in the network. Given that the default rule is applied by each firewall each node in each fdd_i satisfies the consistency and the completeness conditions. We construct fdd_i by using the same algorithm depicted in 2.3 for the collection of rules of each $path_i$. The proprieties already defined for a direct path in a simple firewall remains the same, only for sets $dp_i.rules$ and $dp_i.action$. In fact, we have to precise for each rule the firewall that belongs to it. Therefore, we define direct path $dp_j \in fdd_i$ as follows:
$dp_j = dp_j.srce \wedge dp_j.dest \wedge dp_j.port \wedge dp_j.protocol \wedge dp_j.rules \wedge dp_j.action$ where $dp_j.rules = \{r_{h_j}^k\}$ here k is the index of the each firewall through which the traffic flow in the path $path_i$. The action of each direct path depends on the actions of each first rule handled by this direct path from each firewall in this path, so we have:

- $dp_j.action = accept$ if $\forall r_{1_j}^k \in dp_j.rules$, $action(r_{1_j}^k) = accept$.
- $dp_j.action = deny$ if $\exists! r_{1_j}^k \in dp_j.rules$, $action(r_{1_j}^k) = deny$.

3 Approach Overview

3.1 Discovering and removing superfluous rules

A rule is superfluous if and only if it can be removed without altering the firewall behavior if we have a simple firewall and the path configuration behavior if we are in a distributed environment. To verify if a rule is superfluous, we need to ensure that removing it from each direct path will not affect the action of this path.

Definiton : A rule is considered to be superfluous in a simple firewall, if this rule exists in the set of rules handled by a direct path then this rule is shadowed (i.e. it is not the first rule to be applied in this direct path) or redundant to the second rule in this path. *Formally*, A rule r_i is superfluous iff $\forall dp_j \in fdd$, if $r_i \in dp_j.rules$ then r_i verifies one of these two conditions :

1. $r_i \neq r_{1j}$.
2. $r_i = r_{1j}$ and $action(r_i) = action(r_{2j})$.

To address this challenge, we use the inference system shown in Fig. 2. The rules of this inference system are applied to three components (fc, fc_f, fdd), the first component fc is the initial firewall configuration, the second component fc_f is the updated version of fc by removing all superfluous rules and the third component fdd is the set of direct paths that represents relations between fc rules. **Remove** is the main inference rule in this inference system. It deals with each rule r_i from the firewall configuration fc. Applying this inference rule implies updating the set of rules fc_f by removing superfluous rules. The inference rule **Stop** is applied when we parse all the filtering rules of fc. Thus, we conclude that this process provides configuration optimization, which reduces the firewall rule size and subsequently improves its performance.

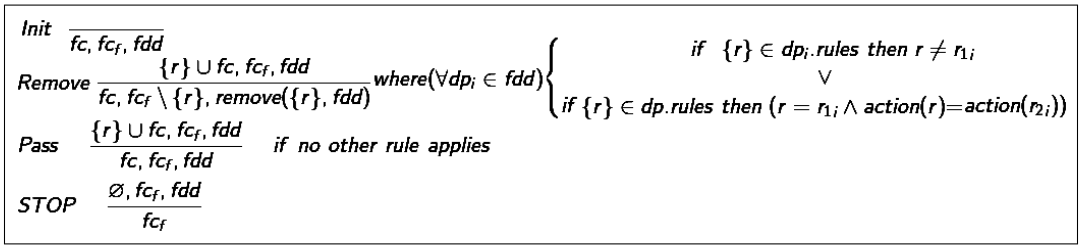

Figure 2: Discovering and removing superfluous rules in a simple firewall

3.2 Misconfigurations classification and discovering

We have two types of misconfigurations: **Partial** and **Total** misconfigurations: (TMC) A direct path $DP_i \in FDD$ is **totally** misconfigured iff the packets mapped by this path apply a different action as applied in the security policy on these packets.(PMC) A direct path $DP_i \in FDD$ is **partially** misconfigured iff *some* packets mapped by this path apply a different action as applied in the security policy on these packets, Formally :

TMC : A direct path $DP_i \in FDD$ is **totally** misconfigured iff $\exists r_{mi} \in DP_i$ where $action(r_{mi}) \neq DP_i.action$ and *all* the packets mapped by this path apply a different action as applied in SP on these packets.

PMC : A direct path $DP_i \in FDD$ is **partially** misconfigured iff $\exists r_{mi} \in DP_i$ where $action(r_{mi}) \neq DP_i.action$ and *some* packets mapped by this path apply a different action as applied in SP on these packets.
Once all misconfigurations have been discovered using an inference system, we can start their correction process.

In Figure 3 we propose an inference system that presents necessary and sufficient steps to discover Total and partial misconfigurations. The rules of this inference system apply to triple (FDD, TMC, PMC). The first component FDD represents the direct paths extracted from the firewall configuration as explained in the previous section, $FDD = \{DP_i\}_{(i:1 \to n)}$. The second component TMC is the set of extracted *Total misconfigurations* and the third component PMC represents the set of partial misconfiguration. **Extract_TMC** and **Extract_PMC** are the main inference rules for the inference system. The first one detects Total misconfigurations. It deals with each DP_i from FDD and verifies if this DP_i applies totally the same action in the firewall configuration as applied in SP, so we test if this DP_i is included in the set of $SP_{!(action(r_{1i}))}$, if it is the case, DP_i is considered to be a total misconfiguration, because the action of DP_i, which is equal to $action(r_{1_i})$, is different from the action applied by SP on this direct path, so we add DP_i to TMC. The same for the second, **Extract_PMC**, but here we will extract partial misconfigurations. In fact, for each DP_i we test if a part from the domain of this direct path apply a different action on the packets matched by this domain as applied in SP, if it is the case we will add DP_i to the set of partial misconfigurations PMC. The inference rule **Pass** is applied when DP_i does not contain an anomaly between its rules, or when it contains an anomaly and the action applied on this direct path in FDD is same action undertaken by the security policy. So, in this case this anomaly is considered to be *intentional* and not a misconfiguration. Hence, the repeated application of these inference rules ensures the extraction of all misconfigurations (Partial or Total) from the firewall configuration. The rule **Stop** is applied when we parse all the direct paths of FDD.

Our objective is to correct each misconfiguration with the minimum number of modifications and with the minimum number of generated rules. In our approach, for each step, we try to correct a misconfiguration (total or partial). To determine which correction method should be used at each case; we test if the condition of each correction technique is verified. In fact, we parse the set of total and partial misconfigurations then we try to correct them by using one of the inference systems detailed in the next section. For total misconfigurations, we can use the *delete-rule* inference system, if this method could not be applied then we use the *modify-action* inference system and if the condition of the last method is not verified we use the *swap-rules* inference system. If none of them could be applied, we apply the *field-modification* inference system which is also used to correct partial misconfigurations. Then after fixing all misconfigurations we generate the new set of rules using this key property:

- Each rule r_k from the firewall configuration could be represented as follow:
  $r_k = \bigcup_i DP_i \;\; where \;\; r_k \in DP_i.R$

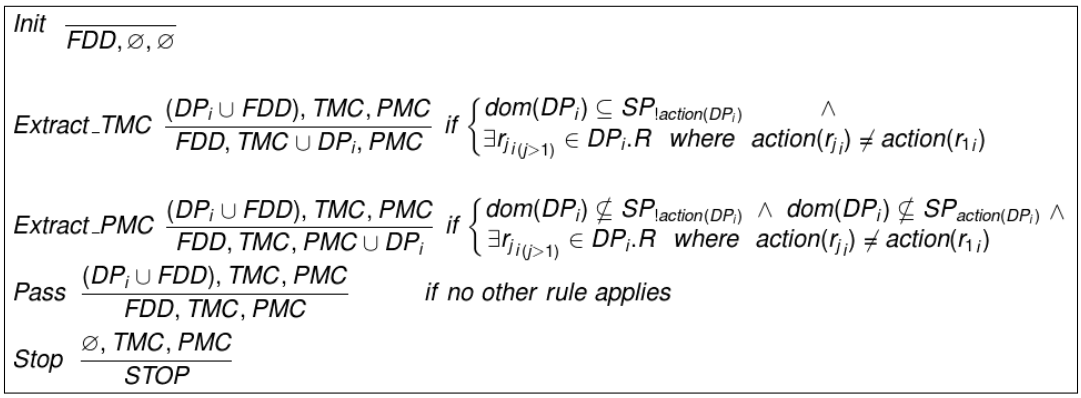

Figure 3: Inference system for Discovering Misconfigurations

3.3 Misconfigurations correction techniques

- **Remove-rule Method** : First, we try to correct Total misconfigurations by removing misconfigured rules using an inference system. We can remove a rule only if this rule exists in a decision path as a first rule, then this path is totally misconfigured and the action of second rules in these paths are different from the actions of first rules. So if we remove this rule we will correct all these misconfigurations.

- **Modify rules Method** : Due to the first-match semantics, modifying the action of a rule affect the semantic of the firewall. Thus, after changing the action of a rule we should not generate new misconfigurations. So, we should verify first if **all** the direct paths that have this rule as a first rule are totally misconfigured. If it is the case, we can change the action of the rule under consideration and using this one modification we will correct all misconfigured direct paths that have this rule as a first one.

- **Swap rules Method** : Changing the order of two rules by swapping them in a firewall configuration affects its functionality. Therefore, before swapping two rules, we need to test and to verify if this modification will generate new misconfigurations between one of the swapped rules and other rules. Also we should verify if this action will correct misconfigurations that exist in paths presented by these two rules.

- **Field modification Method** : To fix partial misconfigurations we use an inference system that allows to divide each partially misconfigured direct path DP_i, into two sets, the first is the set that have the *correct* action as defined in the security policy. And the second DP_i' represents the subset of DP_i that should be fixed by removing all rules having the same action as the first rule from this set DP_i'. So using this method we fix all partial misconfigurations.

4 Initial Results

In order to better assess the effectiveness of our approach, we proved formally the correctness and completeness of our inference systems. Also we implemented the techniques and inference systems in a software tool, using a Boolean satisfiability (SAT) based approach. This approach

reduces the verification problem into Boolean formula and checks its satisfiability. Therefore, we have chosen the Java developing language. On the other hand, the verification of the satisfiability of Boolean expressions is performed using Limboole [14]. This tool allows to check satisfiability respectively tautology on arbitrary structural formulas and not just satisfiability for formulas in conjunctive normal form (CNF), and can handle large set of non-quantified Boolean clauses in reasonably good time. Our formalism for specifying the Firewall configuration and the security policy is a Boolean-based specification language. For example, the functional mapping of precondition r_i components into Boolean variables is shown below :

- We model the source and destination IP addresses with 32 Boolean variables each, namely, (s0, s1, ., s31) and (d0,d1, ..d31);
- Address ranges with masks can be reduced by bit-wise ANDing the masks with the base addresses;
- We have 32 different protocols so protocol type can be reduced using 5 variables (p0,p1...p4);
- We have 65356 different ports number, so in our formalism port numbers are mapped into 16 Boolean variables, namely (n0,...,n15).

The example shown in Figure 4 how we reduce an ip_address using our formalism:

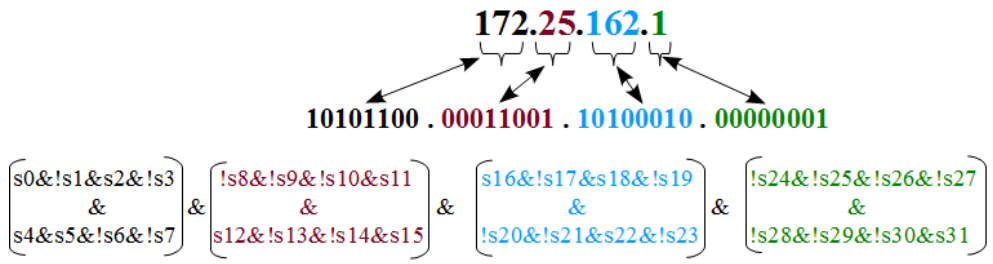

Figure 4: IP address reduction

5 Conclusion

The prevalent use of firewalls in network security emphasizes the importance of efficient and optimal configuration. This paper describes two problems. The first is the problem of rule-sets optimization for simple and distributed firewalls and presents our approach to configuration optimization by removing rules that are no longer needed (called superfluous). The second is firewall misconfiguration discovering and correction. While the current approach focuses on the analysis of Firewall rules, in our future work, we plan to analyze other security equipment configurations.

References

[1] Tarek Abbes, Adel Bouhoula, and Michael Rusinowitch. Detection of firewall configuration errors with updatable tree. *International Journal of Information Security*, pages 1–17, 2015.

[2] Pedro Adão, Claudio Bozzato, G. Dei Rossi, Riccardo Focardi, and Flaminia L. Luccio. Mignis: A semantic based tool for firewall configuration. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 351–365, 2014.

[3] Tawfiq SM. Barhoom and Emad KH. Elrayyes. Model for strengthening accuracy through detection of anomalous firewall policy rules. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(12):7116–7124, 2014.

[4] Amjad Gawanmeh. Automatic verification of security policies in firewalls with dynamic rule sequence. In *11th International Conference on Information Technology: New Generations, ITNG 2014, Las Vegas, NV, USA, April 7-9, 2014*, pages 279–284, 2014.

[5] Amjad Gawanmeh and Sofiène Tahar. Modeling and verification of firewall configurations using domain restriction method. In *6th International Conference for Internet Technology and Secured Transactions, ICITST 2011, Abu Dhabi, UAE, December 11-14, 2011*, pages 642–647, 2011.

[6] Mohamed G. Gouda and Alex X. Liu. Structured firewall design. *Computer Networks Journal (Elsevier)*, 51(4):1106–1120, March 2007.

[7] Swati S. Kachare and P.K. Deshmukh. Firewall policy anomaly management with optimizing rule order. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, 4(2):201–205, 2015.

[8] Alex X. Liu and Mohamed G. Gouda. Diverse firewall design. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 19(8), 2008.

[9] Majda Moussa, Hakima Ould-Slimane, Hanifa Boucheneb, and Steven Chamberland. A formal framework for verifying inter-firewalls consistency. In *IEEE Symposium on Computers and Communications, ISCC 2014, Funchal, Madeira, Portugal, June 23-26, 2014*, pages 1–7, 2014.

[10] E.S. Al-Shaer and H.H. Hamed. firewall policy advisor for anomaly discovery and rule editing. In Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on, pages 17–30, March 2003.

[11] Frédéric Cuppens, Nora Cuppens-Boulahia, and Joaquin Garcia Alfaro. detection and removal of firewall misconfiguration. In CNIS IASTED, Phoenix, AZ, USA novembre, 2005.

[12] Ehab S. Al-Shaer and Hazem H. Hamed. modeling and management of firewall policies. IEEE Transactions on Network and Service Management, 1(1):2–10, 2004.

[13] Hongxin Hu, Gail-Joon Ahn, and Ketan Kulkarni. detecting and resolving firewall policy anomalies. IEEE Transactions on Dependable and Secure Computing, 9(3):318–331, 2012.

[14] limboole sat solver, 2012.

[15] Naveen Mukkapati and Ch.V.Bhargavi. detecting policy anomalies in firewalls by relational algebra and raining 2d-box model. IJCSNS International Journal of Computer Science and Network Security, 13(5):94–99, 2013.

[16] Nihel Ben Youssef, Adel Bouhoula, and Florent Jacquemard. automatic verification of conformance of firewall configurations to security policies. In ISCC, pages 526–531, 2009.

[17] Padmalochan Bera, Santosh K. Ghosh, and Pallab Dasgupta. integrated security analysis framework for an enterprise network - a formal approach. IET Information Security, 4(4):283–300, 2010.

[18] Soutaro Matsumoto and Adel Bouhoula. automatic verification of firewall configuration with respect to security policy requirements. In CISIS, pages 123–130, 2008.

[19] Lihua Yuan, Jianning Mai, Zhendong Su, Hao Chen, Chen-Nee Chuah, and Prasant Mohapatra. fireman: A toolkit for firewall modeling and analysis. In Proceedings of the 2006 IEEE Symposium on Security and Privacy, SP '06, pages 199–213, Washington, DC, USA, 2006. IEEE Computer Society.

[20] Jagruti Kailas Patil Mansi Prabhakar Kini Renuka Nagpure, Pranali Jaypraksh Dhuri and Aarti Jaywant Patil. Detection and resolution of firewall policy anomalies. *THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE*, 3(2):59–62, 2015.