A demonstration of VEREFOO: an automated framework for virtual firewall configuration

A demonstration of VEREFOO: an automated framework for virtual firewall configuration

Daniele Bringhenti, Riccardo Sisto, Fulvio Valenza
Dip. Automatica e Informatica, Politecnico di Torino, Torino, Italy, Emails: {first.last}@polito.it

***Abstract*—Nowadays, security automation exploits the agility characterizing network virtualization to replace the traditional error-prone human operations. This dynamism allows user-specified high-level intents to be rapidly refined into the concrete configuration rules which should be deployed on virtual security functions. In this revolutionary context, this paper proposes the demonstration of a novel security framework based on an optimized approach for the automatic orchestration of virtual distributed firewalls. The framework provides formal guarantees for the firewall configuration correctness and minimizes the size of the firewall allocation scheme and rule set. The framework produces rules that can be deployed on multiple types of real virtual function implementations, such as iptables, eBPF firewalls and Open vSwitch.**

***Index Terms*—network security, firewall, optimization**

I. Introduction

Security misconfiguration reportedly represents a serious problem for network management. According to the most recent Data Breach Investigations Report produced by Verizon[1], misconfiguration is the most critical cause of breaches for cyber security attacks, and it is mainly produced by the same system administrators who should prevent them. Human fallibility, increasing network size and complexity, function heterogeneity are only few of the reasons why traditional trial-and-error configuration approaches cannot work anymore. Manual operations are not compatible with the dynamism introduced by softwarization paradigms such as Network Function Virtualization, and cannot cope with the fast evolution cyberattacks are undergoing.

Automation may be the best solution to improve the security of the configuration of next-generation computer networks. In the literature, the application of intent-based approaches for security management has been recently gaining big traction, thanks to the contribution of EU Projects such as SECURED[2], ANASTACIA[3] and ASTRID[4]. After a human administrator specifies high-level intents describing the expected security behavior (e.g., to prevent future attacks or repair an existing security flow), these intents are transformed into the actual configuration rules and they are installed on the virtual functions without requiring further human interventions [1]. A security function that may particularly benefit from a similar strategy is the distributed packet filtering firewall, the most commonly used function for enforcing connectivity security requirements [2]. However, most of the related work has limitations that undermine its potentiality, e.g., [3] does not provide a cost-efficient solution for minimizing resource consumption, and [4] does not guarantee the correctness of the firewall configuration.

This paper presents the demonstration of a framework, called ***VE**rified **REF**inement and **O**ptimized **O**rchestration* (*VEREFOO*), which surpasses these limitations. This framework relies on a formal approach, initially shared in [5] and later finalized in [6], which combines automation, formal verification and optimization for the configuration of distributed firewalls. The objectives of this demonstration are to show how: (i) VEREFOO improves the user experience of security configuration throughout a user-friendly interface for intent specification and virtual network description; (ii) it computes the firewall allocation scheme and configuration in an optimized and fast way, compatibly with the dynamism of network virtualization; (iii) it can be easily used without the requirement of knowing the syntax of the softwarized firewall implementation (e.g., iptables, ipfirewall).

The remainder of this paper is structured as follows. Section II described the high-level architecture of VEREFOO. Section III details how the VEREFOO demonstration will be showcased. Section IV discusses future work.

II. Architecture

VEREFOO is an automated framework that can compute the firewall allocation scheme and configuration for a virtual network, providing assurance of their correctness and optimizing their size. As shown in Fig. 1, the high-level architecture of VEREFOO is organized in three main phases.

Network Description and Intent Specification: The human user specifies two inputs. The first input is the description of the virtual network topology where security must be enforced. The topology description is a graph representing the network functions, their interconnections and configurations (e.g., the addresses that must be changed by a network address translator). The second input is a set of security intents expressing connectivity requirements, i.e., specifying which traffic flows are allowed to reach their destination and which other ones must be blocked. The network and intent representations are expressed with a user-friendly language, which represents a link between humans and the automated

[1]https://www.verizon.com/business/resources/reports/dbir
[2]https://www.secured-fp7.eu/
[3]http://www.anastacia-h2020.eu/
[4]https://www.astrid-project.eu/

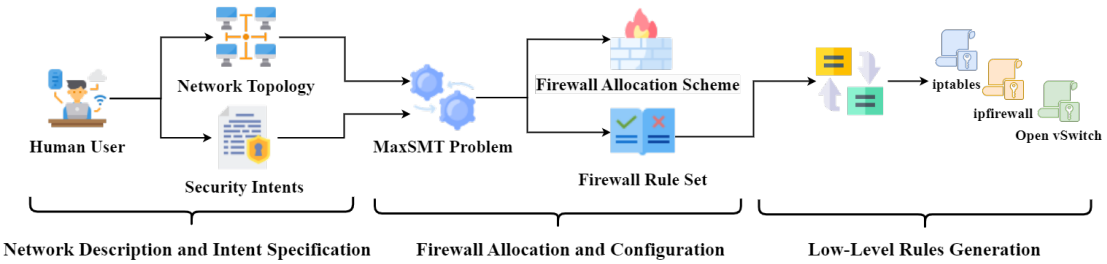


Fig. 1: VEREFOO Architecture

process. This phase is the only one that requires active human intervention.

Firewall Allocation and Configuration: After receiving the input description, VEREFOO employs it for the formulation of a constraint programming problem, called Maximum Satisfiability Modulo Theories (MaxSMT) problem. This formulation enables both a-priori formal assurance that the solution is correct and optimization [7]. After the problem resolution, the computed firewall allocation scheme and configuration are both guaranteed to satisfy the specified intents, and they are composed of the minimum number of needed firewall instances and rules. The representation of the output rules is characterized by a generic syntax, which abstracts the vendor-specific syntaxes of the different security solutions.

Low-Level Rules Generation: The output rules must be converted to the specific syntax of the real firewalls deployed in the network (e.g., iptables, Open vSwitch, eBPF-based firewall). Therefore, VEREFOO embeds a translator, which can automatically transform the vendor-independent representation into multiple syntaxes. Then, the user can install the configuration files produced by VEREFOO on the firewalls, or pass them to an orchestrator for their automated deployment.

This architecture has been designed for the automatic firewall configuration, but it is flexible enough to be extended to support other function types, e.g., VPN gateways [8] and SDN switches [9]. The problem of automating the configuration of firewalls and VPN gateways simultaneously has been also addressed in a paper accepted for presentation in SecSoft 2023, a workshop co-located with IEEE Netsoft 2023 [10].

III. Demonstration

VEREFOO is an open-source Java framework[5], developed as a Spring Boot application embedding the Apache Tomcat server. When running, VEREFOO can be accessed by a user in two ways: 1) by interacting with the REST APIs exposed by the framework interface (e.g., through REST clients); 2) by employing a user-friendly Graphical User Interface (GUI), developed with the Javascript-based React library.

This demonstration is focused on the firewall configuration and it is based on a specific version of VEREFOO, called Budapest[6]. The demonstration simply requires the presenter's

[5]https://github.com/netgroup-polito/verefoo.
[6]https://github.com/netgroup-polito/verefoo/tree/Budapest.

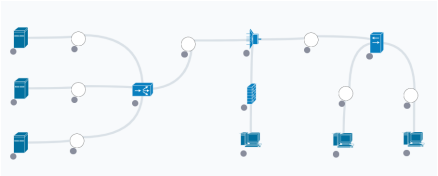


Fig. 2: Example of user-specified network topology

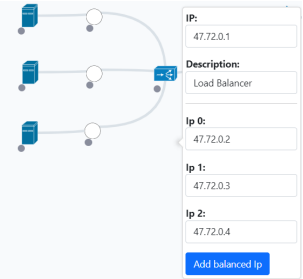


Fig. 3: Example of load balancer configuration

laptop, as every network component is softwarized. An external monitor, a keyboard and a mouse can be useful to improve the audience interaction with the framework.

The showcased operations correspond to the three phases of the VEREFOO architecture. Here, we summarize how each phase is planned to be demonstrated.

Network Description and Intent Specification: The user employs the GUI to specify both the network description and the security intents. The network topology can be intuitively designed in the main page of the GUI through a drag-and-drop mechanism (Fig. 2): the user can select the network function type, position it in the network, and create the links to other network elements. Clicking on each node of the network topology opens a box that allows the user to set configuration parameters (e.g., Fig. 3 shows that for a load balancer it is possible to specify the IP addresses of the servers for which traffic load balancing must be performed). Among the nodes that compose the network, the user can also select Allocation Places (APs), representing candidate positions for firewall allocation. Additionally, the user can define the security intents (Fig. 4), by specifying for each one the action that must be applied on certain traffic (deny or allow), and the IP 5-tuple values identifying that traffic (source and destination IP address, source and destination port, transport-level protocol).

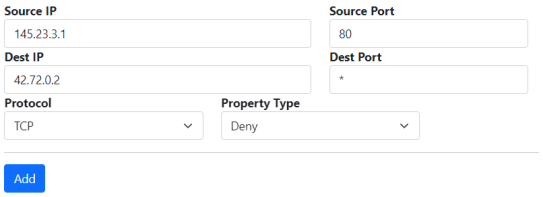


Fig. 4: Example of user-specified intent

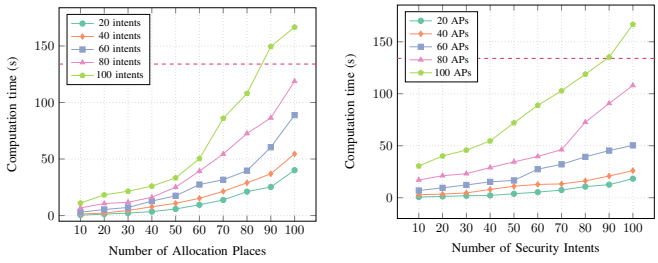
```
Vendor-independent firewall configuration:
<node name="192.168.56.6" functional_type="FIREWALL">
  <neighbour name="192.168.56.3"/>
  <neighbour name="192.168.57.4"/>
  <configuration name="conf3" description="b0">
    <firewall defaultAction="ALLOW">
          <elements>
            <action>DENY</action>
            <source>145.23.3.1</source>
            <destination>42.72.0.2</destination>
            <protocol>TCP</protocol>
            <src_port>80</src_port>
            <dst_port>*</dst_port>
      </elements>
        </firewall>
  </configuration>
</node>


iptables configuration:
#!/bin/sh
cmd="sudo iptables"
${cmd} -F
${cmd} -P INPUT ACCEPT
${cmd} -P FORWARD ACCEPT
${cmd} -P OUTPUT ACCEPT
${cmd} -A FORWARD -p tcp -s 145.23.3.1
       -d 42.72.0.2 --sport 80 -j DROP
```

Fig. 5: Example of output firewall configuration

All these pieces of information are locally stored as an XML file, so that the user can later reload it to continue working on a specific topology or set of security intents. Alternatively, the user may decide to directly write this XML file by themselves.

Firewall Allocation and Configuration: When the user has created all the inputs, the GUI or the REST Client sends an HTTP Request to VEREFOO, including them in its body with XML format. This allows the framework to validate the inputs against an XML schema, checking their syntax and preventing the framework from failures due to bad requests. After this initial check, VEREFOO internally builds the constraints composing the constraint programming problem modeling the firewall configuration problem. Then it employs a state-of-the-art MaxSMT solver by Microsoft Research, called Z3, to solve the MaxSMT problem. If a correct solution for the built problem cannot be found (e.g., a user-specified intent cannot be enforced in the corresponding topology), VEREFOO raises an error to the user. Instead, it generates an XML file describing the firewall allocation scheme and configuration, and sends it back in an HTTP Response. The GUI parses it and graphically shows the outputs, enabling the user to modify them for a next run of the framework.

Low-Level Rules Generation and Deployment: In case of positive outcome, VEREFOO also performs the conversion of the XML representation of the firewall configuration to the syntax of a specific firewall implementation selected by the user (e.g., Fig. 5 shows this translation for an iptables configuration). Then, the produced low-level configuration is pushed in firewall containers deployed in a virtual network orchestrated by Docker Compose. The user can thus test the correctness of the generated rules, by pinging selected



(a) Allocation Places (b) Security Intents

Fig. 6: Scalability of VEREFOO [6]

endpoints to assess if the communications are blocked or allowed.

All these operations are executed quickly, compatibly with the dynamism of network virtualization. Fig. 6 illustrates the scalability of VEREFOO with respect to the input APs and security intents, and it confirms that the framework can be efficiently applied to networks of considerable size.

IV. Conclusions and Future Work

This paper describes the behavior and demonstration of VEREFOO, where its React GUI is efficiently used to automate and optimize the configuration of virtual firewalls in a formally correct way. This demonstration also shows how VEREFOO hides the heterogeneity of firewall implementations by using user-friendly intent-based languages. Future work envisions applying VEREFOO for the configuration of other function types, so as to provide automation for a complete security configuration.

References

[1] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," *J. Netw. Syst. Manag.*, vol. 15, no. 4, pp. 447–480, 2007.

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

[3] D. Ranathunga, M. Roughan, P. Kernick, and N. Falkner, "The mathematical foundations for mapping policies to network devices," in *Proc. of the Inter. Joint Conf. on e-Business and Telecommunications*, 2016.

[4] P. Verma and A. Prakash, "FACE: A firewall analysis and configuration engine," in *Proc. of the IEEE/IPSJ Inter. Symp. on Applications and the Internet*, 2005.

[5] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "Automated optimal firewall orchestration and configuration in virtualized networks," in *Proc. of the IEEE/IFIP Network Operations and Management Symp.*, 2020.

[6] ——, "Automated firewall configuration in virtual networks," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 2, pp. 1559–1576, 2023.

[7] D. Bringhenti, G. Marchetto, R. Sisto, S. Spinoso, F. Valenza, and J. Yusupov, "Improving the formal verification of reachability policies in virtualized networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18 (1), 2021.

[8] D. Bringhenti, G. Marchetto, R. Sisto, and F. Valenza, "Short paper: Automatic configuration for an optimal channel protection in virtualized networks," in *Proc. of the ACM Work. on Cyber-Security Arms Race*, 2020.

[9] D. Bringhenti, J. Yusupov, A. M. Zarca, F. Valenza, R. Sisto, J. B. Bernabé, and A. F. Skarmeta, "Automatic, verifiable and optimized policy-based security enforcement for sdn-aware iot networks," *Comput. Networks*, vol. 213, p. 109123, 2022.

[10] D. Bringhenti, R. Sisto, and F. Valenza, "Automating the configuration of firewalls and channel protection systems in virtual networks," in *Proc. of 9th IEEE International Conference on Network Softwarization*, 2023.