POLITECNICO DI TORINO
Repository ISTITUZIONALE

An Optimized Approach for Assisted Firewall Anomaly Resolution

Original
An Optimized Approach for Assisted Firewall Anomaly Resolution / Bringhenti, Daniele; Seno, Lucia; Valenza, Fulvio. - In: IEEE ACCESS. - ISSN 2169-3536. - ELETTRONICO. - 11:(2023), pp. 119693-119710. [10.1109/ACCESS.2023.3328194]

Availability:
This version is available at: 11583/2983435 since: 2023-12-13T13:49:07Z

Publisher:
IEEE

Published
DOI:10.1109/ACCESS.2023.3328194

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

11 November 2024

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

An Optimized Approach for Assisted Firewall Anomaly Resolution

DANIELE BRINGHENTI[1], LUCIA SENO[2], and FULVIO VALENZA[1]
[1]Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy (e-mail: daniele.bringhenti@polito.it, fulvio.valenza@polito.it)
[2]Cnr Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni (CNR-IEIIT), Padova, Italy (e-mail: lucia.seno@cnr.it)

Corresponding author: Daniele Bringhenti (e-mail: daniele.bringhenti@polito.it).

ABSTRACT The security configuration of firewalls is a complex task that is commonly performed manually by network administrators. As a consequence, among the rules composing firewall policies, they often introduce anomalies, which can be classified into sub-optimizations and conflicts, and which must be solved to allow the expected firewall behavior. The severity of this problem has been recently exacerbated by the increasing size and heterogeneity of next-generation computer networks. In this context, a main research challenge is the definition of approaches that may help the administrators in identifying and resolving the anomalies afflicting the policies they write. However, the strategies proposed in literature are fully automated, and thus potentially dangerous because the error-fixing process is not under human control. Therefore, this paper proposes an optimized approach to provide assisted firewall anomaly detection and resolution. This approach solves automatically only sub-optimizations, while it interacts with human users through explicit queries related to the resolution of conflicts, as their automatic resolution may lead to undesired configurations. The proposed approach also reduces the number of required interactions, with the aim to reduce the workload required to administrators, and employs satisfiability checking techniques to provide a correct-by-construction result. A framework implementing this methodology has been finally evaluated in use cases showcasing its applicability and optimization.

INDEX TERMS firewall, policy anomaly management, policy-based systems

I. INTRODUCTION

In modern computer networks, firewalls are pervasively deployed to protect network assets from cyber threats by blocking undesired and malicious traversing traffic [1]. Firewalls can be either dedicated hardware devices or software programs running on general-purpose servers, relying on traditional operating systems or virtualized environments, as in the case of networks based on the Network Functions Virtualization (NFV) paradigm. Firewalls are commonly placed at the edge of the protected network, to act as a first line of defense against attacks coming from the outside. Also, increasingly often, firewalls are deployed within the protected network, following the *defense in depth* principle. The latter design allows to differentiate among globally and locally enforced security policies, enables fine-grained control of traffic flowing within the network and fast attack response, as well as limits the propagation of successful both outside and inside attacks.

The way a firewall filters traversing traffic is dictated by its configuration, establishing which packets must be forwarded on the next hop toward their destination, and which ones must be discarded as undesired or malicious. Writing and maintaining firewall configurations is a cumbersome task that is still often performed manually by network administrators, i.e., without the support of automatic or semi-automatic tools. As a consequence, the probability of anomalies (i.e., sub-optimizations and conflicts) affecting firewall configurations is considerably high, especially in the case of complex, heterogeneous networks, characterized by the presence of multiple firewalls with a high number of rules. Firewall misconfigurations, in particular, have proven to recurrently lead to unfortunate events such as data breaches or access control violations. According to a recent Data Breach Investigations Report produced by Verizon, in 2021 82% of breaches were caused by human errors in security configuration [2]. The Verizon analysis clearly shows that traditional manual approaches to firewall configuration reveal inefficient and unreliable and are not the best course of action for modern networks.

For the above reasons, one of the research challenges that

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

has been addressed in the last decade is the development of methodologies and tools for identifying and resolving anomalies afflicting human-written firewall configurations. This problem is quite complex, as multiple anomaly classes must be accounted for, and those highlighting policy sub-optimizations and errors must be detected and fixed. Some state-of-the-art methodologies have been proposed to perform both the detection and resolution tasks automatically. However, fully automated anomaly resolution strategies are potentially very dangerous as the error-fixing process is not under the network administrator control. As an example, if multiple conflicting firewall rules are detected, a fully automated tool may make a decision that is not coherent with the original vision, i.e., the security intents of administrators.

In light of these considerations, this paper proposes an optimized approach to provide assisted resolution of intra-firewall policy anomalies (i.e., anomalies that occur within the same firewall rule list). This approach automates all anomaly resolution operations that do not require external human interventions, i.e., the ones related to sub-optimizations. Instead, it interacts with network administrators through explicit queries related to the resolution of conflicts, which, if automatically addressed, may lead to undesired resulting configurations. The approach is able to remove all rules that are unnecessary in the firewall policy and to reorder the remaining rules to create a final anomaly-free firewall policy. The scope of the proposal lies in intra-firewall policy resolution, because addressing the issue of solving all anomaly types for a rule list is a research problem complex by itself, as also proved by the large number of studies proposed in literature within the same scope.

The main novelties of this proposal are the following ones:

- Differently from state-of-the-art approaches, which are still manual or fully automatic, our proposed approach is semi-automatic. This feature contributes to avoid that the human user manually solves each anomaly, but at the same time it allows him to interact with the fixing process to define his operational guidelines, thus overcoming the previously mentioned limitations of the related literature.
- The proposed approach aims to reduce the workload required to administrators. In fact, the proposed methodology has been designed so as to reduce the number of queries that are actually needed to solve all errors in the original firewall configuration. This feature makes the process swifter and suitable for large firewall configurations where the number of detected anomalies may be consistently high.
- The ordering of the firewall rules composing the final anomaly-free firewall policy is established by employing satisfiability checking, and modeling the problem as a Satisfiability Modulo Theories (SMT) problem. Employing an automated SMT solver, whose algorithms have already been proved correct and sound, provides assurance about the outcome of this final operation, which can be said to be correct by construction. Moreover, as reported in the literature [3], the output of this mathematical formulation does not require the application of any other a-posteriori formal verification, and state-of-the-art open-source SMT solvers can efficiently solve SMT problem instances in polynomial times on average.

The remainder of this paper is structured as follows. Section II discusses related work, highlighting the differences with respect to our proposal. Section III provides an overview of the proposed approach for assisted firewall anomaly detection and resolution. Section IV describes the firewall and rule relationship models, which are at the basis of the proposed methodology. Section V discusses the strategy used for the analysis of firewall policy anomalies. Section VI provides a detailed description of the designed algorithm for assisted anomaly resolution. Section VII describes how the algorithm has been implemented in a prototype tool and validated. Finally, Section VIII draws conclusions and discusses future work.

II. RELATED WORK

In the latest years, policy-based management has again become a relevant network management paradigm. According to the original definition provided in the RFC-3198 (Terminology for Policy-Based Management) [4], ''a network security policy is a set of rules to administer, manage, and control access to the network resource''. Specifically, when a network security policy is a firewall policy (i.e., it expresses information about a firewall behavior), each rule composing it typically has the following structure:

$$< order >< src\_IP >< src\_Port >< dst\_IP >$$
$$< dst\_Port >< protocol >< action >$$

The filtering rule order determines its position relative to other filtering rules. The IP addresses can be the address of a specific network peer (e.g., 140.192.37.120) or a network address (e.g., 140.192.37.*). Ports can be either a specific port number, a range, or a placeholder, indicated by 'any' or '*', to represent all port numbers. A filtering action can be 'allow' or 'deny'.

As recently shown in the survey by Jabal et al. [5], several studies have been proposed in the literature on a specific operation related to network security policies, i.e., policy anomaly analysis. This operation aims to check three main properties of a set of network security policies: correctness, consistency, and minimality.

- Correctness analysis consists in identifying errors that may have been introduced in policy specification so that the policies are free of faults and compliant with their intended goals and system requirements. Ensuring policy correctness also includes validating their syntax and verifying their ability to achieve their goals in all possible contexts and scenarios.
- Consistency analysis consists in identifying conflicts among the policy rules, i.e., identifying all the situations wherein the presence of a rule impacts the behavior of

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

another one. For example, a rule contradicts another one if it is applied to the same packet set, but enforces a different filtering action.
- Minimality analysis consists in ensuring that the policy rule set does not include redundant rules, i.e., detecting sub-optimizations. Redundant rules increase the administrative work required to manage the policy rules and decrease the overall performance, without providing any benefit.

If the analyzed policy rule set lacks any of these three properties, then policy anomaly resolution is the operation that is executed to remove the identified anomalies and reinstate the missing properties.

Policy anomaly analysis does not correspond to policy verification, even if both operations are key concepts of policy-based management. Differently from anomaly analysis, policy verification consists in checking whether a policy rule set is correctly enforced in a system or, in other words, whether the system implementation actually matches the policy semantics [6]–[9]. Therefore, policy verification is commonly an operation that is performed after policy analysis has checked the policy rule set is correct, consistent and minimal, and after it has been enforced on the actual security system. Besides, policy anomaly analysis is also different from policy refinement [10]–[13], which consists in transforming high-level policies into low-level configurations, and usually is a complementary operation in policy-based management. In fact, sometimes performing a full refinement of security policies is not feasible because it would be time-consuming when applied to large networks.

Policy anomaly analysis and resolution have been investigated for the vast domain of access control policies, which may have different characteristics and features. Some of the most relevant studies about it are [14]–[17]. However, the scope of this work lies in the anomaly management of a specific kind of security policy, that is inter-firewall policy. Therefore, as our proposed approach deals with both firewall policy anomaly analysis and resolution, this section analyses research studies related to those two operations, and it compares them with our proposal. In particular, Subsection II-A reports the most significant contributions concerning firewall policy anomaly analysis. Instead, Subsection II-B describes the most effective tools and approaches proposed for resolving firewall anomalies and the relations and differences with our approach.

A. FIREWALL POLICY ANOMALY ANALYSIS

The study by Al-Shaer and Hamed [18] was the first one in literature to address the anomaly analysis problem for firewall policies. This initial study presents a classification scheme for packet filtering rule relations, defining four types of intra-firewall policy rule anomalies, i.e., anomalies that may occur in the same firewall instance. Later, Al-Shaer et al. [19] extend this initial work to detect inter-firewall policy rule anomalies, i.e., anomalies related to policies belonging to different firewall instances in a distributed firewalling architecture. In particular, the inter-firewall policy analysis evaluates rule relations between serially-connected packet filters.

Based on these initial studies of Al-Shaer et al., other researchers proposed alternative models and classification schemes. The most significant ones are [20]–[22]. Specifically, FIREMAN [20], proposed by Yuan et al., uses binary decision diagrams to represent packet filtering policies. Golnabi et al. [21] extend Al-Shaer et al.'s analysis using a data mining technique, named association rule mining. The anomaly detection based on the mining exposes many hidden but not detectable anomalies by analyzing only the firewall policy rules. Such approach allows the identification of two new non-systematic misconfiguration anomalies: blocking existing service and allowing traffic to non-existing service anomalies. The first misconfiguration case blocks legitimate traffic from a trusted network to an ''existing'' service, while the other case of the misconfiguration permits traffic destined for a non-existing service. Then, Bouhoula et al. [22] propose a different approach, based on an inference system to detect intra-firewall policy anomalies. They use the inference system to construct a tree representation of the policy, so that this process stops the construction of a specific branch when no anomaly can be found.

The investigation of the anomaly analysis problem have continued up to now, with another set of more recent studies [23]–[26]. In greater detail, the anomaly analysis modules developed by Togay et al. [23] is based on Prolog, a programming language that adopts the paradigm of logic programming. There, firewall policy rules represent nonconditional information and are transformed into fact clauses, so that anomaly detection predicates can be used to make field-wise analyzes between rule pairs. Lin et al. [24] use an asymmetric double decision tree to detect anomalies. In particular, when detecting an incremental anomaly with respect to a policy, they only need to compare the asymmetric simplified equivalent decision tree to find other anomalies. Both the proposals by Komadina et al. [25] and by Andalina et al. [26] perform anomaly analysis applying data mining and machine learning on data logs. However, machine learning represents a less commonly pursued strategy to address this problem, because the accuracy rate of the results is not 100%, and generally unsupervised methods struggle to detect policy anomalies.

The main limitation of all studies mentioned so far, from Al Shaer et al.'s preliminary work to the most recent ones, is that they exclusively address the firewall anomaly analysis problem. After the strategies proposed in these studies identify an anomaly, they are not removed or solved from the original policy set. Instead, differently from them, our proposed approach also embeds a resolution strategy, which helps human administrators solve the identified anomalies through a reduced number of interactions with them.

Another relevant classes of related work about the topic of firewall anomaly analysis is composed of studies ([27]–[34]) that propose platforms based on Graphic User Interfaces (GUIs), which provide functionalities such as zooming and manual adjustment of the firewall filtering effect areas to help

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

human administrators to identify firewall policy anomalies. Specifically, Tran et al. [27] propose PolicyVis, a 2D graphic interface that allows a thorough visualization of all possible firewall behaviors by considering all rules fields, actions and orders. PolicyVis is flexible enough to let users choose their desired fields as 2D graph coordinates, and eases the anomaly analysis operation introducing the features of compressing, focusing, zooming, and segmentation. Chao [28] creates a systematic visualization for the firewall Rule Anomaly Relation tree (RAR tree) that is created on the basis of the investigated firewall rules. Callouts, arrays and other graphical symbols are used to depict intuitively the presence of anomalies, and a specific GUI view is employed to categorize them. Mansmann et al. [29], [30] introduce a graphical tool named Visual Firewall Editor, which depicts a hierarchical view on firewall rules by using the Sunburst visualization. This tool focuses on two main tasks, i.e., to explore firewall rules and object groups, and to test certain hypotheses about a rule set (e.g., to assess the presence of anomalies). In order to perform them, it uses an interactive tree views to show the firewall rule list, the object group hierarchies and enable navigation in them, with features such as auto-zooming and scaling. Kim et al. [31] propose Firewall Policy Checker to further improve user-friendliness in anomaly analysis. Their tool offers a 3D visualization based on two main graphical views. Their representation is composed of two major views. The first one provides an overall view to the entire firewall policies and the detected anomalies, represented by spheres of different colors. The second one allows to perform risk and illegal service discovery, on the basis of the anomalies identified with the first view. Kim et al. [32] realized the visualization of firewall policies in a 6-dimensional space in 3D format through a tool named FRuVATS. Its visualization model allows to represent all ranges of source and destination IP addresses, thus enabling control on each application service, to identify active and inactive domains, and to detect possible anomalies. Lee et al. [33] propose HSViz, a firewall policy visualization tool that offers multiple view to analyses firewall policies. Among them, the most useful ones are the hierarchy view, which visualizes firewall policy ranges based on destination IP octets at the user's choice, and the anomaly and distributed views, which represent the policy in parallel coordinate charts, easing anomaly detection. Kim et al. [34] introduces a system, named F/Wvis, that allows to visualizes both firewall policies and anomalies on the layout of an user interface with a 3D representation. From that, the user can almost immediately identify how many rules composed the firewall policy and which rule has anomalies with other rules from the same policy. Its GUI also eases the investigation of the firewall policy with a searching tool.

Even if these GUI-based approach can help humans in detecting anomalies, the resolution must still be performed manually. Therefore, those manual human operations are still prone to errors, and they may introduce even a higher number of anomalies. Differently from those GUI-based approaches, our proposal introduces a human-assisted semi-automatic mechanism, where the resolution is guided by human answers, but without requiring the users to actually perform it. This guarantees higher efficiency and avoidance of errors, also thanks to the SMT formulation relying on the correctness-by-construction principle.

B. FIREWALL POLICY ANOMALY RESOLUTION

A second class of studies ([35]–[40]) proposes algorithm for anomaly resolution. However, all of them have shortcomings that our approach aims to overcome.

Liu et al. [35] and Jeffrey et al. [40] propose resolution techniques that simply focus only on detecting and removing redundant rules. On the one hand, Liu et al. [35] categorize redundant rules into two main categories: upward redundant rules and downward redundant rules. Upward redundant rules are rules that are never matched, whereas downward redundant rules are rules that are matched but enforce the same action as rules with lower priority. These sub-optimizations are detected and solved through a model based on a data structure named Firewall Decision Diagram. On the other hand, Jeffrey et al. [40] suggest using a SAT solver for redundancy analysis. Their problem formulation has reduced complexity, allowing for greater performance. However, both strategies cannot detect or resolve policy conflicts. Instead, our approach can also manage this anomaly type, which nonetheless represents the most dangerous one for a firewall configuration.

Cuppens at al. [36] propose an anomaly resolution approach for solving shadowing and redundancy anomalies among intra-firewall policies. A main feature of this study is the definition of a policy rewriting process, which not only solves the aforementioned anomaly types, but can also rewrite a policy in its positive or negative form. The positive form of a policy contains only rules with 'allow' actions, whereas the negative form only rules with 'allow' actions. Then, the authors extend their model to support also inter-firewall policy analysis [41]. Anyhow, their approach only considers some anomaly classes, while our approach addresses all the possible ones. Besides, their resolution strategy is fully automated, and therefore it also takes decisions that instead should be explicitly addressed by security administrators.

Hu et al. [37], [38] propose a rule-based segmentation technique and a grid-based representation to identify policy anomalies and a policy reordering algorithm for anomaly resolution. They also present a proof-of-concept implementation of a visualization-based firewall policy analysis tool called Firewall Anomaly Management Environment. This study has two main limitations to respect to our proposal. First, the reordering algorithm does not guarantee that all policy conflicts and sub-optimizations are actually removed, because there may be cases where no ordering can be found to achieve a complete anomaly resolution. Second, also this resolution strategy is fully automated, and it may not be always adequate.

In summary, to the best of our knowledge, our approach is characterized by manifold features which overcome existing limitations of the state of the art. First, it proposes a joint

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

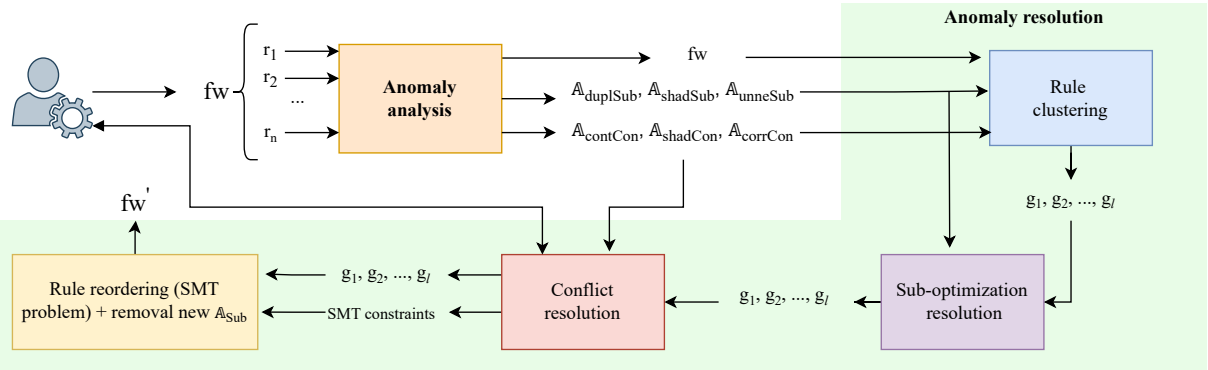

FIGURE 1: Workflow of the proposed approach

algorithm for firewall policy anomaly analysis and resolution, instead of focusing on a single operation. Second, it aims to solve all possible anomaly types, not only sub-optimizations. Third, it involves human intervention to make decisions that may lead to configuration errors if taken autonomously. Fourth, it reduces the number of queries that the administrator must answer to complete the anomaly resolution.

III. OVERVIEW OF THE APPROACH

In this study, we propose a novel methodology for assisting network administrators in performing intra-firewall policy anomaly detection and resolution. The methodology assists administrators in the latter process by guiding them through the shortest series of well-posed questions to ensure that their original security intents are correctly implemented by the firewall. FIGURE 1 illustrates the logical flow of operations in the proposed approach.

A. FIREWALL POLICY ANOMALY DETECTION

At the beginning of the proposed approach, each pair of firewall rules is analyzed so as to detect if they are afflicted by an anomaly. In fact, anomalies are defined as relations among pairs of firewall rules which may reveal either misconfigurations (i.e., mismatches between administrator intents and the actual policy implemented by the firewall based on its current configuration), or sub-optimizations (i.e., non-minimal firewall configurations, typically due to the presence of redundant rules which do not influence the firewall filtering behavior). The definition of anomalies between pairs of rules follows the traditional definitions and allows for fine-grained anomaly detection while also keeping the computational complexity of the detection algorithm within reasonable levels.

The anomalies that are thus detected are then grouped into two classes. On the one hand, *conflict anomalies* identify pairs of rules that apply to common packet subsets (i.e., dependent rules) but are associated with conflicting actions. On the other hand, *sub-optimization anomalies* concern pairs of dependent rules which redundantly process packets in the same way.

The formal definition of the anomaly classes and the formulas used for their identification are detailed in Section V.

B. FIREWALL POLICY ANOMALY RESOLUTION

The task of firewall policy anomaly resolution is composed of four steps: i) firewall rule clustering; ii) sub-optimization resolution; iii) conflict resolution; iv) rule reordering.

1) Firewall rule clustering

After all anomalies have been identified and classified, the firewall rules are grouped in clusters. Two rules are included in the same cluster if they do not have disjoint conditions (i.e., if their conditions match a common subset of packets). Since rules belonging to different clusters are characterized by disjoint conditions, i.e., are matched by different types of traffic, sub-optimizations and conflicts may only arise among pairs of rules in the same cluster. This clustering operation will ease the algorithm defined for firewall policy anomaly resolution, because it can be parallelized and applied to each cluster separately. Grouping rules as clusters is also important, because only considering relations among pairs of firewall rules in most cases does not provide administrators with a clear understanding of the current firewall behavior. Indeed, it may happen that some conflicts regard common regions in the packet space, which are in common to multiple rules and not to a single pair of them.

The formalization of the firewall rule clustering operation is presented in Section VI-A.

2) Sub-optimization resolution

Sub-optimization resolution consists in the automatic resolution of all the sub-optimizations inside each rule cluster. This class of anomalies can be automatically resolved because sub-optimization removal does not change the filtering function implemented by the firewall but only makes its configuration more compact, positively affecting its maintenance and performance.

The algorithm used for the resolution of the sub-optimizations is formalized in Section VI-B.

3) Conflict resolution

Conflict resolution consists in a human-assisted resolution of the conflicts. This task is not fully automated, because

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

approaches that automatically resolve anomalies following predefined resolution strategies do not allow fine-grained control of the resulting configurations, increasing the risk of ignoring existing errors or even introducing additional ones. As our approach designs also this task in a cluster-based fashion, only clusters containing at least one pair of conflicting rules are examined by the resolution process.

In particular, the approach envisions a sequence of well-posed queries to the administrators with the aim of correcting actual misconfigurations. Two types of queries can be formulated to the administrator: queries of type A and queries of type B.

On the one hand, queries of type A ask the administrator if all the packets matching the condition of a specific rule should be subject to the action of that same rule. If the administrator answers positively to such kind of question, all the conflicts where that rule is involved can be removed, because that is the "winning rule" that must have higher priority than the other involved rules. In this case, some restrictions about the rule relative order are derived, and they will be useful to reorder the rules remaining in the cluster. Besides, under certain conditions, the rules which were deemed not to be the winning ones for a conflict may also be removed.

On the other hand, queries of type B ask the administrator to decide which action between the actions of two rules should be applied to the packets that match both their condition. Queries of type B are the ones that are also used in some techniques for manual conflict resolution. However, a resolution process consisting only or mostly of queries of type B would be unoptimized, because a query would be asked for each specific conflict. Therefore, our approach reduces the number of human interventions, by following a strategy that allows prioritizing queries of type A, whose answers lead to the simultaneous resolution of a higher number of conflicts. The selection of the rules subject to those queries is based on a cost function, related to the number of conflicts that each rule has with the other rules in the previously computed clusters.

In summary, the answers provided by the administrator may be useful both to remove some rules that are not needed anymore after the resolution of the related conflicts, and to define some restrictions about the rule relative order.

The algorithm used for the interaction with the administrator in the context of conflict resolution is formalized in Section VI-C1.

4) Rule reordering

After each cluster has been depured of rules that were deemed useless after the resolution of their conflicts, the remaining rules must be reordered, so as to satisfy the rule order restrictions deriving from the answers themselves.

In our approach, the rule reordering problem has been formulated as a Satisfiability Modulo Theories (SMT) problem. The SMT problem is an enhanced version of the most traditional satisfiability checking technique, the Satisfiability (SAT) problem. A SAT problem is used to check the satisfiability of formulas where only Boolean variables appear. However, for the rule ordering representation, we also need variables and functions of other types (e.g., integer variables to represent the ordering positions). From this point of view, an SMT problem allows checking the satisfiability of formulas in a decidable combination of first-order theories, including arithmetic and uninterpreted functions theories.

An SMT problem can be automatically solved by state-of-the-art efficient solvers, already available in literature [42]–[44]. The solver assigns a firewall rule index to each integer variable, representing a position in the final anomaly-free cluster, and from these assignments the output ordering is derived. The solution of the solver is also guaranteed to be "correct by construction", as long as the formal models representing the problem components (e.g., firewall rules and their relationship, in our case) capture all the information that may impact the solution correctness. In this way, a human user also has a formal assurance about the result of the automatic sub-processes composing the proposed approach.

The formalization of the SMT problem used to reorder the rules inside each cluster is presented in VI-C2.

Finally, after the rules inside each cluster have been reordered, the algorithm for sub-optimization resolution should be reapplied to ensure that no new sub-optimization was created within the reordering operation. Then, the clusters should be combined to form a single firewall policy. In principle, as there were no anomalies between two rules belonging to different clusters, the relative position of the clusters may be arbitrarily decided, as it is indifferent to the solution correctness. However, the human administrator also has the faculty to order the clusters in a strategic way, e.g., associating higher priority to the cluster whose rules match network packets more frequently, thus optimizing the filtering process of the firewall.

IV. FIREWALL POLICY MODEL

This section recalls the basics of firewall operation and provides a model, derived from the well-established literature on the subject, for describing their configuration. Relations among pairs of firewall rules are also defined and classified.

A. FIREWALLS AND PACKETS

Firewalls filter traversing packets based on the values assumed by a subset of the header fields of the latters (named *firewall filtering fields*). A field is modeled as a range, i.e., a set of consecutive non-negative integers, $P_n = [a_n, b_n]$[1], including all values possibly assumed by the field. As an example, IP-layer filtering is typically performed based on the values of 5 IP header fields: source and destination IP address, source and destination port number and transport protocol, which, for IPv4, are modeled, respectively, as ranges $P_{1,2} = [0, 2^{32} - 1]$, $P_{3,4} = [0, 2^{16} - 1]$, and $P_5 = [0, 2^8 - 1]$.

For a firewall defined over filtering fields $P_1, P_2, \ldots, P_N$, packets are N-tuples $p = (p_1, p_2, \ldots, p_N)$, where $p_n \in P_n$,

[1]In the following, capital letters and the traditional interval notation are used to indicate ranges, e.g., $I = [a, b]$ means that $a, b \in \mathbb{N}$ are respectively the minimum and maximum-valued non-negative integers in range I.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

$n \in [1,N]$, represent the values assumed by the firewall filtering fields in the packet header. If we consider filtering fields to be represented on N orthogonal axes, the set of all possible packets[2], $\mathcal{P} = P_1 \times P_2 \times \ldots \times P_N$, is a N-orthotope (i.e., the generalization of a rectangle in N dimensions) in $\mathbb{N}^N$, while each packet is a single point in $\mathbb{N}^N$.

A *firewall* defined over filtering fields $P_1, P_2, \ldots, P_N$ is then a list of rules $fw = [r_1, r_2, \ldots, r_{|fw|}]$, where the number of rules, $|fw|$, is the *firewall cardinality*. A *rule* r_i, $i \in [1, |fw|]$, is defined as $r_i = \langle c^i, a^i \rangle$, where c^i is a condition and a^i is an action. For a firewall defined over filtering fields $P_1, P_2, \ldots, P_N$ a *condition* is a N-tuple of ranges $c^i = (C_1^i, C_2^i, \ldots, C_N^i)$, where $C_n^i \subseteq P_n$, $n \in [1,N]$. A condition implicitly defines a logical predicate over packets $p \in \mathcal{P}$, i.e., $\phi_{c^i}(p) = p_1 \in C_1^i \wedge p_2 \in C_2^i \wedge \ldots \wedge p_N \in C_N^i$ and identifies the set of packets, $\mathcal{S}_{c^i} = C_1^i \times C_2^i \times \ldots \times C_N^i \subseteq \mathcal{P}$ (also a N-orthotope in $\mathbb{N}^N$), satisfying the condition itself, i.e., for which predicate $\phi_{c^i}(p)$ holds true. Moreover, we assume possible firewall actions to be such that $a^i \in \{allow, deny\}$.

Any time a packet $p = (p_1, p_2, \ldots, p_N)$ reaches a firewall $fw = [r_1, r_2, \ldots, r_{|fw|}]$ defined as above, the packet is checked against the firewall rules according to their position in the list. As soon as the first rule whose condition is satisfied by the packet (first *matched rule*) is found, the rule action is applied and the packet is either forwarded (if the action is *allow*) or discarded (if, conversely, is *deny*). The *priority* of a rule is thus given by the rule position within the firewall list as, whenever a packet satisfies the conditions of multiple rules within a firewall, the applied action is the one of the rule firstly appearing in the list. We assume the priority of a rule r_i, $i \in [1, |fw|]$, within a firewall $fw = [r_1, r_2, \ldots, r_{|fw|}]$, to be returned by function $\pi(r_i) = i$, where the lower $\pi(r_i)$ the higher the rule priority.

According to the above description, a firewall can be seen as a function (*firewall filtering function*) associating to any packet $p \in \mathcal{P}$ one among the possible firewall actions. To make sure that firewall filtering functions are defined over the whole packet set $\mathcal{P}$, the last rule in any firewall (*default rule*), is supposed to be in the form $r_{|fw|} = \langle (P_1, P_2, \ldots, P_N), a^{|fw|} \rangle$, i.e., to be matched by any packet (*completeness hypothesis).*

In the following, the traditional dotted decimal notation, with wildcards $*$, is also used to define ranges referring to IP addresses (e.g., $10.10.10.*$ stands for $10.10.10.0/8$). Moreover, names will be used for protocols (e.g, *TCP* is a possible protocol type name).

B. RELATIONS AMONG FIREWALL RULE CONDITIONS

Here, we explore all possible relations among pairs of rule conditions which, in turn, can straightforwardly be derived from possible relations among pairs of condition components (i.e., ranges).

Given any two ranges $C_n = [a_n, b_n]$ and $C_m = [a_m, b_m]$, they can be seen as two sets of values and, thus, they are in one of the following, mutually exclusive, relations:

- *equivalence* ($=$): two ranges are equivalent iff they they are the same set, i.e.,

$$C_n = C_m \Leftrightarrow C_m = C_n \Leftrightarrow a_n = a_m \wedge b_n = b_m \quad (1)$$

- *disjointness* ($\perp$): two ranges are disjoint iff they are disjoint sets, i.e.,

$$C_n \perp C_m \Leftrightarrow C_m \perp C_n \Leftrightarrow b_n < a_m \vee a_n > b_m \quad (2)$$

- *dominance* ($\prec, \succ$): one range dominates (or is dominated by) the other iff it is a proper superset (or subset) of the other, i.e.,

$$C_n \succ C_m \Leftrightarrow C_m \prec C_n \Leftrightarrow (a_n < a_m \wedge b_n \geq b_m) \vee (a_n = a_m \wedge b_n > b_m) \quad (3)$$

$$C_n \prec C_m \Leftrightarrow C_m \succ C_n \Leftrightarrow (a_n > a_m \wedge b_n \leq b_m) \vee (a_n = a_m \wedge b_n < b_m) \quad (4)$$

- *correlation* ($\sim$): two ranges are correlated iff their intersection and their mutual differences are all non-empty, i.e., the ranges are not equivalent, nor disjoint, and neither of them dominates the other:

$$C_n \sim C_m \Leftrightarrow C_m \sim C_n \Leftrightarrow (a_n > a_m \wedge a_n \leq b_m \wedge b_n \geq b_m) \vee (b_n < b_m \wedge b_n \geq a_m \wedge a_n \leq a_m) \quad (5)$$

Analogously to ranges, given any two rule conditions $c^i = (C_1^i, C_2^i, \ldots, C_N^i)$ and $c^j = (C_1^j, C_2^j, \ldots, C_N^j)$, they can be univocally identified with the corresponding packet sets $\mathcal{S}_{c^i}$ and $\mathcal{S}_{c^j}$, and are, thus, in one of the following, mutually exclusive, relations:

- *equivalence* ($=$): two conditions are equivalent iff the corresponding packet sets are the same set, i.e.,

$$c^i = c^j \Leftrightarrow c^j = c^i \Leftrightarrow C_n^i = C_n^j \; \forall n \in [1,N] \quad (6)$$

- *disjointness* ($\perp$): two conditions are disjoint iff the corresponding packet sets are disjoint, i.e.,

$$c^i \perp c^j \Leftrightarrow c^j \perp c^i \Leftrightarrow \exists\, \bar{n} \in [1,N] \mid C_{\bar{n}}^i \perp C_{\bar{n}}^j \quad (7)$$

- *dominance* ($\prec, \succ$): one condition dominates (or is dominated by) the other iff the corresponding packet set is a proper superset (or subset) of the packet set of the other, i.e.,

$$c^i \succ c^j \Leftrightarrow c^j \prec c^i \Leftrightarrow C_n^i \succeq C_n^j \; \forall n \in [1,N] \wedge \exists\, \bar{n} \in [1,N] \mid C_{\bar{n}}^i \succ C_{\bar{n}}^j \quad (8)$$

$$c^i \prec c^j \Leftrightarrow c^j \succ c^i \Leftrightarrow C_n^i \preceq C_n^j \; \forall n \in [1,N] \wedge \exists\, \bar{n} \in [1,N] \mid C_{\bar{n}}^i \prec C_{\bar{n}}^j \quad (9)$$

- *correlation* ($\sim$): two conditions are correlated iff the intersection and the mutual differences between the corresponding packet sets are all non-empty, i.e., the conditions are not equivalent, nor disjoint, and neither of them dominates the other:

$$c^i \sim c^j \Leftrightarrow c^j \sim c^i \Leftrightarrow C_n^i \not\perp C_n^j \; \forall n \in [1,N] \wedge \exists\, \bar{n} \in [1,N] \mid C_{\bar{n}}^i \sim C_{\tilde{n}}^j \quad (10)$$

[2]In the following, Italic capital letters are used to indicate packet sets.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

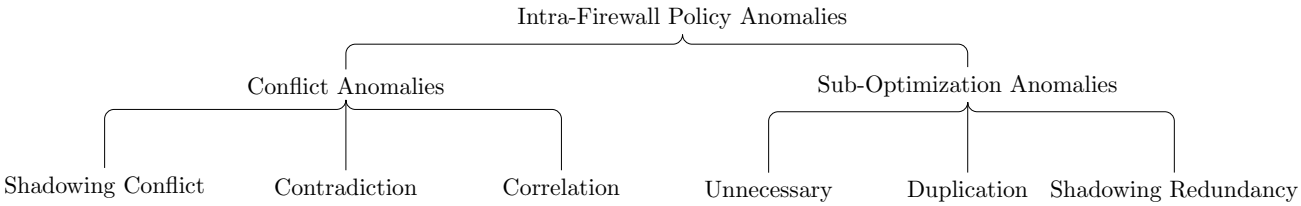

FIGURE 2: Taxonomy of Intra-Firewall Policy Anomalies

Note that, for both ranges and rule conditions, equivalence, correlation and disjointness relations are symmetric, while dominance is asymmetric and transitive. Note that symbols $\succeq, \preceq$ respectively indicate that a range/condition *dominates/is dominated or equivalent* to another, and symbols $\neq$, $\not\perp$, $\not\sim$ state that two ranges/conditions are, respectively, *not equivalent*, *not disjoint* or *not correlated*.

V. FIREWALL POLICY ANOMALY ANALYSIS

As mentioned in the previous sections, this work focuses on firewall anomalies afflicting pairs of rules within the same firewall list (*intra-firewall policy anomalies*).

Firewall configuration anomalies only arise among pairs of rules ($r_i = \langle c^i, a^i \rangle$ and $r_j = \langle c^j, a^j \rangle$) with non-disjoint conditions ($c^i \not\perp c^j$), i.e., which are matched by a common subset of packets.

As shown in FIGURE 2, three are two classes of anomalies: *conflict anomalies*, which involve pairs of rules specifying different actions ($a^i \neq a^j$), and *sub-optimization anomalies*, which, conversely, involve pairs of rules characterized by the same action ($a^i = a^j$). The *conflict anomalies* are divided into three categories: *Contradiction*, *Shadowing Conflict*, and *Correlation* anomalies. Instead, the *sub-optimizations* anomalies are divided into three different categories: *Duplication*, *Shadowing Redundancy*, and *Unnecessary* anomalies.

For conflict anomalies, only administrators can clarify the intended fate for packets satisfying both rule conditions. Thus, resolving conflict anomalies requires interacting with the administrator, i.e., it cannot be carried out automatically and possibly leads to modifying the firewall filtering function. Conversely, sub-optimization anomalies highlight the presence of redundant rules, unnecessarily stating the same action for a specific kind of traffic. The resolution of sub-optimization anomalies is meant to increase configuration compactness by removing redundancy (which is beneficial to firewall maintenance and filtering performance) and, thus, does not impact the firewall filtering function and can be automatically conducted.

In the following subsections, we will describe the characteristics of each type of anomaly and the formulas to detect them.

1) Conflict anomalies

- *Contradiction anomaly* $\mathcal{A}_{contCon}(r_i, r_j)$:
  A contradiction anomaly exists between two rules r_i and r_j iff

$$c^i = c^j \wedge a^i \neq a^j \tag{11}$$

  The two rules are matched by exactly the same set of packets ($c^i = c^j$), but they indicate different actions ($a^i \neq a^j$). This means that the lower priority rule is never actually applied. Note that the contradiction anomaly is symmetric: $\mathcal{A}_{contCon}(r_i, r_j) \Leftrightarrow \mathcal{A}_{contCon}(r_j, r_i)$.
  The resolution of a contradiction anomaly consists in deleting one of the two involved rules.
  The set of all contradiction anomalies $\mathbb{A}_{contCon}$ in *fw* is defined as: $\{\{r_i, r_j\}, i, j \in [1, |fw|], \mid \mathcal{A}_{contCon}(r_i, r_j)\}$.
- *Shadowing conflict anomaly* $\mathcal{A}_{shadCon}(r_i, r_j)$:
  A shadowing conflict anomaly exists between two rules r_i and r_j iff

$$\pi(r_i) < \pi(r_j) \wedge c^i \succ c^j \wedge a^i \neq a^j \tag{12}$$

  The rule with the lower priority ($\pi(r_i) < \pi(r_j)$) is matched by a proper subset of the packets matching the higher priority one ($c^i \succ c^j$), and the rules have different actions ($a^i \neq a^j$). This means that the lower priority rule is shadowed by the higher priority one.
  The resolution of a shadowing conflict anomaly consists in either the lower priority rules deletion or the rules relative order change.
  The set of all shadowing conflict anomalies $\mathbb{A}_{shadCon}$ in *fw* is defined as: $\{\{r_i, r_j\}, i, j \in [1, |fw|], \mid \mathcal{A}_{shadCon}(r_i, r_j) \vee \mathcal{A}_{shadCon}(r_j, r_i)\}$.
- *Correlation anomaly* $\mathcal{A}_{corrCon}(r_i, r_j)$:
  A correlation anomaly exists between two rules r_i and r_j iff

$$c^i \sim c^j \wedge a^i \neq a^j \tag{13}$$

  The sets of packets matching the two rules have both non-null intersections ($c^i \sim c^j$) and differences, and the rules have different actions. Note that the correlation anomaly is symmetric: $\mathcal{A}_{corrCon}(r_i, r_j) \Leftrightarrow \mathcal{A}_{corrCon}(r_j, r_i)$.
  The resolution of a correlation anomaly consists in either the deletion of the lower priority rule or the change of the relative rule order.
  The set of all correlation anomalies $\mathbb{A}_{corrCon}$ in *fw* is defined as: $\{\{r_i, r_j\}, i, j \in [1, |fw|], \mid \mathcal{A}_{corrCon}(r_i, r_j)\}$.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

2) Sub-optimization anomalies

- *Duplication anomaly* $\mathcal{A}_{duplSub}(r_i, r_j)$:
  A duplication anomaly exists between two rules r_i and r_j iff:

$$c^i = c^j \wedge a^i = a^j \wedge \nexists r_z | \pi(r_y) < \pi(r_z) < \pi(r_x) \wedge c^z \not\perp c^x \wedge a^z \neq a^x \quad (14)$$

  The two rules are exactly the same ($c^i = c^j \wedge a^i = a^j$), if there not exists another rule in the middle ($\nexists r_z | \pi(r_y) < \pi(r_z) < \pi(r_x)$), with a different action ($a^z \neq a^x$), that matches a common set of packets of the duplicated rules ($c^z \not\perp c^x$). Note that the duplication anomaly is symmetric: $\mathcal{A}_{duplSub}(r_i, r_j) \Leftrightarrow \mathcal{A}_{duplSub}(r_j, r_i)$.
  Duplication anomalies are automatically resolved by deleting the lower priority rule in the pair, which is never actually applied.
  The set of all duplication anomalies $\mathbb{A}_{duplSub}$ in fw is defined as: $\{\{r_i, r_j\}, i,j \in [1, |fw|], \mid \mathcal{A}_{duplSub}(r_i, r_j)\}$.
- *Shadowing redundancy anomaly* $\mathcal{A}_{shadSub}(r_i, r_j)$:
  A shadowing redundancy anomaly exists between two rules r_i and r_j iff:

$$\pi(r_i) < \pi(r_j) \wedge c^i \succ c^j \wedge a^i = a^j \wedge (\nexists r_z | c^z \not\perp c^j \wedge a^z \neq a^j) \quad (15)$$

  The rule with lower priority matches a subset of the packet set matched by the higher priority rule ($\pi(r_i) < \pi(r_j) \wedge c^i \succ c^j$), and there not exists another rule with a different action ($a^z \neq a^x$) that matches a common set of packets of the shadowed rule ($c^z \not\perp c^j$). Under these conditions, also this rule relationship is considered a sub-optimization (i.e., it can be dealt with without information loss).
  Shadowing redundancy anomalies are automatically resolved by deleting the lower priority rule, which is never actually applied.
  The set of all shadowing redundancy anomalies $\mathbb{A}_{shadSub}$ in fw is defined as $\{\{r_i, r_j\}, i,j \in [1, |fw|], \mid \mathcal{A}_{shadSub}(r_i, r_j) \vee \mathcal{A}_{shadSub}(r_j, r_i)\}$.
- *Unnecessary anomaly* ($\mathcal{A}_{unneSub}(r_i, r_j)$):
  An unnecessary anomaly exists between two rules r_i and r_j iff:

$$\pi(r_i) < \pi(r_j) \wedge c^i \prec c^j \wedge a^i = a^j \wedge (\nexists r_z | c^z \not\perp c^i \wedge a^z \neq a^i) \quad (16)$$

  The rule with higher priority matches a subset of the packet set matched by the higher priority rule ($\pi(r_i) < \pi(r_j) \wedge c^i \prec c^j$), and it does not exist another rule with a different action ($a^z \neq a^i$) that matches a common set of packets of the smaller rule ($c^z \not\perp c^i$). Under these conditions, also this rule relationship is considered a sub-optimization (i.e., it can be dealt with without information loss).
  Unnecessary anomalies are automatically resolved by deleting the higher priority rule, whose presence does not impact the firewall filtering function.
  The set of all unnecessary anomalies $\mathbb{A}_{unneSub}$ in fw is defined as $\{(r_i, r_j), i,j \in [1, |fw|], \mid \mathcal{A}_{unneSub}(r_i, r_j) \vee \mathcal{A}_{unneSub}(r_j, r_i)\}$.

VI. FIREWALL ANOMALY RESOLUTION

In this section, all the steps of the proposed methodology for firewall anomaly resolution are described. Specifically, the aim of the approach is to assist network administrators in resolving policy anomalies. In doing so, it executes some steps automatically, while for others, it requires interaction with the administrator, i.e., their answer to a series of well-posed queries.

In accordance with the diagram in FIGURE 1, the input of the resolution algorithm is a firewall configuration $fw = (r_1, r_2, \ldots, r_{|fw|})$, defined as in Section IV, and the sets of all configuration anomalies IV, i.e., $\mathbb{A}_{contCon}$, $\mathbb{A}_{shadCon}$, $\mathbb{A}_{corrCon}$, $\mathbb{A}_{duplSub}$, $\mathbb{A}_{shadSub}$ and $\mathbb{A}_{unneSub}$, already identified with the strategy discussed in Section V.

A. RULE CLUSTER COMPUTATION

As a first step, a partition $\mathbb{G} = \{g_1, g_2, \ldots, g_L\} \subseteq 2^{\{r_1, r_2, \ldots, r_{|fw|}\}}$ of the set of the firewall rules is defined.

Specifically, elements $g_\ell \in \mathbb{G}$, $\ell \in [1, L]$, are all and only *clusters* of rules within fw, i.e., non-empty subsets of the set of firewall rules, which satisfy both the following properties:

$$\begin{aligned} &(i)\ \forall\, i,j \in [1, |fw|], i \neq j, g_\ell \in \mathbb{G},\ r_i = \langle c^i, a^i \rangle, \\ &\quad r_j = \langle c^j, a^j \rangle \in g_\ell \Rightarrow c^i \not\perp c^j \\ &(ii)\ \forall\, i,j \in [1, |fw|], i \neq j, g_\ell, g_h \in \mathbb{G},\ r_i = \langle c^i, a^i \rangle \in g_\ell, \\ &\quad r_j = \langle c^j, a^j \rangle \in g_h \Rightarrow c^i \perp c^j \end{aligned} \quad (17)$$

Properties (17) state that clusters indicate all and only groups of rules belonging to fw with non-disjoint conditions. Clusters $g_\ell \in \mathbb{G}$, $\ell \in [1, L]$ can be separately examined with respect to the detection and resolution of configuration anomalies. Indeed, since rules belonging to different clusters are characterized by disjoint conditions, i.e., are matched by different types of traffic, conflicts may only arise among pairs of rules in the same cluster. Moreover, the resolution of a conflict involving rules in a cluster does not affect rules in the others. For these reasons, the second step of our methodology consists in computing, for each $g_\ell \in \mathbb{G}$, $\ell \in [1, L]$, the subsets $\mathbb{A}_{contCon,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$, $\mathbb{A}_{duplSub,\ell}$, $\mathbb{A}_{shadSub,\ell}$ and $\mathbb{A}_{unneSub,\ell}$ grouping anomalies of each type only involving rules in cluster g_ℓ.

The advantages of creating this partition are manifold. On the one hand, human administrators can more intuititely understand how each rule is relationed with the other ones, and therefore they may have a clearer understanding of the firewall behavior. On the other hand, as there are no anomalies between two rules belonging to different clusters, the remainder of the resolution algorithm can be parallelized, thus improving the overall efficiency of the process.

Once this step is concluded, the resolution of anomalies in each cluster is separately conducted with a positive impact

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

Algorithm 1 for sub-optimization resolution

Input: the cluster g_ℓ, the sub-optimization sets $\mathbb{A}_{duplSub,\ell}$, $\mathbb{A}_{shadSub,\ell}$, $\mathbb{A}_{unneSub,\ell}$, the conflict sets $\mathbb{A}_{contCon,,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$
Output: the modified cluster g_ℓ, the modified sets $\mathbb{A}_{contCon,,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$

$R_{toRemove} \leftarrow \emptyset$
for each $\{r_i, r_j\} \in (\mathbb{A}_{duplSub,\ell} \cup \mathbb{A}_{shadSub,\ell})$ **do**
  if $\pi(r_i) > \pi(r_j)$ **then**
    $R_{toRemove} \leftarrow R_{toRemove} \cup \{r_i\}$
    $g_\ell \leftarrow g_\ell \backslash \{r_i\}$
  else
    $R_{toRemove} \leftarrow R_{toRemove} \cup \{r_j\}$
    $g_\ell \leftarrow g_\ell \backslash \{r_j\}$
for each $\{r_i, r_j\} \in \mathbb{A}_{unneSub,\ell}$ **do**
  if $\pi(r_i) < \pi(r_j)$ **then**
    $R_{toRemove} \leftarrow R_{toRemove} \cup \{r_i\}$
    $g_\ell \leftarrow g_\ell \backslash \{r_i\}$
  else
    $R_{toRemove} \leftarrow R_{toRemove} \cup \{r_j\}$
    $g_\ell \leftarrow g_\ell \backslash \{r_j\}$
for each $r_k \in R_{toRemove}$ **do**
  for each $\mathbb{A}_{X,\ell} \in \{\mathbb{A}_{contCon,\ell}, \mathbb{A}_{shadCon,\ell}, \mathbb{A}_{corrCon,\ell}\}$ **do**
    for each $(r_m, r_n) \in \mathbb{A}_X$ **do**
      if $r_k = r_m \vee r_k = r_n$ **then**
        $\mathbb{A}_{X,\ell} \leftarrow \mathbb{A}_{X,\ell} \backslash \{r_m, r_n\}$
return q_ℓ, $\mathbb{A}_{contCon,,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$

on computational complexity, so from now on we assume to consider a single reference cluster $g_\ell \in \mathbb{G}$, $\ell \in [1, L]$.

B. SUB-OPTIMIZATION ANOMALY RESOLUTION

The second step of the proposed resolution methodology consists in the automatic resolution of all detected sub-optimization anomalies according to the strategy formalized in Algorithm 1. In words, this algorithm fulfills the following resolution policies:

- $\forall \{r_i, r_j\} \in (\mathbb{A}_{duplSub,\ell} \cup \mathbb{A}_{shadSub,\ell})$:
  1) delete the lower priority rule in the pair from the firewall configuration;
- $\forall \{r_i, r_j\} \in \mathbb{A}_{unneSub,\ell}$:
  1) delete the higher priority rule in the pair from the firewall configuration;
- for any deleted rule, delete all conflict anomalies in $(\mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell} \cup \mathbb{A}_{corrCon,\ell})$ involving the rule.

C. CONFLICT ANOMALY RESOLUTION

Once all firewall sub-optimization anomalies have been resolved and conflict anomalies have been updated accordingly, the resolution of firewall conflict anomalies needs to take place. This task is composed of two main operations: 1) a series of well-posed queries to the network administrator, with the objective of establishing constraints about rule priority or rule deletion for the resolution of all anomalies; 2) the resolution of an SMT problem to establish the actual priority of the remaining rule, so that the output rule set is conflict-free.

1) Queries for conflict resolution

Our approach envisions two types of queries that can be posed to the network administrators:

1) Queries of "type A" are in the form:

$$queryTypeA = (r_i)? \tag{18}$$

where the semantics of the query is as follows

$$\begin{gathered}\text{Shall set of packets } \{p \in \mathcal{P} \mid p \text{ satisfies } c^i\} \\ \text{be subject to action } a^i?\end{gathered} \tag{19}$$

In other words, the administrator is asked if the action a^i of rule r_i should actually be applied to all the packets matching the condition c^i of r_i. The possible answers the administrator can provide to the query are "yes" or "no". If the administrator answers "yes", all the conflicts where r_i is involved can be solved, as it has been established that a^i must be applied to all packets matching c^i. In particular, in the final anomaly-free policy, r_i must have higher priority than the rules it had conflicts with. Those rules may also be removed, if they have become useless after conflict resolution. Instead, if the administrator answers "no", then some packets matching c^i must not be subject to the action a^i. This means that there is another rule, in an anomalous relationship with r_i, that must have higher priority than r_i.

2) Queries of "type B" are in the form:

$$queryTypeB = (r_i, r_j)? \tag{20}$$

where the semantics of the query is as follows

$$\begin{gathered}\text{Shall set of packets } \{p \in \mathcal{P} \mid p \text{ satisfies both } c^i \text{and } c^j\} \\ \text{be subject to action } a^i \text{ or } a^j?\end{gathered} \tag{21}$$

In other words, the administrator is asked which action, between a^i and a^j, should be applied to the packet space which originated a conflict among r_i and r_j. The possible answers the administrator can provide to the query are a^i or a^j. If the administrator answers a^i, then in the final anomaly-free policy r_i must have higher priority than r_j (or vice-versa, if the administrator answers a^j).

Queries of type A may lead to solving multiple conflicts with a single answer, thus reducing the total number of queries the administrator must answer and optimizing the process as the human operator is assisted in the conflict resolution. Queries of type B are asked only when it is not possible to ask queries of type A anymore. This happens when all queries of type A whose answers may lead to a simoultaneous resolution of multiple conflicts have already been asked (independently of the answers), and all remaining conflicts only involve pairs of rules. However, it is expected that a large number of conflicts are already solved before the process starts to ask queries of type B to the administrator. This feature represents an optimization with respect to traditional approaches for firewall policy conflict resolution, where only variants of

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

Algorithm 2 for conflict resolution

Input: the cluster g_ℓ, the conflict sets $\mathbb{A}_{contCon,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$
Output: the modified cluster g_ℓ, the set of constraints K

1: **for each** r_k in g_ℓ **do**
2: $toAsk[r_k] \leftarrow$ true
3: *endFirstPart* $\leftarrow$ *false*
4: **while** *endFirstPart* = *false* **do**
5: *conflictNo* $\leftarrow$ *compConfNo*$(g_\ell, \mathbb{A}_{contCon,\ell}, \mathbb{A}_{shadCon,\ell},$
6: $\mathbb{A}_{corrCon,\ell})$
7: $g_\ell \leftarrow orderByDescConflictNo(g_\ell, conflictNo)$
8: **for each** r_i in g_ℓ **do**
9: **if** $conflictNo[r_i] \leq 1$ **then**
10: *endFirstPart* $\leftarrow$ *true*
11: **break**
12: **if** $toAsk[r_i] =$ *false* **then**
13: **continue**
14: $response \leftarrow queryTypeA(r_i)$
15: $toAsk[r_i] \leftarrow$ false
16: **if** $response =$ "*yes*" **then**
17: **for each** $\{r_i, r_j\} \in \mathbb{A}_{contCon,\ell} \vee \{r_i, r_j\} \in \mathbb{A}_{shadCon,\ell} |$
18: $c^i \succeq c^j$ **do**
19: $toAsk[r_j] \leftarrow$ false
20: $g_\ell \overleftarrow{} g_\ell \backslash \{r_j\}$
21: $\mathbb{A}_{contCon,\ell} \leftarrow \mathbb{A}_{contCon,\ell} \backslash \{r_i, r_j\}$
22: $\mathbb{A}_{shadCon,\ell} \leftarrow \mathbb{A}_{shadCon,\ell} \backslash \{r_i, r_j\}$
23: **for each** $\mathbb{A}_{X,\ell} \in \{\mathbb{A}_{contCon,\ell}, \mathbb{A}_{shadCon,\ell},$
24: $\mathbb{A}_{corrCon,\ell}\}$ **do**
25: **for each** $(r_m, r_n) \in \mathbb{A}_{X,\ell}$ **do**
26: **if** $r_j = r_m \vee r_j = r_n$ **then**
27: $\mathbb{A}_{X,\ell} \leftarrow \mathbb{A}_{X,\ell} \backslash \{r_m, r_n\}$
28: **for each** $\{r_i, r_j\} \in \mathbb{A}_{corrCon,\ell} \vee \{r_i, r_j\} \in \mathbb{A}_{shadCon,\ell} |$
29: $c^j \succeq c^i$ **do**
30: $toAsk[r_j] \leftarrow$ false
31: $\mathbb{A}_{corrCon,\ell} \leftarrow \mathbb{A}_{corrCon,\ell} \backslash (r_i, r_j)$
32: $\mathbb{A}_{shadCon,\ell} \leftarrow \mathbb{A}_{shadCon,\ell} \backslash (r_i, r_j)$
33: $k_{ij} \leftarrow$ "$greater(i, j) = true$"
34: $K \leftarrow K \cup \{k_{ij}\}$
35: **for each** $(r_i, r_j) \in \mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell} \cup \mathbb{A}_{corrCon,\ell}$ **do**
36: $response \leftarrow queryTypeB(r_i, r_j)$
37: **if** $response = a^i$ **then**
38: $k_{ij} \leftarrow$ "$greater(i, j) = true$"
39: **else**
40: $k_{ij} \leftarrow$ "$greater(j, i) = true$"
41: $K \leftarrow K \cup \{k_{ij}\}$
42: **return** q_ℓ, K

queries of type B are asked to the network administrator, who therefore must solve the conflicts one by one.

The algorithm works on the rule of a single cluster g_l, after sub-optimizations have been removed as previously described. Therefore, the algorithm can be executed in parallel on each cluster, thus improving the efficiency of the conflict resolution process. The conflict resolution methodology proposed in this paper is detailed in Algorithm 2.

In the beginning, the algorithm initializes some variables and associative arrays (lines 1-3). In greater detail, for each rule r_k of the analyzed cluster, the corresponding value of the associative array $toAsk[r_k]$ is set to true: this means that conflicts related to that rule still require queries to be posed to the administrator. Instead, the Boolean variable *endFirstPart* is set to false, and it will be used in termination conditions. After this preliminary phase, the core of the algorithm is composed of two main blocks: in the former (lines 4-34) the administrator is asked queries of type A, in the latter (lines 35-41) he is asked queries of type B to solve the remaining conflicts.

The first block (lines 4-34) has an iterative structure, as it is repeatedly executed until *endFirstPart* is not modified to true. In each iteration, the number of conflicts that are afflicting the rules that are still present in the cluster g_ℓ is computed with the *compConfNo* function, which is detailed in Algorithm 3. This function simply checks if any rule r_k appears in the rule pairs composing the three sets $\mathbb{A}_{contCon,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$, and sets the value of the associative array $conflictNo[r_k]$ to the total number of current conflicts for that rule. Then the algorithm reorders the rules in g_ℓ in descending order of conflict number. The quick sort algorithm, well-known in literature, is employed for this reordering operation. At that point, a single rule r_i of the reordered cluster is analyzed at a time. If this rule has only one conflict or it does not have any conflict (i.e., $conflictNo[r_i] \leq 1$), the first part of the algorithm must conclude (i.e., the Boolean variable *endFirstPart* is set to true), because it is not possible anymore to solve multiple conflicts at the same time asking single queries of type A. Otherwise, if asking a question for that rule is still useful (i.e., $toAsk[r_i]$ = true), a query of type A is posed to the administrator, asking if the action a^i must be applied to all packets matching the condition c^i. Independently of the answer, $toAsk[r_i]$ is set to false, because the query has been asked for that rule. Then, a positive answer means that the rule r_i wins all the conflicts where it is involved, and those conflicts can be automatically removed from their respective sets $\mathbb{A}_{contCon,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$. Their simultaneous removal with a single query represents a strong point of the proposed methodology, as it avoids specific queries for each conflict. In addition to removing them from those sets, other operations must be performed to ensure the conflict removal in the final firewall policy that will be produced at the end of the whole resolution algorithm. On the one hand, if there was a contradiction between r_i and another rule r_j, or if there was a shadowing conflict where r_i shadowed r_j, then r_j can be directly deleted from the cluster g_ℓ, and all the conflicts where it was involved removed from the sets $\mathbb{A}_{contCon,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$. On the other hand, if there was a correlation between r_i and another rule r_j, or if there was a shadowing conflict where r_i was shadowed by r_j, r_j cannot be directly removed. Instead, a constraint is created for the SMT problem (i.e., $greater(i, j) = true$), stating that r_i must have higher priority than r_j in the final anomaly-free policy. More details about the role of these constraints will be explained in Section VI-C2. After all these operations, the whole first block is repeated until the previously explained termination condition is verified.

The second block (lines 35-41) is more similar to traditional conflict resolution strategies. For each conflict between

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

Algorithm 3 for conflict number computation

Input: the cluster g_ℓ, the conflict sets $\mathbb{A}_{contCon,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$
Output: the associative array *conflictNo*, associating the conflict number for each rule in g_ℓ

1: **for each** r_k in g_ℓ **do**
2: $\quad conflictNo[r_k] \leftarrow 0$
3: **for each** $(r_i, r_j) \in \mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell} \cup \mathbb{A}_{corrCon,\ell}$ **do**
4: $\quad conflictNo[r_i] \leftarrow conflictNo[r_i] + 1$
5: $\quad conflictNo[r_j] \leftarrow conflictNo[r_j] + 1$
6: **return** *conflictNo*

two rules r_i and r_j that is still present in the sets $\mathbb{A}_{contCon,\ell}$, $\mathbb{A}_{shadCon,\ell}$, $\mathbb{A}_{corrCon,\ell}$, as the queries of type A were not enough for its removal, a specific query of type B is asked to the administrator. The answer is used to generate again a constraint for the SMT problem, i.e., $greater(i, j) = true$ or $greater(j, i) = true$ depending on the selected action between a^i and a^j. Nevertheless, it is expected that the number of queries of type B is extremely reduced, as any answer to a query of type A could already simultaneously remove a large number of conflicts.

After this algorithm concludes, the cluster g_ℓ has been depured of rules that were unnecessary after the decisions taken by the administrator in terms of conflict resolution. Besides, a partial set of constraints K has been created for the SMT problem, establishing restrictions about rule priority.

2) SMT problem formulation

The rules that are included in the cluster g_ℓ after the execution of Algorithm 2 must be ordered to create an alternative rule list g'_ℓ that satisfies all the constraints of the set K, derived from the answers the administrator made to the posed queries. Guaranteeing the satisfaction of these constraints is essential to respect the decisions of the administrator about the solved conflicts.

In the proposed approach, this reordering problem is formulated as an SMT problem, whose main features have already been discussed in Section III. In particular, all the constraints of the SMT problem are based on two main formal elements:

- Considering $|g_\ell|$ the cardinality of the cluster after the modifications applied in Algorithm 2, $|g_\ell|$ integer variables are created: x_1, x_2, ..., $x_{|g_\ell|}$. Each variable represents the index of the rule of $|g_\ell|$ that is put in a certain position in the final reordered list, where the position is identified by the subscript of the x variables. For simplicity, we assume that the rule positioned at a lower index in g'_ℓ has higher priority. For example, $x_2 = 4$ means that the rule r_4 of g_ℓ will be positioned at index 2 in g'_ℓ. Nevertheless, each variable is initially left free because the solver will have the task of assigning an integer number that satisfies all the constraints of the SMT problem.
- The predicate $greater(i, j)$ is true when the rule r_i of g_ℓ must have a higher priority than the rule r_j of g_ℓ, when both are reordered in g'_ℓ.

Two classes of standard constraints that are always included in the SMT problem, independently of the decisions taken by the administrator in Algorithm 2, are the following:

1) The integer variables x_1, x_2, ..., $x_{|g_\ell|}$ can only be assigned with values corresponding to the subscripts of the rules included in $|g_\ell|$. This requirement is enforced by introducing the following constraint class in the SMT problem, limiting the solution space to those specific integer values:

$$\forall i = \{1, 2, ..., |g_\ell|\}. \bigvee_{r_j \in g_\ell} (x_i = j) \tag{22}$$

2) For each pair of variable x_i and x_j, we assume that the rule whose index is assigned to x_i has higher priority than the rule whose index is assigned to x_j in the final reordered cluster $|g'_\ell|$. This assumption is guaranteed by introducing the following constraint class in the SMT problem, generalized for any pair of variables:

$$\forall i, j \mid i = \{1, 2, ..., |g_\ell|\}, j = \{1, 2, ..., |g_\ell|\}, i < j. \\ greater(x_i, x_j) = true \tag{23}$$

In addition to these basic classes, all the constraints of the set K computed with Algorithm 2 are included in the formulated SMT problem. Those constraints reflect priority relationships between rules as a consequence of the conflict resolution decisions taken by the network administrator. Besides, for any pair of rules r_i and r_j belonging to g_ℓ such that no constraint exists in K (e.g., because no conflict was present between them), then a simple constraint on the $greater$ predicate is included so as to preserve their original mutual ordering, i.e., $greater(i, j) = true$ if r_i had higher priority, $greater(j, i) = true$ otherwise.

An automated SMT solver is then used to find a correct solution to the problem. If any, the solver assigns a specific integer number among the subscripts of the rules in g_ℓ to each variable, and this information is easily used to reorder all rules in the output cluster g'_ℓ.

Finally, as the rule reordering determined by constraints of the SMT problem did not take into account how sub-optimizations were previously removed, Algorithm 1 should simply be applied to the final g'_ℓ, to remove any possible sub-optimization caused by the new order. Then, all clusters can be combined either in an arbitrary way or with a strategic ordering. The former is always a feasible choice, because pairs of rules of different clusters do not have any anomaly. The latter may be possible under specific circumstances, e.g., the administrator knows which rules are matched by network packets the most, and decides to assign higher priority to their cluster.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

D. TIME COMPLEXITY ANALYSIS

Here we discuss the worst-case time complexity analysis of the algorithm formalized for firewall anomaly resolution. As previously mentioned, this algorithm is composed of two main sub-algorithms: the first one for sub-optimization anomaly resolution (Algorithm 1), the second one for conflict anomaly resolution (Algorithm 2, which internally calls Algorithm 3 as a component). For their time complexity analysis, the interactions with the user have not been considered.

The worst-case time complexity of Algorithm 1, used for sub-optimization resolution in a single cluster, can be estimated as the sum of the time complexities of two main sequential code blocks. Lines 2-15 have $O(|\mathbb{A}_{duplSub,\ell} \cup \mathbb{A}_{shadSub,\ell} \cup \mathbb{A}_{anneSub,\ell}|)$ complexity, because $O(1)$ operations are performed on each element of the three anomaly sets sequentially. In fact, even if lines 9-8 are the same for the elements of two sets, they are still executed sequentially on them. Lines 16-20 have $O(|g_\ell| \cdot |\mathbb{A}_{duplSub,\ell} \cup \mathbb{A}_{shadSub,\ell} \cup \mathbb{A}_{anneSub,\ell}|)$ complexity. The difference with respect to the previous code block is that the second one consists of two nested loops. The external one iterates on each element of $R_{toRemove}$, which coincides with the g_ℓ set (except for an element) in the worst case, whereas the internal one requires a number of iterations that is again equal to the number of elements of the three sub-optimization sets. All the other lines of the algorithm are $O(1)$ operations. Summing up, the overall worst-case time complexity for sub-optimization resolution is equal to the time complexity of the second code block, i.e., $O(|g_\ell| \cdot |\mathbb{A}_{duplSub,\ell} \cup \mathbb{A}_{shadSub,\ell} \cup \mathbb{A}_{anneSub,\ell}|)$, because that term is dominant with respect to the other one, where the g_ℓ was not involved.

The worst-complexity of Algorithm 2, which also embeds Algorithm 3, used for conflict resolution in a single cluster, can be estimated as the sum of the time complexities of three sequential code blocks. The first block is the less impacting on the overall time complexity. Indeed, lines 1-2 have $O(|g_\ell|)$ complexity, because a single $O(1)$ operation is performed on each element of the modified cluster. Instead, the second block is the most impacting, because it also includes the whole Algorithm 3. This second block, composed of lines 4-34, is a loop of three sub-blocks on all the elements of g_ℓ in the worst case:

1) line 6 is a call to Algorithm 3, whose worst-case time complexity is $O(|g_\ell|) + O(|\mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell} \cup \mathbb{A}_{corrCon,\ell}|)$ because it is composed of two sequential loops one on the rules of the cluster, the other one on the elements of the three conflict sets;
2) line 7 is a call to the quick sort algorithm for rule reordering, and therefore it has $O(|g_\ell| \cdot log(|g_\ell|))$;
3) lines 8-34 is another nested loop, and has a $O(|g_\ell| \cdot |\mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell}| \cdot |\mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell} \cup \mathbb{A}_{corrCon,\ell}|^2)$ complexity in the worst case, that occurs when the condition in line 16 is true.

Therefore, by summing the different contributions and adding the multiplying factor due to the external loop, the time complexity related to the second block has an overall $O(|g_\ell|^2 \cdot (log(|g_\ell|) + |\mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell}| \cdot |\mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell} \cup \mathbb{A}_{corrCon,\ell}|^2))$ complexity. Then, the third block, composed of lines 35-41, simply have $O(|\mathbb{A}_{contCon,\ell} \cup \mathbb{A}_{shadCon,\ell} \cup \mathbb{A}_{corrCon,\ell}|)$ complexity, as it is composed of a single loop on the elements of the three sets. Summing up, the time complexity of the whole algorithm is the one of the second block, because it is the dominant term for the asymptotic complexity.

VII. IMPLEMENTATION AND CASE STUDY EVALUATION

The proposed optimized approach has been implemented as a Java-based framework, which offers an interactive view for human administrators to read the questions that are posed by the methodology, and to answer them so that the approach can subsequently resolve the conflicts. The tool interacts with the Java APIs of Z3, a state-of-the-art theorem prover developed by Microsoft Research [43], for the resolution of the SMT problem.

The framework has been validated on a machine with an Intel i7-6700 CPU at 3.40 GHz and 32GB of RAM. Specifically, it has been evaluated on a series of use cases, so as to assess if the produced result is actually compliant with the objectives of the human administrators. Here, we describe a complete use case, which was useful to stress the defined algorithm in the operations of anomaly resolution, because all anomaly types are included.

TABLE 1 reports a list of rules composing a firewall configuration. This list size has been chosen for multiple reasons. On the one hand, the scope of our proposal consists in intra-firewall anomaly resolution, a scenario where the number of rules is limited because otherwise humans would not be able to manage all of them by themselves. On the other hand, the role of this validating case study is also to explain how the proposed approach works in all the phases. For sake of visualization simplicity, the conditions of all rules specified in TABLE 1 only select packets depending on the values of their source and destination IP addresses. In this way, it will be possible to graphically represent their packet spaces as rectangles in a 2-space, instead of penteracts in a 5-space.

Initially, the anomaly analysis algorithm identifies all anomalies (i.e., sub-optimizations and conflicts) among the rules of TABLE 1, classifying them into the sets $\mathbb{A}_{contCon}$, $\mathbb{A}_{shadCon}$, $\mathbb{A}_{corrCon}$, $\mathbb{A}_{duplSub}$, $\mathbb{A}_{shadSub}$ and $\mathbb{A}_{unneSub}$, according to the anomaly classification metrics discussed in Section V:

- $\mathbb{A}_{contCon} = \{\{r_6, r_{11}\}\}$;
- $\mathbb{A}_{shadCon} = \{\{r_4, r_{13}\}\}$;
- $\mathbb{A}_{corrCon} = \{\{r_1, r_2\}, \{r_1, r_{11}\}, \{r_1, r_{17}\}, \{r_2, r_{14}\}, \{r_3, r_4\}, \{r_4, r_5\}, \{r_4, r_9\}, \{r_4, r_{14}\}, \{r_7, r_{16}\}, \{r_8, r_{16}\}\}$;
- $\mathbb{A}_{duplSub} = \{\{r_5, r_9\}\}$;
- $\mathbb{A}_{shadSub} = \{\{r_4, r_{15}\}\}$;
- $\mathbb{A}_{unneSub} = \{\{r_{10}, r_{14}\}\}$.

After the anomaly analysis is completed, the rules are clustered, so that any two rules belonging to the same cluster have non-disjoint conditions:

- $g_1 = [r_1, r_2, r_3, r_4, r_5, r_6, r_9, r_{10}, r_{11}, r_{13}, r_{14}, r_{15}, r_{17}]$;

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

TABLE 1: Initial firewall policy of the use case

Rule	Source IP Address	Destination IP Address	Action	Priority
r_1	[124.60.0.22, 124.60.0.32]	[152.88.36.14, 124.60.0.78]	deny	1
r_2	[124.60.0.16, 124.60.0.72]	[152.88.36.64, 124.60.0.92]	allow	2
r_3	[124.60.0.134, 124.60.0.146]	[152.88.36.42, 124.60.0.58]	deny	3
r_4	[124.60.0.84, 124.60.0.146]	[152.88.36.22, 124.60.0.100]	allow	4
r_5	[124.60.0.130, 124.60.0.140]	[152.88.36.74, 124.60.0.126]	allow	5
r_6	[124.60.0.26, 124.60.0.62]	[152.88.36.24, 124.60.0.32]	deny	6
r_7	[124.60.0.162, 124.60.0.192]	[152.88.36.124, 124.60.0.132]	allow	7
r_8	[124.60.0.186, 124.60.0.214]	[152.88.36.88, 124.60.0.108]	allow	8
r_9	[124.60.0.130, 124.60.0.140]	[152.88.36.74, 124.60.0.126]	allow	9
r_{10}	[124.60.0.76, 124.60.0.80]	[152.88.36.120, 124.60.0.130]	deny	10
r_{11}	[124.60.0.26, 124.60.0.62]	[152.88.36.24, 124.60.0.32]	allow	11
r_{12}	[124.60.0.180, 124.60.0.196]	[152.88.36.20, 124.60.0.30]	deny	12
r_{13}	[124.60.0.110, 124.60.0.120]	[152.88.36.28, 124.60.0.36]	deny	13
r_{14}	[124.60.0.62, 124.60.0.98]	[152.88.36.82, 124.60.0.144]	deny	14
r_{15}	[124.60.0.94, 124.60.0.102]	[152.88.36.54, 124.60.0.60]	allow	15
r_{16}	[124.60.0.180, 124.60.0.188]	[152.88.36.92, 124.60.0.126]	deny	16
r_{17}	[124.60.0.10, 124.60.0.24]	[152.88.36.24, 124.60.0.32]	allow	17

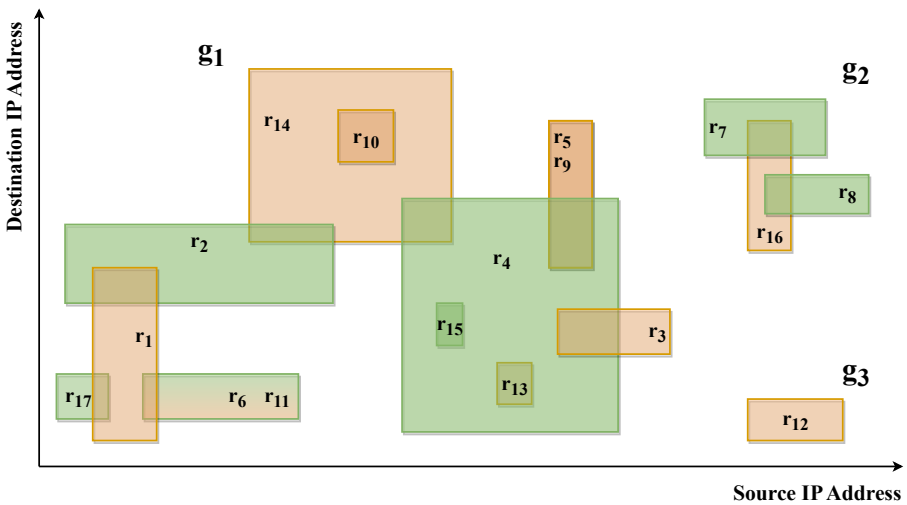

FIGURE 3: Visual representation of the clusters

- $g_2 = [r_7, r_8, r_{16}]$;
- $g_3 = [r_{12}]$.

The result of the clustering operation is also graphically shown in FIGURE 3.

After all rules have been clustered, the anomaly resolution algorithm is applied to each cluster independently from the other ones, as it can be parallelized.

The cluster with the higher number of rules and anomalies is g_1. For g_1, the six sets $\mathbb{A}_{contCon,1}$, $\mathbb{A}_{shadCon,1}$, $\mathbb{A}_{corrCon,1}$, $\mathbb{A}_{duplSub,1}$, $\mathbb{A}_{shadSub,1}$ and $\mathbb{A}_{unneSub,1}$ are derived from the original sets $\mathbb{A}_{contCon}$, $\mathbb{A}_{shadCon}$, $\mathbb{A}_{corrCon}$, $\mathbb{A}_{duplSub}$, $\mathbb{A}_{shadSub}$ and $\mathbb{A}_{unneSub}$, by keeping only the anomalies that involve rules of g_1. Specifically:

- $\mathbb{A}_{contCon,1} = \{\{r_6, r_{11}\}\}$;
- $\mathbb{A}_{shadCon,1} = \{\{r_4, r_{13}\}\}$;
- $\mathbb{A}_{corrCon,1} = \{\{r_1, r_2\}, \{r_1, r_{11}\}, \{r_1, r_{17}\}, \{r_2, r_{14}\}, \{r_3, r_4\}, \{r_4, r_5\}, \{r_4, r_9\}, \{r_4, r_{14}\}\}$;
- $\mathbb{A}_{duplSub,1} = \{\{r_5, r_9\}\}$;
- $\mathbb{A}_{shadSub,1} = \{\{r_4, r_{15}\}\}$;
- $\mathbb{A}_{unneSub,1} = \{\{r_{10}, r_{14}\}\}$.

First, the sub-optimization resolution strategy, previously formalized in Algorithm 1, is applied to all the sub-optimizations included in the sets $\mathbb{A}_{duplSub,1}$, $\mathbb{A}_{shadSub,1}$ and $\mathbb{A}_{unneSub,1}$:

- as the sub-optimization between r_5 and r_9 is a duplication anomaly, one of the two rules is removed, for simplicity the one with lower priority, i.e., r_9;
- as the sub-optimization between r_4 and r_{15} is a shadowing redundancy, the rule with lower priority is removed, i.e., r_{15};
- as the sub-optimization between r_{10} and r_{14} is an unnecessary anomaly, the rule with higher priority is removed, i.e., r_{10}.

After these removal operations, the cluster g_1 has become as follows: $g_1 = [r_1, r_2, r_3, r_4, r_5, r_6, r_{11}, r_{13}, r_{14}, r_{16}]$. Instead, the three conflict sets have been modified as follows: $\mathbb{A}_{contCon,1} = \{\{r_6, r_{11}\}\}$, $\mathbb{A}_{shadCon,1} = \{\{r_4, r_{13}\}\}$, $\mathbb{A}_{corrCon,1} = \{\{r_1, r_2\}, \{r_1, r_{11}\}, \{r_1, r_{17}\}, \{r_2, r_{14}\}, \{r_3, r_4\}, \{r_4, r_5\}, \{r_4, r_{14}\}\}$.

Second, the conflict resolution strategy, previously for-

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

TABLE 2: Output firewall policy of the use case

Rule	Source IP Address	Destination IP Address	Action	Priority
r_4	[124.60.0.84, 124.60.0.146]	[152.88.36.22, 124.60.0.100]	allow	1
r_3	[124.60.0.134, 124.60.0.146]	[152.88.36.42, 124.60.0.58]	deny	2
r_5	[124.60.0.130, 124.60.0.140]	[152.88.36.74, 124.60.0.126]	allow	3
r_{14}	[124.60.0.62, 124.60.0.98]	[152.88.36.82, 124.60.0.144]	deny	4
r_1	[124.60.0.22, 124.60.0.32]	[152.88.36.14, 124.60.0.78]	deny	5
r_2	[124.60.0.16, 124.60.0.72]	[152.88.36.64, 124.60.0.92]	allow	6
r_{11}	[124.60.0.26, 124.60.0.62]	[152.88.36.24, 124.60.0.32]	allow	7
r_{17}	[124.60.0.10, 124.60.0.24]	[152.88.36.24, 124.60.0.32]	allow	8
r_{16}	[124.60.0.180, 124.60.0.188]	[152.88.36.92, 124.60.0.126]	deny	9
r_7	[124.60.0.162, 124.60.0.192]	[152.88.36.124, 124.60.0.132]	allow	10
r_8	[124.60.0.186, 124.60.0.214]	[152.88.36.88, 124.60.0.108]	allow	11
r_{12}	[124.60.0.180, 124.60.0.196]	[152.88.36.20, 124.60.0.30]	deny	12

malized in Algorithm 2, is applied to solve all the conflicts throughout an interaction with the human administrator.

In the first iteration of the first code block of Algorithm 2, considering the content of the three sets $\mathbb{A}_{contCon,1}$, $\mathbb{A}_{shadCon,1}$, $\mathbb{A}_{corrCon,1}$, according to the computation of the values for the associative array *conflictNo* performed through Algorithm 3, the rule with the higher number of conflicts is r_4, as it appears four times in the conflict sets. Therefore, a query of type A is posed to the administrator, asking them if all packets satisfying the condition c^4 should be subject to the action a^4. Supposing that the administrator answers "yes" to this question, all the conflicts involving r_4 are solved. When the shadowing anomaly between r_4 and r_{13} is solved, r_{13} is directly removed from the cluster g_1. Instead, when the three correlation anomalies where r_4 is involved are solved (i.e., the ones with r_3, r_5 and r_{14}), three constraints are generated for the SMT problem: $greater(4, 3) = true$, $greater(4, 5) = true$, $greater(4, 14) = true$. After these operations, the three conflict sets have become as follows: $\mathbb{A}_{contCon,1} = \{\{r_6, r_{11}\}\}$, $\mathbb{A}_{shadCon,1} = \emptyset$, $\mathbb{A}_{corrCon,1} = \{\{r_1, r_2\}, \{r_1, r_{11}\}, \{r_1, r_{17}\}, \{r_2, r_{14}\}\}$.

In the second iteration of the first code block of Algorithm 2, the rule r_1 is selected as the subject of a query of type A, as it is involved in three conflicts. Supposing that the administrator again answers "yes", other three correlation conflicts are solved by generating the corresponding constraints for the SMT problem: $greater(1, 2) = true$, $greater(1, 11) = true$, $greater(4, 14) = true$. After these operations, the three conflict sets have become as follows: $\mathbb{A}_{contCon,1} = \{\{r_6, r_{11}\}\}$, $\mathbb{A}_{shadCon,1} = \emptyset$, $\mathbb{A}_{corrCon,1} = \{\{r_2, r_{14}\}\}$.

In the third iteration of the first code block of Algorithm 2, each rule for which not all conflict has not yet been solved (r_2, r_6, r_{11}, r_{14}) is involved in a single conflict, i.e., $conflictNo(r_2) = conflictNo(r_6) = conflictNo(r_{11}) = conflictNo(r_{14}) = 1$. Therefore, it is not possible anymore to ask queries of type A to solve multiple conflicts simultaneously.

After this termination condition is reached for the first code block of Algorithm 2, in the second code block two queries of type B are asked to the administrator, related to the two remaining conflicts, so that the user can singularly address them. Supposing that the administrator decides that in the remaining contradiction conflict the rule that must be preserved is r_{11}, then r_6 is simply removed from the cluster g_1. Instead, supposing that the administrator decides that in the remaining correlation conflict the winning rule is r_{14}, then a constraint for the SMT problem is generated to establish its priority with respect to r_2: $greater(14, 2) = true$.

After Algorithm 2 ends, the cluster g_1 has become as follows: $g_1 = [r_1, r_2, r_3, r_4, r_5, r_{11}, r_{14}, r_{16}]$. Consequently, eight integer variables x_1, x_2, ..., x_8 are created for establishing the ordering among those eight rules. These variables are used to create an SMT problem, composed of all the constraints derived from the execution of Algorithm 2, and all the basic constraints illustrated in Section VI-C2. After creating this problem, the Z3 solver finds a correct solution based on this variable assignment: $x_1 = 4$, $x_2 = 3$, $x_3 = 5$, $x_4 = 14$, $x_5 = 1$, $x_6 = 2$, $x_7 = 11$, $x_8 = 17$. Consequently, the reordered cluster is $g'_1 = [r_4, r_3, r_5, r_{14}, r_1, r_2, r_{11}, r_{17}]$. Indeed, there may have been other valid solutions that are compliant with all the previously described constraints. However, as all of them are valid, it is enough that the solver finds one of them.

The whole process repeats for the other two clusters, even though their related anomalies are much less and easier to solve. On the one hand, for the cluster g_2, supposing that the administrator answers "yes" to a query of type A based on the rule r_{16}, both correlation conflicts are solved in a single step, and the resulting reordering operation is trivial, leading to: $g'_2 = [r_{16}, r_7, r_8]$. On the other hand, the cluster g_3 is composed of a single rule, because r_{12} does not have any anomalous relationship with any other rule. Therefore, no query is actually needed for that cluster.

Finally, the three reordered clusters are combined. For simplicity, we can assume that there is no specific requirement from the administrator about their relative priority, and therefore the final solution simply concatenates g'_1, g'_2 and g'_3. The resulting firewall policy is reported in TABLE 2. Note that Algorithm 1 should be reapplied to this policy to solve any sub-optimization that may have been caused by this reordering, but in this specific use case no new sub-optimization was actually created.

The application of the proposed methodology to this use case showcases the benefits in terms of optimization. In

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

particular, the solution to the anomaly resolution problem was identified through only five queries (three queries of type A and two queries of type B), which were enough to solve all fifteen anomalies. In the traditional manual approach for anomaly resolution, the human administrator should have manually solved all the anomalies one by one, with a much higher overhead in his workload. Moreover, the reordered policy is proved to be a correct solution, thanks to the correctness-by-construction principle on which the SMT problem formulation is based on.

VIII. CONCLUSIONS AND FUTURE WORK

This paper presented an assisted optimized approach for firewall policy anomaly analysis and resolution. Human administrators are guided toward the resolution of all the anomalies afflicting a manually written policy by answering a series of well-posed queries. The formulation of these queries is defined so as to reduce their number, thus lessening the workload of the administrators. Satisfiability checking is also employed to formulate the anomaly-free rule reordering problem as an SMT problem, thus providing assurance about its correctness. The proposed approach is convenient with respect to both manual anomaly resolution, which commonly fails in solving all anomalies or introduces other ones, and automatic strategies, which are potentially dangerous because their conflict-fixing process is not under the administrators control. These benefits, including the optimization in terms of workload reduction, have been showcased by describing how the framework implementing our proposed approach is applied to a realistic comprehensive use case.

As future work, we plan to extend the proposed models and algorithms to make them compliant with problems related to attribute-based access control, thus providing a broader generalization of the approach. We also plan to perform empirical assessments with network administrators to assess how much the proposed approach can help administrators in reducing the number of anomalies in real-world environments. Finally, we will investigate an extension of the proposed resolution strategy to inter-firewall policy anomaly, for which additional features must be accounted, such as rule matching number, firewall location in the defense-in-depth networking environment, and bi-directional configuration.

REFERENCES

[1] P. P. Mukkamala, S. Rajendran, A survey on the different firewall technologies, Inter. J. of Engin. Appl. Scien. and Tech. 5 (1) (2020).

[2] Verizon, 2022 Data Breach Investigations Report, Available: https://www.verizon.com/business/resources/reports/dbir/, Visited: 2023-07-10 (2022).

[3] R. Robere, A. Kolokolova, V. Ganesh, The proof complexity of SMT solvers, in: Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II, Vol. 10982 of Lecture Notes in Computer Science, Springer, 2018, pp. 275–293. doi:10.1007/978-3-319-96142-2_18.
URL https://doi.org/10.1007/978-3-319-96142-2_18

[4] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser, Terminology for Policy-Based Management, RFC 3198 (Informational), http://www.rfc-editor.org/rfc/rfc3198.txt (November 2001).

[5] A. A. Jabal, M. Davari, E. Bertino, C. Makaya, S. B. Calo, D. C. Verma, A. Russo, C. Williams, Methods and tools for policy analysis, ACM Comput. Surv. 51 (6) (2019) 121:1–121:35. doi:10.1145/3295749.
URL https://doi.org/10.1145/3295749

[6] A. Panda, O. Lahav, K. J. Argyraki, M. Sagiv, S. Shenker, Verifying reachability in networks with mutable datapaths, in: Proc. of the 14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017, USENIX Association, 2017, pp. 699–718.
URL https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/panda-mutable-datapaths

[7] F. Valenza, S. Spinoso, R. Sisto, Formally specifying and checking policies and anomalies in service function chaining, J. Netw. Comput. Appl. 146 (2019). doi:10.1016/j.jnca.2019.102419.
URL https://doi.org/10.1016/j.jnca.2019.102419

[8] Y. Kim, M. Kang, Formal verification of sdn-based firewalls by using TLA+, IEEE Access 8 (2020) 52100–52112. doi:10.1109/ACCESS.2020.2979894.
URL https://doi.org/10.1109/ACCESS.2020.2979894

[9] D. Bringhenti, G. Marchetto, R. Sisto, S. Spinoso, F. Valenza, J. Yusupov, Improving the formal verification of reachability policies in virtualized networks, IEEE Trans. Netw. Serv. Manag. 18 (1) (2021) 713–728. doi:10.1109/TNSM.2020.3045781.

[10] M. A. Rahman, E. Al-Shaer, Automated synthesis of distributed network access controls: A formal framework with refinement, IEEE Trans. Parallel Distributed Syst. 28 (2) (2017) 416–430. doi:10.1109/TPDS.2016.2585108.
URL https://doi.org/10.1109/TPDS.2016.2585108

[11] N. Schnepf, R. Badonnel, A. Lahmadi, S. Merz, Rule-based synthesis of chains of security functions for software-defined networks, Electron. Commun. Eur. Assoc. Softw. Sci. Technol. 76 (2018). doi:10.14279/tuj.eceasst.76.1075.
URL https://doi.org/10.14279/tuj.eceasst.76.1075

[12] D. Bringhenti, F. Valenza, C. Basile, Toward cybersecurity personalization in smart homes, IEEE Secur. Priv. 20 (1) (2022) 45–53. doi:10.1109/MSEC.2021.3117471.
URL https://doi.org/10.1109/MSEC.2021.3117471

[13] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, J. Yusupov, Automated firewall configuration in virtual networks, IEEE Trans. Dependable Secur. Comput. 20 (2) (2023) 1559–1576. doi:10.1109/TDSC.2022.3160293.

[14] F. Cuppens, N. Cuppens-Boulahia, M. B. Ghorbel, High level conflict management strategies in advanced access control models, in: Proceedings of the First Workshop in Information and Computer Security, ICS@SYNASC 2006, Timisoara, Romania, September 30, 2006, Vol. 186 of Electronic Notes in Theoretical Computer Science, Elsevier, 2006, pp. 3–26. doi:10.1016/j.entcs.2007.01.064.
URL https://doi.org/10.1016/j.entcs.2007.01.064

[15] W. M. Fitzgerald, F. Turkmen, S. N. Foley, B. O'Sullivan, Anomaly analysis for physical access control security configuration, in: 7th International Conference on Risks and Security of Internet and Systems, CRiSIS 2012, Cork, Ireland, October 10-12, 2012, IEEE Computer Society, 2012, pp. 1–8. doi:10.1109/CRISIS.2012.6378953.
URL https://doi.org/10.1109/CRISIS.2012.6378953

[16] M. Davari, M. Zulkernine, Policy modeling and anomaly detection in ABAC policies, in: Risks and Security of Internet and Systems - 16th International Conference, CRiSIS 2021, Virtual Event, Ames, USA, November 12-13, 2021, Revised Selected Papers, Vol. 13204 of Lecture Notes in Computer Science, Springer, 2021, pp. 137–152. doi:10.1007/978-3-031-02067-4_9.
URL https://doi.org/10.1007/978-3-031-02067-4_9

[17] K. Vijayalakshmi, V. Jayalakshmi, Identifying considerable anomalies and conflicts in abac security policies, in: 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 1273–1280. doi:10.1109/ICICCS51141.2021.9432162.

[18] E. S. Al-Shaer, H. H. Hamed, Firewall policy advisor for anomaly discovery and rule editing, in: Proc. of the IEEE Eighth International Symposium on Integrated Network Management., IEEE, 2003, pp. 17–30. doi:10.1109/inm.2003.1194157.

[19] E. Al-Shaer, H. Hamed, R. Boutaba, M. Hasan, Conflict classification and analysis of distributed firewall policies, IEEE Journal on Selected Areas in Communications 23 (10) (2005) 2069–2084. doi:10.1109/JSAC.2005.854119.

[20] L. Yuan, H. Chen, J. Mai, C.-n. Chuah, FIREMAN: A Toolkit for FIREwall

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3328194

Modeling and ANalysis, in: Proc. of the IEEE Symposium on Security and Privacy, IEEE, 2006, pp. 199–213. doi:10.1109/SP.2006.16.

[21] K. Golnabi, R. K. Min, L. Khan, E. Al-Shaer, Analysis of firewall policy rules using data mining techniques, in: Proc. of the10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), IEEE, 2006, pp. 305–315. doi:10.1109/NOMS.2006.1687561.

[22] T. Abbes, A. Bouhoula, M. Rusinowitch, An inference system for detecting firewall filtering rules anomalies, in: Proc. of the ACM symposium on Applied computing (SAC08), ACM, 2008, pp. 2122–2128. doi:10.1145/1363686.1364197.

[23] C. Togay, A. Kasif, C. Catal, B. Tekinerdogan, A firewall policy anomaly detection framework for reliable network security, IEEE Trans. Reliab. 71 (1) (2022) 339–347. doi:10.1109/TR.2021.3089511.
URL https://doi.org/10.1109/TR.2021.3089511

[24] Z. Lin, Z. Yao, Firewall anomaly detection based on double decision tree, Symmetry 14 (12) (2022) 2668. doi:10.3390/sym14122668.
URL https://doi.org/10.3390/sym14122668

[25] A. Komadina, I. Kovacevic, B. Stengl, S. Gros, Detecting anomalies in firewall logs using artificially generated attacks, in: 17th International Conference on Telecommunications, ConTEL 2023, Graz, Austria, July 11-13, 2023, IEEE, 2023, pp. 1–8. doi:10.1109/ConTEL58387.2023.10198912.
URL https://doi.org/10.1109/ConTEL58387.2023.10198912

[26] A. Andalib, S. M. Babamir, Anomaly detection of policies in distributed firewalls using data log analysis, The Journal of Supercomputing (2023) 1–42doi:https://doi.org/10.1007/s11227-023-05417-7.

[27] T. Tran, E. S. Al-Shaer, R. Boutaba, Policyvis: Firewall security policy visualization and inspection, in: P. Anderson (Ed.), Proceedings of the 21th Large Installation System Administration Conference, LISA 2007, Dallas, Texas, USA, November 11-16, 2007, USENIX, 2007, pp. 1–16.
URL http://www.usenix.org/events/lisa07/tech/tran.html

[28] C. Chao, A flexible and feasible anomaly diagnosis system for internet firewall rules, in: 13th Asia-Pacific Network Operations and Management Symposium, APNOMS 2011, Taipei, Taiwan, September 21-23, 2011, IEEE, 2011, pp. 1–8. doi:10.1109/APNOMS.2011.6077012.
URL https://doi.org/10.1109/APNOMS.2011.6077012

[29] F. Mansmann, D. A. Keim, S. C. North, B. Rexroad, D. Sheleheda, Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats, IEEE Trans. Vis. Comput. Graph. 13 (6) (2007) 1105–1112. doi:10.1109/TVCG.2007.70522.
URL https://doi.org/10.1109/TVCG.2007.70522

[30] F. Mansmann, T. Göbel, W. R. Cheswick, Visual analysis of complex firewall configurations, in: D. Schweitzer, D. Quist (Eds.), 9th International Symposium on Visualization for Cyber Security, VizSec '12, Seattle, WA, USA, October 15, 2012, ACM, 2012, pp. 1–8. doi:10.1145/2379690.2379691.
URL https://doi.org/10.1145/2379690.2379691

[31] U. Kim, J. Kang, J. Lee, H. Kim, S. Jung, Practical firewall policy inspection using anomaly detection and its visualization, Multim. Tools Appl. 71 (2) (2014) 627–641. doi:10.1007/s11042-013-1673-8.
URL https://doi.org/10.1007/s11042-013-1673-8

[32] H. Kim, S. Ko, D. S. Kim, H. K. Kim, Firewall ruleset visualization analysis tool based on segmentation, in: 14th IEEE Symposium on Visualization for Cyber Security, VizSec 2017, Phoenix, AZ, USA, October 2, 2017, IEEE Computer Society, 2017, pp. 1–8. doi:10.1109/VIZSEC.2017.8062196.
URL https://doi.org/10.1109/VIZSEC.2017.8062196

[33] H. Lee, S. Lee, K. Kim, H. K. Kim, Hsviz: Hierarchy simplified visualizations for firewall policy analysis, IEEE Access 9 (2021) 71737–71753. doi:10.1109/ACCESS.2021.3077146.
URL https://doi.org/10.1109/ACCESS.2021.3077146

[34] T. Kim, T. Kwon, J. Lee, J. Song, F/wvis: Hierarchical visual approach for effective optimization of firewall policy, IEEE Access 9 (2021) 105989–106004. doi:10.1109/ACCESS.2021.3100141.
URL https://doi.org/10.1109/ACCESS.2021.3100141

[35] A. X. Liu, M. G. Gouda, Complete Redundancy Detection in Firewalls, in: Proc. of the 19th Working Conference on Data and Applications Security (IFIP WG 11.3), Springer, 2005, pp. 193–206. doi:10.1007/11535706_15.

[36] F. Cuppens, N. Cuppens-Boulahia, J. Garcia-Alfaro, Detection and Removal of Firewall Misconfiguration, in: Proc. of the 2005 IASTED International Conference on Communication, Network and Information Security, IASTED, 2005, pp. 154–162.

[37] H. Hu, G.-J. Ahn, K. Kulkarni, FAME: a firewall anomaly management environment, in: Proc. of the 3rd ACM workshop on Assurable and usable security configuration (SafeConfig10), ACM, Chicago, USA, 2010, pp. 17–26. doi:10.1145/1866898.1866902.

[38] H. Hu, G. J. Ahn, K. Kulkarni, Detecting and resolving firewall policy anomalies, IEEE Trans. Dependable Sec. Comput. 9 (3) (2012) 318–331. doi:10.1109/TDSC.2012.20.

[39] S. Ferraresi, S. Pesic, L. Trazza, A. Baiocchi, Automatic conflict analysis and resolution of traffic filtering policy for firewall and security gateway, in: Proc. of the IEEE International Conference on Communications (ICC'07), IEEE, 2007, pp. 1304–1310. doi:10.1109/ICC.2007.220.

[40] A. Jeffrey, T. Samak, Model Checking Firewall Policy Configurations, in: Proc. of the IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY2009), IEEE, 2009, pp. 60–67. doi:10.1109/POLICY.2009.32.

[41] F. Cuppens, N. Cuppens-Boulahia, J. Garcia-Alfaro, Detection of network security component misconfiguration by rewriting and correlation, in: Proc. of the 1th joint conference on security in network architectures (SAR) and security of information systems (SSI), 2006, pp. 6–9.

[42] D. Detlefs, G. Nelson, J. B. Saxe, Simplify: a theorem prover for program checking, J. ACM 52 (3) (2005) 365–473. doi:10.1145/1066100.1066102.

[43] L. M. de Moura, N. S. Bjørner, Z3: an efficient SMT solver, in: Proc. of Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings, Vol. 4963 of Lecture Notes in Computer Science, Springer, 2008, pp. 337–340. doi:10.1007/978-3-540-78800-3_24.

[44] C. W. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, C. Tinelli, CVC4, in: Proc. of Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings, Vol. 6806 of Lecture Notes in Computer Science, Springer, 2011, pp. 171–177. doi:10.1007/978-3-642-22110-1_14.

DANIELE BRINGHENTI received the M.Sc. degree (summa cum laude) and the Ph.D. degree (summa cum laude) in computer engineering from the Politecnico di Torino, Torino, Italy, in 2019 and 2022 respectively, where he is currently a Postdoctoral Researcher. His research interests include novel networking technologies, automatic orchestration and configuration of security functions in virtualized networks, formal verification of network security policies.

LUCIA SENO received the B.S. and M.S. degrees in automation engineering and the Ph.D. degree in information engineering from the University of Padova, Padova, Italy, in 2004, 2007, and 2011, respectively. Since then, she has been with the Institute of Electronics, Information Engineering, and Telecommunications, National Research Council of Italy, where she is currently a Researcher. Her research interests include formal verification of systems security and communication protocols.

FULVIO VALENZA received the M.Sc. degree (summa cum laude) and the Ph.D. degree (summa cum laude) in computer engineering from the Politecnico di Torino, Torino, Italy, in 2013 and 2017, respectively, where he is currently a Tenure-Track Assistant Professor. His research activity focuses on network security policies, orchestration and management of network security functions in SDN/NFV-based networks, and threat modeling.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4