Towards a Framework for Automatic Firewalls Configuration via Argumentation Reasoning

Erisa Karafili
University of Southampton
e.karafili@soton.ac.uk

Fulvio Valenza
Politecnico di Torino
fulvio.valenza@polito.it

Yichen Chen
Imperial College London
yichen.chen@imperial.ac.uk

Emil C. Lupu
Imperial College London
e.c.lupu@imperial.ac.uk

***Abstract*—Firewalls have been widely used to protect not only small and local networks but also large enterprise networks. The configuration of firewalls is mainly done by network administrators, thus, it suffers from human errors. This paper aims to solve the network administrators' problem by introducing a formal approach that helps to configure centralized and distributed firewalls and automatically generate conflict-free firewall rules. We propose a novel framework, called *ArgoFiCo*, which is based on argumentation reasoning. Our framework automatically populates the firewalls of a network, given the network topology and the high-level requirements that represent how the network should behave. *ArgoFiCo* provides two strategies for firewall rules distribution.**

***Index Terms*—Firewall, Security policy, Network security, Argumentation reasoning**

I. Introduction

Nowadays, attacks carried out through the network space are very tangible menaces. About 70% of cyber-attacks discovered in 2019, had as target network/cloud services and resources [19]. Enforcing network security in a large and distributed network is a complex and sensitive task. Firewalls have been widely used as the first frontier to protect not only small individual and local networks but also large enterprise networks. However, the correct and optimal configuration of firewalls in a network is challenging, since the main decisions are taken by the network administrator that can bring human errors. Moreover, it is difficult to maintain a unique and coherent strategy, when there are different administrators that configure large networks.

The impact of the human error factor, in continuously growing networks, is not negligible, as nearly 60% of the security breaches occurred in 2019, are attributable to errors made by systems and network administrators [19]. The typical approach of security administrators is *trial and error*. When one or more misconfigurations are reported, the administrators correct them by creating ad hoc rules and repeat the process until no more errors are discovered. This methodology, although simple, is only a temporary palliative as it can produce serious maintenance problems in the future. Guaranteeing the absence of misconfigurations is difficult, without an appropriate software tool. Therefore, there is a need for a practical solution to evaluate the policy actually enforced [2].

In this paper, we introduce a novel framework based on a formal approach that helps to configure centralized and distributed firewalls and to automatically generate conflict-free firewall rules. Our approach, called *ArgoFiCo*, is based on argumentation reasoning. This solution automatically generates the firewalls' configuration when the user gives as input the network topology and the high-level requirements. These requirements describe how the network should behave. *ArgoFiCo*'s result is the configuration of the network's firewalls.

The reasoning core of *ArgoFiCo* is based on a *preference-based argumentation* reasoning [8], [10], which permits us to deal with conflicting rules and order them. In particular, we use the Gorgias [8] tool, which is a preference-based argumentation reasoning tool. The conflicts between rules are the anomalies that can be created between rules of the same firewall or different ones. Our approach analyzes the requirements, by using a similar methodology as the one provided in [18], and automatically identifies the conflicts/anomalies between them. The conflicts/anomalies are automatically solved by ordering the rules. The ordering is made following various strategies for conflicts/anomalies resolution. This approach permits to relieve the work of the network administrator, by automatically configuring the firewalls of the network. Furthermore, as the process is automatic, it reduces the chances of human errors.

The introduced approach provides flexibility to the administrators that can decide between two ordering strategies, as our tool provides two types of firewall rules placement. The first type is a general one, called the *max configuration*, where the rules are distributed in the firewalls to permit or deny the flow of information in all possible paths between source and destination. In the second type of ordering, called the *min configuration*, the rules are distributed in an efficient way, in order to have the minimal number of firewall rules, by considering only the path derived from the routing between the source and destination.

We present in Section II some related work for firewall configuration. In Section III we provide a general overview of the introduced approach together with the used formalism, which is a preference-based argumentation reasoning. We provide an example of application of *ArgoFiCo* in Section IV. We conclude and present the future work in Section V.

II. Related Work

Policy-based management in network security mainly focuses on *policy analysis* and *policy refinement*. The main contributions in policy analysis deal with *anomaly analysis*

of firewall policies. Anomaly analysis looks for incorrect policy specifications that administrators may introduce in the network. It includes checks for potential errors, conflicts, and sub-optimizations affecting either a single policy or a set of security policies [17]. The work in [1] introduces an analysis of filtering configurations via First-Order Logic (FOL) formulas, and it classifies and analyzes the local anomalies, arising in a single firewall (*intra-firewall* anomaly) and the global ones, taking into account several distributed firewalls (*inter-firewall* anomaly). Another interesting work that performers a policy analysis is introduced in [3], where the authors use an argumentation framework for specifying security requirements. This work does not deal with the configuration of different firewalls inside the same network and with inter-firewall anomalies. A formal model for detecting anomalies among different security functions (inter-function anomalies) is introduced in [**?**], [5], [18]. The proposed model detects several kinds of errors and anomalies that originate from correlations between configuration rules of different network functions.

Policy refinement is the process that "determines the resources needed to satisfy policy requirements and translates high-level policies into operational policies that may be enforced by the system" [6]. FIRMATO [4] is the first proposed solution to the policy refinement problem that supports packets filtering firewalls. FACE [20] is a firewall analysis and configuration engine that uses a similar approach to FIRMATO. A refinement model was introduced in [6], which allows the translation of high-level security requirements into low-level configuration settings for the virtual network security functions.

III. The *ArgoFiCo* framework

The goal of our framework is to automatically configure the firewalls in a given network. Our framework, called *ArgoFiCo* based on argumentation reasoning, configures the centralized and distributed firewalls of the network, given the network topology, which represents how the entities of the network are related, and a set of high-level security requirements about the network. An overall presentation of the constructed framework is given in Figure 1. The configured firewalls are constructed by scratch and do not have inter- or intra-firewall anomalies/conflicts, as such were captured and solved. The user (i.e., the network administrator) provides the inputs that are analyzed by the framework.

ArgoFiCo's first step is the analysis of the given high-level requirements and the identification of various conflicts and anomalies. The conflicts/anomalies identification and later resolution are performed using the argumentation reasoning. The use of the argumentation reasoning permits us to solve all the identify anomalies and conflicts, by ordering the rules. We introduce the used theoretical formalism in Section III-A. The conflicts/anomalies resolution is done by taking into account the network topology and the resolution strategies. Once the firewall rules are ordered, then *ArgoFiCo* decides how to populate the firewalls of the network by avoiding conflicts and anomalies. Finally, the rules are translated into firewall rules and the firewalls' configurations are generated.

A. Preference-Based Argumentation Framework

Our framework is based on preference-based argumentation [8], [10], which permits us to work with conflicting rules, given its *non-monotonic* nature. *ArgoFiCo* uses the Gorgias [8] tool, which is a preference-based argumentation reasoning tool that uses abduction [9]. We decided to use argumentation reasoning as the basic formalism for our framework given the extended use of argumentation in other security problems, e.g., policy analysis [3], secure data sharing in the cloud environment [11], [13], swarm of drones [7], [12] and forensics investigation [14], [15].

An *argumentation theory* is a pair $(\mathcal{T}, \mathcal{P})$ of argument rules $\mathcal{T}$ and preference rules $\mathcal{P}$. The argument rule describe the premises to reach a particular conclusion, while the preference rules describe the preferences between argument rules.

In our approach the rules of the preference-based reasoning are the high-level requirements and are denoted as follows:

$$req(allow/deny, source, destination, type \text{ of traffic})$$

where the premises of the rule are the $source$, $destination$, $type$ of traffic and information related to the location of the source and destination, and the conclusion is the decision of *allowing* or *denying* the traffic. The preferences between rules are the order between the rules. In particular, the priority rules permit us to order the rules and to have a total order for the set of rules. The preferences are introduced automatically by *ArgoFiCo*, by applying the conflict resolution strategies.

B. ArgoFiCo*'s Inputs, Outputs, and Configurations*

The *inputs* of our framework are: the network policies, which are given in the form of high-level requirements for the network; and the network topology together with the knowledge base. The *high-level network requirements* specify which communications are allowed and which are denied. We permit the user to specify these requirements using statements that are close to natural language and are user-friendly. The user provides information about the topology of the network, with its components (hosts and firewalls), their positions, *containing*[1] relations, and the routing tables. The *knowledge base* is information that permits to translate high-level requirements to a low-level network layer information. The network topology and knowledge base are needed by *ArgoFiCo* to correctly derive from the high-level requirements the low-level configuration for every firewall.

The *outputs* of *ArgoFico* are the firewalls' low-level configurations. These configurations are generated automatically and are composed of all the ordered rules that need to apply in each firewall.

The administrators can choose if they want to focus on the reachability aspects of the network and make it more robust by using the *max configuration*, or if they want to have a more

[1]A containing relation represents the relation of a system component being part of one or different networks/subnetworks/zones.

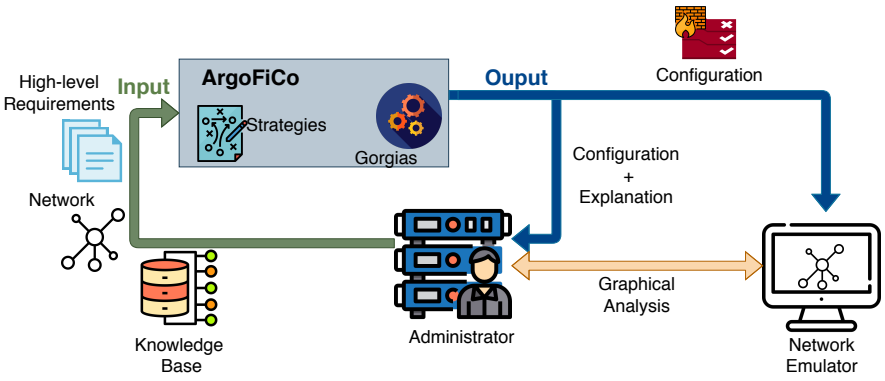

Fig. 1. An overview of our framework for Automatic Firewalls Configuration via Argumentation Reasoning (ArgoFiCo)

slim and fast routing/filtering by using the *min configuration*. We tested our framework using various realistic scenarios. *ArgoFico* always configured correctly the rules of the firewalls, for both max and min configurations. The given firewalls' configurations were tested using the network emulator that confirmed the correctness of the firewalls' rules.

C. Inter-Firewalls and Intra-Firewall Anomalies

ArgoFiCo uses *resolution strategies* that describe how the anomalies/conflicts should be solved in the policy specifications. In particular, *ArgoFiCo*'s conflicts/anomalies resolution applies different strategies for different types of anomalies between rules.

ArgoFiCo avoids the anomalies introduced in [1] for the anomalies between rules of the same firewall. *ArgoFiCo* is able to avoid the *shadowing*, *correlation*, *generalization*, *redundancy*, and *irrelevance* anomalies. The *shadowing* anomaly occurs when a rule is shadowed, in particular, a previous rule matches all its packets, and the shadowed rule is never activated. The *correlation* anomaly occurs when two rules with conflicting actions match some packets of each other. The *generalization* anomaly occurs when two rules with conflicting actions and all the packets matched from one of the rules are a subset of the packets of the second rule. The *redundancy* anomaly occurs when two rules with the same actions have matching or partially matching packets. The *irrelevance* anomaly occurs when a rule that does not match any traffic that might flow in that network.

ArgoFiCo is able to identify and avoid all the potential anomalies between rules of different firewalls (inter-firewall anomalies) introduced in [1]. *ArgoFiCo* avoids the *shadowing*, *spuriousness*, *redundancy*, and *correlation* inter-firewall anomalies. The *shadowing* anomaly between firewalls' rules occurs when an upstream firewall blocks the traffic accepted by a downstream firewall. The *spuriousness* anomaly between firewalls' rules occurs when an upstream firewall permits the traffic denied by a downstream firewall. The *redundancy* anomaly between firewalls' rules occurs when a downstream firewall blocks the traffic already blocked by an upstream firewall. The *correlation* anomaly between firewalls' rules occurs when two rules with conflicting actions and all the packets matched by one of the rules are a subset of the packets of the second one, where one rule is an upstream firewall and the other is a downstream firewall.

IV. Use Case for *ArgoFiCo*

Let us give an example of how *ArgoFiCo* works. Our case study is a cyclic network topology, shown in Figure 2. This network is composed of three different subnetworks, where each of them contains two to three hosts, with three firewalls connecting each subnetwork to the others. The high-level requirements provided by the administrator are given below:

1) $req_1(deny, subnet1, subnet2, all)$
2) $req_2(allow, bob, david, tcp)$
3) $req_3(allow, bob, subnet2, tcp)$
4) $req_4(deny, subnet1, subnet3, all)$
5) $req_5(deny, eve, subnet4, all)$
6) $req_6(allow, subnet1, subnet3, all)$

The source and destination names are already provided in the topology and the protocols are: tcp, udp or all.

The above requirements are analyzed by *ArgoFiCo* , which provides the following potential conflicts together with their resolution:

- Conflict 1 - redundancy: rule req_2 is included in rule req_3 and their actions are the same. Since req_2 is more specific and it does not exist any rule req_x with a different action, included in req_3 and that includes req_2, then req_2 is removed;
- Conflict 2 - shadowing: rule req_3 is included in rule req_1 but they have different actions. Since rule req_3 is more specific, it has a higher priority with respect to rule req_1, $(req_3 > req_1)$;
- Conflict 3 - irrelevance: rule req_5 contains as the source and destination hosts that are not in the topology, thus it is irrelevant and is removed;
- Conflict 4 - shadowing: req_6 is exactly matching req_4 but they have different actions. Thus, req_6 is removed.

The result of the resolution and ordering module is as below:

1) $req_3(allow, bob, subnet2, tcp)$
2) $req_1(deny, subnet1, subnet2, all)$
3) $req_4(deny, subnet1, subnet3, all)$

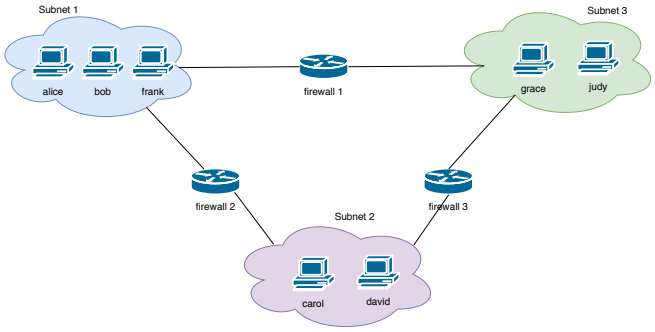

Fig. 2. Network topology for the Use Case

where req_2, req_5, and req_6 are removed.

Let us now assume that the max configuration is selected. The first step is to understand which are the paths for a certain packet to go from the source to the destination. For req_3 we have two paths: the first uses Fw_1[2] and Fw_3 and the second uses Fw_2; for req_1 we have two paths: the first uses Fw_1 and Fw_3 and the second uses Fw_2; and for req_4 we have two paths: the first uses Fw_1 and the second uses Fw_2 and Fw_3. req_3 should be applied in all paths, as it is an "*allow*" action, thus, it should be placed in all firewalls. For req_1 and req_4 because their action is "*deny*", the rules are put in the most upstream firewall along each path. The output of the distribution module, with max configuration, is as follows:

Fw_1

1) $req_3(allow, bob, subnet2, tcp)$
2) $req_1(deny, subnet1, subnet2, all)$
3) $req_4(deny, subnet1, subnet3, all)$

Fw_2

1) $req_3(allow, bob, subnet2, tcp)$
2) $req_1(deny, subnet1, subnet2, all)$
3) $req_4(deny, subnet1, subnet3, all)$

Fw_3

1) $req_3(allow, bob, subnet2, tcp)$

In case the min configuration is selected, the result is:

1) $req_3(allow, bob, subnet2, tcp)$

where req_1 and req_4 are removed, as there is no rule with a lower priority than them that "allows" a related requirement, and we expect to have the "deny all" rule at the end of the firewalls. The distribution module puts req_3 in Fw_2.

V. Conclusion

In this paper, we proposed a novel approach that permits to automatically configure distributed firewalls in a network. Our framework, called *ArgoFiCo*, is useful to network administrators, as the latter can provide high-level requirements for the network, together with the network topology and get as result, from *ArgoFiCo*, the configuration for each firewall of the network. The main goal of the tool is to help the administrator during the firewalls' configuration by avoiding human errors.

We plan to extend *ArgoFiCo* in order to be able to update the existing firewalls configuration. Furthermore, it is interesting to integrate the approach with reinforcement learning that permits to learn from the administrator the conflict resolution strategies and in the future to fully automate this process. As future work, we plan to use our tool in real commercial networks with a higher level of complexity.

[2]For the sake of simplicity we denote the firewalls with Fw.

Acknowledgments

Erisa Karafili was supported by the European Union's H2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 746667.

References

[1] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan. Conflict classification and analysis of distributed firewall policies. *IEEE J. on Selected Areas in Communications*, 23(10):2069–2084, 2005.
[2] S. Arunkumar, S. Pipes, C. Makaya, E. Bertino, E. Karafili, E. Lupu, and C. Williams. Next generation firewalls for dynamic coalitions. In *IEEE SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, pages 1–6, 2017.
[3] A. K. Bandara, A. Kakas, E. C. Lupu, and A. Russo. Using argumentation logic for firewall policy specification and analysis. In *Large Scale Management of Distributed Systems*, pages 185–196, 2006.
[4] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems*, 22(4):381–420, 2004.
[5] C. Basile, D. Canavese, A. Lioy, C. Pitscheider, and F. Valenza. Inter-function anomaly analysis for correct sdn/nfv deployment. *International Journal of Network Management*, 26(1):25–43, 2016.
[6] C. Basile, F. Valenza, A. Lioy, D. R. Lopez, and A. Pastor Perales. Adding support for automatic enforcement of security policies in nfv networks. *EEE/ACM Trans. Netw.*, 27(2):707–720, 2019.
[7] A. Cullen, E. Karafili, A. Pilgrim, C. Williams, and E. Lupu. Policy support for autonomous swarms of drones. In *ETAA@ESORICS 2018*, pages 56–70, 2018.
[8] A. Kakas and P. Moraitis. Argumentation based decision making for autonomous agents. In *AAMAS '03*, pages 883–890, 2003.
[9] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *J. Log. Comput.*, 2(6):719–770, 1992.
[10] A. C. Kakas, P. Mancarella, and P. M. Dung. The acceptability semantics for logic programs. In *ICLP*, pages 504–519, 1994.
[11] E. Karafili and E. Lupu. Enabling data sharing in contextual environments: Policy representation and analysis. In *SACMAT 2017*, pages 231–238, 2017.
[12] E. Karafili, E. Lupu, S. Arunkumar, and E. Bertino. Argumentation-based policy analysis for drone systems. In *IEEE SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, pages 1–6, 2017.
[13] E. Karafili, K. Spanaki, and E. Lupu. An argumentation reasoning approach for data processing. *Computers in Industry*, 94:52–61, 2018.
[14] E. Karafili, L. Wang, A. Kakas, and E. Lupu. Helping forensic analysts to attribute cyber-attacks: An argumentation-based reasoner. In *PRIMA 2018*, pages 510–518, 2018.
[15] E. Karafili, L. Wang, and E. Lupu. An argumentation-based approach to assist in the investigation and attribution of cyber-attacks. *CoRR*, abs/1904.13173, 2019.
[16] J.D. Moffett and M.S. Sloman. Policy hierarchies for distributed systems management. *IEEE Journal on Selected Areas in Communications*, 11(9):1404–1414, 1993.
[17] F. Valenza, C. Basile, D. Canavese, and A. Lioy. Classification and analysis of communication protection policy anomalies. *IEEE/ACM Trans. Netw.*, 25(5):2601–2614, 2017.
[18] F. Valenza, S. Spinoso, C. Basile, R. Sisto, and A. Lioy. A formal model of network policy analysis. In *RTSI*, pages 516–522, 2015.
[19] Verizon. Data Breach Investigations Report, 2019.
[20] P. Verma and A. Prakash. FACE: A Firewall Analysis and Configuration Engine. In *SAINT05*, pages 74–81, 2005.