GreenShield: Optimizing Firewall Configuration for Sustainable Networks

GreenShield: optimizing firewall configuration for sustainable networks

Daniele Bringhenti, Fulvio Valenza

***Abstract*—Sustainability is an increasingly critical design feature for modern computer networks. However, green objectives related to energy savings are affected by the application of approximate cybersecurity management techniques. In particular, their impact is evident in distributed firewall configuration, where traditional manual approaches create redundant architectures, leading to avoidable power consumption. This issue has not been addressed by the approaches proposed in literature to automate firewall configuration so far, because their optimization is not focused on network sustainability. Therefore, this paper presents GreenShield as a possible solution that combines security and green-oriented optimization for firewall configuration. Specifically, GreenShield minimizes the power consumption related to firewalls activated in the network while ensuring that the security requested by the network administrator is guaranteed, and the one due to traffic processing by making firewalls to block undesired traffic as near as possible to the sources. The framework implementing GreenShield has undergone experimental tests to assess the provided optimization and its scalability performance.**

***Index Terms*—firewall, network sustainability, power consumption**

I. Introduction

In recent years, the sustainability of computer networks has progressively become a design feature that cannot be neglected. Specifically, energy-efficient network management has become crucial for a two-fold motivation: economic and social [1]. On the one hand, the size of modern networks is getting so big that network providers have to minimize power consumption in order to reduce operating expenditures. On the other hand, nowadays, global awareness about "green" themes has become pervasive in all sectors, and therefore also network providers must rightly meet the expectations of their current and potentially future customers by implementing energy-oriented strategies.

A primary challenge that arises in sustainable networks is the management of the impact that cybersecurity has on power consumption. Providing adequate countermeasures to a high number of attack classes is undeniably essential in next-generation networks, where attacks have shorter exploitation times, are based on highly mutable vectors, and may come from different simultaneous sources. Defense in depth is a principle related to security by design, which is often adopted to safeguard network assets from those attacks. Its main idea is that different, complementary devices are used as multiple defense lines [2]. However, in many cases, administrators limit themselves to placing devices of the same type and with duplicated configuration, thinking that would be enough. However, not only do such architectures not reach the objectives of defense in depth, but they also introduce a redundancy that is in sharp contrast with green-oriented network management strategies because each additionally deployed function contributes to the overall network power consumption.

A network security function for which this issue is increasingly perceived is the distributed packet filtering firewall, which represents the most commonly used essential defense mechanism to oppose the most common cyberattacks. The behavior of a distributed firewall impacts the network green economy with two contributions. First, by itself, each firewall composing a distributed filtering architecture is associated with a different average power consumption, determined by the simple reason that the function has been activated. Second, this power consumption can get higher when the firewall must process a larger amount of traffic.

Such redundancy is mainly due to the the fact that the traditional approaches for firewall configuration are still manual, approximate, and trial-and-error. Within the big size of modern computer networks, human administrators already struggle to find a configuration that is at least correct, as they commonly introduce configuration anomalies that alter the expected firewall behavior [3]. Moreover, they often limit themselves to create defense lines with identical configurations, failing in achieving defense in depth, and it is difficult to expect from them a configuration that is also optimized. To address these shortcomings, in literature few studies have investigated approaches relying on network security automation for firewall configuration [4]. These approaches have often been successful in avoiding human errors. Sometimes, they have also contributed to reaching optimized firewall configurations, e.g., where firewall and rule numbers are minimized. However, albeit significant, this optimization does not represent a real turning point for sustainable networks, because it fails to achieve two essential green objectives. On the one hand, a fine-grained firewall energy assessment is lacking. In a network, each firewall implementation is characterized by different power consumption (e.g., depending on the additional features it can provide), which must be taken into account to optimize network-wide energy efficiency. On the other hand, intermediate network nodes should process the least amount of traffic as possible to avoid reaching peak power consumption levels, and this requires allocating firewalls as near as possible to the sources of the communications. State-of-the-art proposals do not reach these goals, because they treat all firewalls in the same way, and often put them in central

Daniele Bringhenti and Fulvio Valenza are with the Politecnico di Torino, Dip. Automatica e Informatica; e-mail: {first.last}@polito.it.

positions of the network, thus distant from the traffic sources.

In view of all these considerations, this paper proposes a novel methodology named GreenShield, which provides sustainable security through an automatic configuration of distributed packet filtering firewalls with attention to network sustainability. This methodology achieves automation by following a principle named policy-based management, where human administrators simply specify their security desires related to blocked or allowed communications as sentences named policies, which are expressed with a user-friendly language, and later are refined into the concrete network security configuration. Besides, unlike most related work, GreenShield combines automation with two other features, i.e., optimization and formal verification. Specifically, GreenShield achieves the two aforementioned energy-oriented goals: i) it activates the firewalls of the distributed architecture in a way to minimize the overall average power consumption; ii) it configures the firewalls so as to block communications as closely as possible to their sources, thus reducing the number of traffic flows processed by each network middlebox. As it can be inferred by these goals, from an operational point of view, the savings achieved by GreenShield are related to the power consumption during network operation (i.e., in an already deployed network), and therefore with a constant environmental impact at the equipment production level, as the machines are assumed to be already included in the network topology. Moreover, GreenShield provides formal assurance that the computed firewall configuration is correct and compliant with the requests of human administrators, thus boosting confidence in using this security automation approach. These two features are embedded in the proposed methodology by formulating the configuration problem as a Maximum Satisfiability Modulo Theories (MaxSMT) problem based on constraint programming.

The definition and implementation of GreenShield stems from VEREFOO (VErified REFinement and Optimized Orchestration) [5]. The idea has been to avoid starting to work on firewall automation from scratch, in view of the richness of the related literature. VEREFOO was selected as starting point of this study because it is the most feature-complete approach for firewall configuration existing in literature to date, and the code of its implementation is available as open source. However, VEREFOO was designed for virtualized networks and completely neglects the requirements of sustainable networks. Therefore, GreenShield goes beyond that approach, proposing new formal models that capture the way firewalls consume power in a network, and introducing new optimization objectives related to energy savings.

The remainder of this paper is structured as follows. Section II discusses the related work, highlighting its limitations in addressing the problem of power consumption in firewall configuration. Section III describes the GreenShield approach from a high-level perspective. Section IV presents the formal models of the network and security policies. Section V formalizes the MaxSMT problem on top of those models. Section VI describes the implementation developed for GreenShield and shows the results of its experimental validation. Finally, Section VII draws conclusions and prospects future work.

II. Related Work

This section dissects the related work, focusing on three classes of studies: policy-based approaches for sustainable networks (Subsection II-A), green-aware approaches for network security (Subsection II-B), and automated approaches for firewall configuration (Subsection II-C). After highlighting their limitations, it discusses the contributions to the state of the art introduced by the proposal of this paper (Subsection II-D).

A. Policy-based approaches for sustainable networks

In response to the increasing relevance of green networking since the beginning of the previous decade [6], policy-based management may represent an effective solution to address network sustainability problems. Nevertheless, according to an exhaustive survey about policy-based approaches for sustainable networks [7], sustainability has been rarely taken into account in the design of policy refinement, i.e., the process in charge of moving a high-level requirement representation into the concrete device configuration.

The only study where sustainability is explicitly addressed is the one described in [8]. There, a methodological approach for sustainability-oriented policy refinement is illustrated, working through five different intermediate policy representation levels: business view (e.g., service level agreements), system view (e.g., sustainability and performance indicators), network view (e.g., metrics for network operations related to its technology), device view (e.g., metrics for device operation), and instance view (e.g., management information bases). The proposed refinement is a rule-based transformation because it works on predefined rules, thus allowing a more precise domain specificity. At the same time, the general idea presented in [8] is not concretely applied to and validated with network security functions such as firewalls.

Other refinement methods could theoretically support sustainability parameters, but they do not explicitly model them. The most relevant examples are [9] and [10]. The former is a case-based reasoning approach, where the policies are derived from the output of a decision tree classification algorithm applied to an existing system configuration, and they are later refined taking into account the distribution statistics of relevant system attributes. The latter is a logic-based approach, as it uses a variant of Event Calculus for the problem representation, and it works on so-called decomposition rules representing how actions and objects described at a high level relate to those at a lower level. From a theoretical point of view, both approaches may be general enough to allow extensions for the interpretation of policies related to network sustainability. However, in practice, such extensions, especially the ones related to network security policies, have not been proposed in literature, so their applicability to network security problems is not guaranteed. Besides, their generality would still represent an obstacle to successfully achieving more specific green optimization goals, such as positioning a firewall as nearest as possible to the source of the traffic flows to be blocked.

B. Green-aware approaches for network security

The literature has also started to investigate the impact of network security on energy consumption. This problem has been deemed relevant for networks of mobile systems, characterized by a power-constrained nature [11], but also for traditional computer networks, which are nowadays subject to an increasing number of cyber attacks, requiring energy-consuming security mechanisms as defense [12].

In this research field, several analyses have been carried out to assess the degree of this impact and to evaluate possible trade-offs between energy saving and security. The study described in [13] investigates the impact of identifying and discarding malicious packets in an aggressive way, by moving the Intrusion Prevention System (IPS) analysis from network fringes to the central routers. The experimental results of this investigation show that, through that shift of paradigm for the IPS analysis, the total energy consumption of the network is reduced because fewer devices must route packets, and at the same time, the routing delays are not significantly larger than the ones occurring when the analysis is performed at the destination. Instead, in [14], the energy-aware optimal placement of Virtual Network Functions (VNFs) is investigated for softwarized networks. This analysis takes into account the fact that different additional power consumption factors may occur depending on the network server and slice where each VNF is placed, and aims to minimize the overall consumption while satisfying security constraints related to vulnerabilities that the position of multiple VNFs in the same server may create. Then, [15] analyzes already existing energy optimization techniques and load balancing techniques for multi-class firewall rules as possible as possible solutions for resource-limited wireless networks, to assess how they differ in terms of computation load.

Even if all these studies analyze combinations between security and energy saving, they address different problems with respect to our proposal (e.g., IPS analysis, VNF placement, load balancing). Instead, our approach aims to reach green-oriented objectives in the problem of distributed firewall configuration, i.e., a problem that has not yet been addressed in literature, as we will discuss in the remainder of this section.

C. Automated approaches for firewall configuration

Firewall configuration automation has been extensively investigated in literature, as firewalls are the most commonly used security function type in computer networks. It is also a complex problem, because configuring a firewall requires solving two sub-problems: defining where firewalls should be allocated (or which ones should be activated, if already placed in the network) and computing their filtering rules. With only two exceptions, represented by the approaches discussed in [5], [16], all state-of-the-art proposals focus on addressing only one of these two sub-problems.

For what concerns firewall allocation and activation, it represents a specification of the more general network (security) service chaining or composition problem [17], from which the firewall-oriented research line stemmed. In this area, two relevant studies are [18], [19]. However, the approaches described there have limitations independent of the lack of power consumption optimization. In fact, they can add firewalls to a network that only has endpoints and routers, neglecting more complex middleboxes such as NATs, and they cannot provide their filtering rules. Besides, [18] does not use formal verification techniques.

For what concerns filtering rule computation, this second research line was initiated by the study in [20], which, albeit limited to centralized firewalls, posed the problem of automating their computation for the first time. Then, it has continued over the years up to now, initially with some simple extensions, i.e., with [21] [22], of that original proposal, and then through a series of alternative proposals that aimed to address the problem for distributed firewalls, in different scenarios (e.g., virtual networks). Some of the most recent studies, i.e., [23]–[26], also introduce formal verification as a means to provide assurance about the produced configuration. However, all of them still have intrinsic limitations. The methodologies illustrated in [23], [24] can only refactor existing firewall rule sets. On the contrary, the strategies proposed in [25], [26] limit themselves to producing correct firewall rules without pursuing any optimization goal (e.g., rule minimization).

As previously mentioned, the only two studies that address these two sub-problems simultaneously are [16] [5], which consequently represent the most advanced ones in this literature area. [16] have more limitations because it can only synthesize firewall chains, but it cannot allocate them in a network graph. Instead, [5] proposes a full-fledged approach, named VEREFOO, that combines automation, formal verification, and optimization to address the firewall configuration problem in ramified network topologies.

Nevertheless, all the studies mentioned here, including the state-of-the-art VEREFOO, are not designed to work in sustainable networks. Therefore, their output does not contribute to minimizing power consumption.

D. Our contributions

In view of the limitations of the previously discussed studies, the main contributions of this paper to the related literature are the following:

- To the best of our knowledge, GreenShield is the first approach in literature to automate distributed firewall configuration while pursuing green objectives. It thus represents a bridge between the studies on policy-based management for sustainable networks, which does not focus on network security functions, and the studies on automatic firewall configuration, which overlook its impact on power consumption.
- Differently from the models proposed in the studies on policy-based management, most of which were not designed explicitly for sustainability albeit theoretically supported, in GreenShield new models for policy refinement have specifically been defined, so as to achieve concrete green objectives.
- Differently from most related work on firewall configuration, GreenShield combines automation with formal

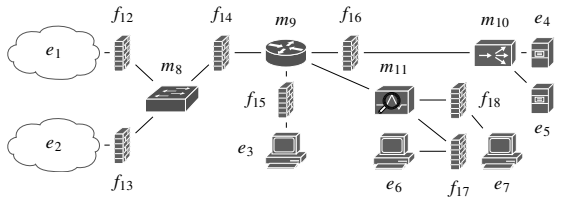


Fig. 1: Example of input Network Graph

TABLE I: IP addresses and roles of the end points

Identifier	IP address	End point role
e_1	230.40.2.0	Sub-network
e_2	230.40.3.0	Sub-network
e_3	151.11.0.3	Client
e_4	84.0.41.1	Server
e_5	84.0.41.2	Server
e_6	145.132.6.2	Client
e_7	145.132.0.3	Client

verification and optimization to address both configuration sub-problems. This combination was only previously achieved by VEREFOO [5], but without any optimization related to power consumption, as the optimality goals of that approach simply consist in the minimization of the number of firewall instances and of their filtering rules. Instead, unlike VEREFOO, GreenShield introduces new formal models related to the way real firewall implementations consume power when active. It also formalizes two new optimization objectives (i.e., power consumption minimization, firewall activation near communication sources).

III. Approach

This section provides a high-level overview of GreenShield, allowing the readers to understand how the proposed methodology has been designed and works before delving into the problem formalization. In particular, the GreenShield approach is composed of three main phases:

1) the human administrator prepares the two inputs of the approach (Subsection III-A);
2) the firewall configuration problem is modeled as a MaxSMT problem (Subsection III-B);
3) the formally correct and optimized output is automatically produced (Subsection III-C).

A. Inputs of the approach

The input definition is the only manual activity that is requested for the user. In our vision, this does not impact the automation of all next operations, because this task is merely descriptive and does not require computational complex activities. More specifically, GreenShield requires the human administrator to specify two inputs: a network graph and a set of network security policies.

TABLE II: Function types of the middleboxes

Identifier	Function type
m_8	network address translator
m_9	router
m_{10}	load balancer
m_{11}	traffic monitor

TABLE III: Power consumption of the firewalls

Identifier	Power consumption (W)
f_{12}	2330
f_{13}	2330
f_{14}	6100
f_{15}	3550
f_{16}	5270
f_{17}	6100
f_{18}	2330

1) Network Graph: The first input is a Network Graph (NG) representing the computer network where the distributed firewall must be automatically configured in a green-oriented way. A possible example is depicted in Fig. 1.

An NG should provide information about the way network functions are interconnected to each other, and their associated configuration, if it already exists. Besides, the nodes composing the NG can be associated with three main categories, which are described in the following and vary depending on the functionality they exercise in the NG.

- Nodes of Fig. 1 listed in TABLE I are *end points* because they are either the source or destination for the traffic flows crossing the network. An end point may be a single network host (e.g., a web server) or a sub-network representing multiple hosts (e.g., a corporate network to which the IP address range 230.40.2.0/24 is associated).
- Nodes of Fig. 1 listed in TABLE II, providing service functionalities such as network address translation, load balancing, or traffic inspection. Assumptions about elements belonging to this category are that they cannot execute firewalling functionalities and that they are not the source or destination of traffic flows. The administrator must provide a description of these nodes and their configuration. For example, GreenShield requires to know how a network address translator located in the input NG translates the IP addresses of the received packets.
- Nodes of Fig. 1 listed in TABLE III are *firewall* instances of the distributed firewalling architecture. Each firewall is initially in an *inactive* status, because it has been only installed in the network but it is currently turned off. For the sake of simplicity, it is possible to assume that all traffic flows cross inactive firewalls without being ever blocked as if they were crossing simple connection links. All these inactive firewalls are not characterized by any filtering configuration, as their computation is a specific task of GreenShield. Instead, each one is associated with a weight, representing its average power consumption. This design choice is motivated by the fact that, in this study, the power consumption optimization is carried

TABLE IV: Example set of Network Security Policies

Action	IPSrc	IPDst	pSrc	pDst	tProto
Allow	230.40.2.∗	230.40.3.∗	∗	∗	∗
Allow	230.40.3.∗	230.40.2.∗	∗	∗	∗
Allow	230.40.2.∗	84.0.41.∗	∗	80	∗
Allow	84.0.41.∗	230.40.2.∗	∗	∗	∗
Allow	145.132.6.2	145.132.0.3	∗	22	∗
Allow	145.132.0.3	145.132.6.2	∗	∗	∗
Deny	230.40.2.∗	84.0.41.∗	∗	≠80	∗
Deny	230.40.2.∗	145.132.6.2	∗	∗	∗
Deny	230.40.3.∗	145.132.0.3	∗	∗	∗
Deny	151.11.0.3	84.0.41.∗	∗	∗	∗
Deny	151.11.0.3	145.132.6.2	∗	∗	∗
Deny	151.11.0.3	145.132.0.3	∗	∗	∗
Deny	145.132.6.2	145.132.0.3	∗	≠22	∗
Deny	145.132.0.3	84.0.41.∗	∗	∗	∗
Deny	84.0.41.∗	145.132.0.3	∗	∗	∗

out offline, i.e., when the firewalls are not yet active and communications are not yet crossing the computer network. Under these circumstances, it is not possible to provide an estimate of the traffic load, which could be done only with online algorithms, and whose impact will be taken into account in our model in a different way, as it will be explained in Section V.

The weights associated to the firewalls may be different, because each firewall implementation has different energy requirements, as it can be easily seen in their data sheets. Considering two examples of the same vendor, Fortinet, the average power consumption is 2330 W for the 7060E-8 model, whereas it is more than double, 6100 W, for the FG-7081F model. Even if all considered firewalls have the same filtering capabilities (i.e., they are packet filters that analyze IP 5-tuples), they have different power consumption because they may have additional different capabilities (e.g., a firewall may consume more because it also works as an intrusion detection system, or it filters traffic with higher speed). Moreover, as firewalls that consume less power are commonly more expensive, administrators may have only a limited subset of them available in their network. Besides, when providing information about firewalls, the administrator can also specify the requirements of forcing the status of some of these firewalls to *active*. This request may be advanced when, for example, the administrator strictly wants some firewalls to be used in the final configuration. Clearly, this decision may lead to a less globally optimized solution, as some firewalls cannot be left turned off because of this requirement, and therefore the solution space is reduced.

2) Network Security Policies: The second input is a set of Network Security Policies (NSPs), describing which traffic flows must be discarded because potentially malicious, and which other ones must be able to reach their destination to guarantee the availability of some end-to-end communication services. A possible example is depicted in TABLE IV.

Each NSP is characterized by an action and a condition. The action specifies how the firewalling architecture must manage packets satisfying the condition, and it also discriminates NSPs into two classes. In particular, an *isolation NSP* is characterized by a *deny* action, while a *reachability NSP* is characterized by an *allow* action. Instead, the condition is used to identify the packets to which the action must be applied. As this study deals with packet filtering, the condition specifies the IP 5-tuple of the prohibited or allowed flows. As possible (partial) values of the IP 5-tuple fields, the ∗ symbol can be used to specify value aggregation. For example, the value 230.40.2.∗ used as source or destination IP address of the policy condition represents the address range 10.22.34.0/24. Instead, the value ∗ used as source or destination ports express all possible numbers that field may have, i.e., from 0 to 65535.

B. Firewall configuration problem formalization

After receiving these inputs from the administrator, GreenShield uses them as the basis for defining a partial weighted MaxSMT problem. This formalization is a constraint satisfaction problem, composed of first-order constraints stating which relations should hold among some decision variables, and its resolution consists in determining if it is possible to satisfy all those constraints simultaneously. In greater detail, a partial weighted MaxSMT problem is a generalization of the traditional SMT problem. The *partiality* of this formulation derives from the differentiation of two kinds of constraints: hard and soft. Hard constraints must always be satisfied to have a correct solution, whereas soft constraints are relaxable because their satisfaction is not mandatory. Instead, its *weighted* nature consists in the fact that each soft constraint is associated with a weight, and the goal is to maximize the sum of the weights assigned to the satisfied soft constraints. From here on, for the sake of conciseness, the term MaxSMT will be used to refer to partial weighted MaxSMT.

This formulation allows GreenShield to achieve both the features that we wanted to introduce in this automatic methodology, i.e., formal verification and optimization. For what concerns formal verification, if a solution can be found for a MaxSMT problem, the MaxSMT formulation guarantees that all hard constraints are satisfied in that solution, which is formally proved correct a-priori without the need of applying time-consuming a-posteriori formal verification techniques such as theorem proving or model checking. However, this correctness-by-construction assurance is implicitly provided as long as basic components of the constraints (i.e., the input of the approach) are formally represented so that their formal models adhere to the characteristics of their real counterparts and capture all the information that may influence the correctness of the solution. For what concerns optimization, it is provided by the partial weighted nature of the adopted formulation. As the soft constraints do not require strict satisfaction, they are suitable for representing optimality objectives. In this way, if some of them cannot be satisfied with a solution, that simply means that not all optimization objectives can be achieved, but correctness is still guaranteed.

Given these theoretical premises, in GreenShield both the inputs (i.e., the NG description and the NSRs) are formally modeled. Here, a primary challenge has been to define models representing a good trade-off between expressiveness and complexity, so as to maintain correctness by construction while avoiding an excessive impact on the performance. For

example, for the behavior of the middleboxes in the NG, only the forwarding and transformation behaviors have been modeled, as other aspects are not relevant for the enforcement of isolation and reachability policies. Then, starting from these models, first-order logic constraints are generated. On the one hand, hard constraints are used to check for the satisfaction of all the input NSPs. For each NSP, a hard constraint is introduced to state that the NSP must be satisfied in the NG where some firewalls may be activated. At the same time, also for each middlebox in the NG, a set of hard constraints is introduced to constrain how it can forward flows, because this behavior may impact the NSP satisfaction (e.g., a traffic flow may never be able to reach a destination, thus preventing the satisfaction of a corresponding reachability NSP). On the other hand, soft constraints are used to express the two optimization objectives of GreenShield: the minimization of the average power consumption of active firewalls and the blocking of traffic flows that match the condition of isolation NSPs as nearest as possible to their sources.

In the definition of all these constraints, some predicates are left respectively free (i.e., they are not bound to specific values), because solving the firewall configuration problem actually consists in finding a possible assignment for them. Examples of free predicates are those modeling the activation decision of each firewall for which the human administrator has not imposed a strict requirement, and those representing the configuration decisions about the filtering rules they should enforce.

C. Automatic output computation

After the MaxSMT problem has been formulated, GreenShield employs an automated solver to search for the existence of a correct solution and, if so, to produce the solution that optimizes the green optimization goals as much as possible. Even if the worst-case computational complexity of a MaxSMT problem is NP-complete, many MaxSMT instances can be solved in polynomial time on average using state-of-the-art solvers, thanks to algorithms and strategies that have been included in them to reach the best performance. The proposed MaxSMT problem formulation is also independent of the specific solver used for its resolution, so any solver that adheres to the semantics of MaxSMT problems can be employed without altering the problem formulation or producing different solutions.

When called by GreenShield, the solver always provides an answer in a decidable way, i.e., if a solution satisfying all hard constraints does not exist, the solver informs GreenShield (and its user) of the problem unsolvability, otherwise it successfully finds the optimal solution. The decidability nature of the proposed MaxSMT formulation is motivated by the fact that we used only a limited subset of theories to create it (i.e., the Boolean and integer theories, including only relational operators, without quantifiers). We were able to avoid more complex integer theories, like the Peano Arithmetic theory, which includes the multiplication operation but would make the MaxSMT problem undecidable, as they were not necessary to model the inputs of this methodology.

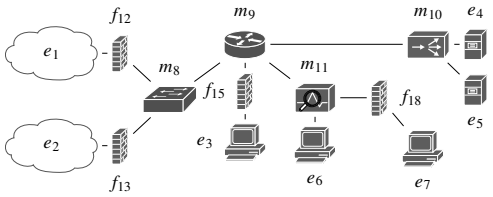


Fig. 2: Example of output firewall activation scheme

TABLE V: Example set of output filtering rules for f_{12}

#	Action	IPSrc	IPDst	pSrc	pDst	tProto
1	Allow	220.124.30.1	130.10.0.4	*	80	TCP
2	Allow	40.40.41.*	130.10.0.4	*	80	TCP
3	Allow	130.10.0.4	*.*.*.*	*	*	*
D	Deny	*.*.*.*	*.*.*.*	*	*	*

If GreenShield informs the network administrator that the solver could not find any solution satisfying all the hard constraints, this means that the firewalls allocated in the network are not enough to satisfy all NSPs, even if all of them were activated. For example, there may be no firewall in the path crossed by a flow that must be blocked on behalf of an isolation NSP. Subsequently, the administrator can use this information about the unsolvability of the problem to modify the inputs. Specifically, the administrator can introduce new firewalls in multiple places of the NG, as long as each one of these newly introduced firewalls is located in a position where the counterpart real implementation may be physically installable. As multiple new firewalls can be added to the NG after a failed run of GreenShield, the solution space is bigger, and the chance of finding a correct solution is higher. In this way, if it is found in a subsequent re-run of the methodology, among the newly introduced firewalls, only the activated one must be really positioned in the physical network.

Instead, if at least a correct solution satisfying the hard constraints exists, the MaxSMT solver provides GreenShield with the predicate and variable assignment corresponding to the solution that maximizes the sum of the weights associated with the satisfied soft constraints. From this information, GreenShield automatically extracts the two outputs of the firewall configuration problem. The first output consists in the activation scheme, i.e., the decision about which firewalls must be activated. The second output is represented by the filtering rules that each firewall must be configured with, in order to block or allow traffic flows as requested by the administrator through the NSPs. Replacing a non-activated firewall with network wires and installing the automatically computed filtering rules on activated firewalls are operations that should be performed manually by the administrator, after the conclusion of the GreenShield execution. Keeping these operations manual is not restrictive because they are simple tasks that do not involve any possible decision-making error. After all, the real objective of GreenShield is to automate the computation of the activation scheme and configuration, a very complex operation that would be error-prone, optimized, and time-consuming if performed manually.

Considering the exemplifying inputs shown in Fig. 1 and

TABLE IV, the output activation scheme would be the one illustrated in Fig. 2. The established distributed scheme is composed of four firewalls: f_{12}, f_{13}, f_{15}, f_{18}. For each one, the corresponding output filtering rule set is also produced. Here, for the sake of conciseness, only the filtering rule set of f_{12} is reported in TABLE V as a representative example. This solution has been chosen by GreenShield because it is the one that achieves the two green-oriented optimization objectives the most. For instance, in the computed activation scheme, both firewalls f_{12} and f_{13} have been selected, instead of having a single firewall f_{14}, which would have been enough to enforce correctly all the input NSPs listed in TABLE IV. This choice made by GreenShield is based on a dual motivation. First, the power consumption of f_{14} (which may assumed to be a Fortinet FG-7081F) is higher than the sum of the power consumptions related to f_{12} and f_{13} (which may be assumed to be two Fortinet FG-7060E). Second, f_{12} and f_{13} are nearer to traffic sources e_1 and e_2 than f_{14}, and also than another candidate f_{16}. Therefore, they can avoid a higher number of middleboxes processing traffic that is bound to be filtered before reaching the destination.

IV. Model

This section describes the formal models for the following elements of the firewall auto-configuration problem: the computer network (Subsection IV-A), network traffic and function behavior (Subsection IV-B), traffic flows (Subsection IV-C), security policies (Subsection IV-D), and firewall configuration (Subsection IV-E). These models will be later used to build the MaxSMT problem.

A. Network Graph model

The NG is modeled as a directed graph $G = (N, L)$, where N is the set of network nodes, and L is the set of directed connections among them. Each $n \in N$ is identified by a unique non-negative integer number through an association represented by the $index^N : N \rightarrow \mathbb{N}_0$ function. It is also assigned with a single IP address (e.g., 124.12.0.1) or an IP address range if the node is a subnetwork (e.g., 10.22.34.∗, standing for 10.22.34.0/24). Mapping each node to the set of its IP addresses is modeled as the $address : N \rightarrow 2^I$ function. Instead, each $l \in L$ is simply identified by the two non-negative integers identifying the extremities of l, and this association is represented by the $index^L$: $L \rightarrow \mathbb{N}_0^2$ function.

N is modeled as a union of multiple subsets, where each subset includes nodes with a different role for the purposes of the MaxSMT problem formulation: $N = N_E \dot{\cup} N_M \dot{\cup} N_F$. N_E is the set of all the end points, which may be the source or destination of traffic flows that the network administrator may want to allow or block. N_M is the set of all the middleboxes, which are not flow sources or destinations, but are in the middle of the network and can provide service functionalities such as network address translation, load balancing, and traffic monitoring. N_F is the set of firewalls that are allocated in the network. Each firewall $n \in N_F$ is associated with a weight w_n, representing the average power consumption when n is active (i.e., it has been powered up).

B. Traffic and network function behavior models

Each $n \in N$ shows a specific behavior toward the received packets. Packets sharing the same characteristics in terms of values characterizing their IP 5-tuple fields are grouped in packet classes, thus avoiding the definition of excessively specific per-packet models.

A packet class t, also named traffic in this paper, is modeled as a disjunction of predicates $q_{t,1} \vee q_{t,2} \vee ... \vee q_{t,n_t}$, where each $q_{t,i}$ is a sub-predicate defined over the IP 5-tuple fields. A packet belongs to a traffic t if its IP 5-tuple fields have values satisfying at least one of the sub-predicates modeling that class t. To provide this, each predicate $q_{t,i}$ is modeled as conjunction of five predicates, one defined over a specific field of the IP 5-tuple. For sake of simplicity, each $q_{t,i}$ is written as a tuple $q_{t,i} = (IPSrc, IPDst, pSrc, pDst, tProto)$. Each one of these predicates may define a grouping condition that is specific (e.g., a single IP address such as 124.51.40.2 or a single TCP port such as 80) or more generic (e.g., an IP address range 192.168.0.0/24, or a port range [100, 140]).

The behavior of a network function has been modeled by considering how it manages a traffic, i.e., a packet class. For the purposes of automatic firewall configuration, it is enough to model two components of network function behavior: the forwarding behavior and the transformation behavior. The forwarding behavior specifies if a traffic is blocked by an intermediate node while traveling toward its destination. Instead, the transformation behavior expresses how a packet class is modified by a network function. From the modeling point of view, the forwarding behavior of a network function $n \in N$ is represented by the *deny*: $N \times T \rightarrow \mathbb{B}$ predicate, which maps a node n and a packet class t to true if n drops all the packets identified by the predicate t, to false otherwise. On the contrary, the transformation behavior of a network function $n \in N$ is modeled by the *transform*: $N \times T \rightarrow T$ function, which maps a node n and an input packet class t to the packet class that may be produced by n as output.

C. Traffic flow model

The traffic and network function behavior models allow us to introduce the concept of traffic flow. In this study, a traffic flow $f \in F$, where F is the set of all possible flows, represents how a packet class originated by a source end point is transformed when it crosses a list of network functions. Formally, f is modeled as an alternating list of network nodes and packet classes: $f = [n_s, t_{sa}, n_a, t_{ab}, n_b, ..., n_j, t_{jk}, n_k, ..., n_p, t_{pd}, n_d]$. In this list, $n_s \in N_E$ is the source end point, $n_d \in N_E$ is the destination end point, and all the other nodes are middleboxes or firewalls, i.e., they belong to $N_M \cup N_F$. Each traffic t_{ij} of this list is a packet class possibly transmitted from node n_i to n_j in the flow, after n_i has applied a transformation (which may also be an identity function, leaving packets at they have been received) and if n_i has not dropped it. In fact, when crossing a node, the traffic can be forwarded, possibly changed, or dropped. In other words, this formalization allows to express how packets are modified and steered to pass through in case they were not stopped, and then the possibility that they could

be dropped will be taken into account by formulating tailored hard constraints in the MaxSMT problem.

Two auxiliary functions will later be used to formalize some constraints of the MaxSMT problem in relation to traffic flows. The first function is $\pi\colon F \rightarrow (N)^*$, which maps a flow to the ordered list of network nodes crossed by that flow. This list includes the destination, but not the source. The second function is $\tau\colon F \times N \rightarrow T$, which maps a flow and a node to the packet class, belonging to that flow, that is received by that node as input.

Given this abstract model, the traffic flow entities concretizing it may diverge depending on how single packets are grouped into classes. Here, we decided to adopt the Atomic Flow concretization, based on the idea of Atomic Predicates, originally proposed by the researchers Yang and Lam in [27], [28] and recently reused by other studies of network management [29], [30]. According to this concept, each complex predicate used to model the network can be split into a disjunction of simpler predicates, named Atomic Predicates, which are unique, minimal, and disjoint. The uniqueness of Atomic Predicates allows to assign them with representative integer identifiers, so that each complex predicate can be denoted as a set of integers, where each integer is the identifier of an Atomic Predicate composing the predicate disjunction.

Applying this concept to concretize the generic traffic flow model means that, after computing the set of Atomic Predicates for all the network predicates, the flow entities, named Atomic Flows, can be computed so that each packet class appearing in its alternating list is represented by an Atomic Predicate. The decision to adopt this grouping strategy has a two-fold advantage. On the one hand, this strategy provides a fine granularity for the definition of packet classes, because each traffic is the minimal one and is disjoint from the others. On the other hand, as previously explained, each traffic related to an Atomic Flow can be associated with an integer identifier, and consequently all operations on packet classes that will be modeled in the MaxSMT problem will simply work on integer numbers instead of complex predicates, thus helping to achieve better performance.

From here on, when we use the term traffic flow in the remainder of the paper, we will assume that it is an Atomic Flow concretizing this abstract model.

D. Network Security Policy model

Denoting as P the set of all NSPs defined by the administrator as input, each $p \in P$ is modeled as a tuple $p = (a, C)$. a expresses the NSP action, which is "allow" if p is a reachability policy, "deny" if it is an isolation policy. C is a predicate representing the NSP condition, which allows the identification of the packet class on which the NSP action must be applied. As C expresses a packet class, it is modeled in the same way as a traffic t, i.e., $C = (IPSrc, IPDst, pSrc, pDst, tProto)$. More precisely, C provides information on how the matching packet class appears at the source and at the destination of its crossed network path. In fact, the predicates *IPSrc* and *pSrc* specify conditions on the traffic generated by the source, while the predicates *IPDst*, *pDst*, and *tProto* conditions on the traffic received by the destination.

Among all the flows that may cross a network, we are interested only in the flows that satisfy the conditions of a $p \in P$. According to the provided model for the condition C, a flow $f = [n_s, t_{s,a}, ..., t_{z,d}, n_d]$ satisfies C if the following three conditions are fulfilled:

1) Its source and destination n_s, n_d have IP addresses matching *IPSrc* and *IPDst* respectively: $address(n_s) \subseteq C.IPSrc$ and $address(n_d) \subseteq C.IPDst$.
2) the packet class generated by the source n_s matches the condition sub-predicates *IPSrc* and *pSrc*: $t_{sa} \subseteq (C.IPSrc, *, C.pSrc, *, *)$.
3) The packet class received by the destination n_d matches the condition sub-predicates *IPDst*, *pDst*, and *tPrt*: $t_{zd} \subseteq (*, C.IPDst, *, C.pDst, C.tPrt)$.

We will denote the subset of flows satisfying $p.C.$ as $F_p \subseteq F$, and we will denote the union set of all these flow subsets as $F_P \subseteq F$.

E. Firewall configuration model

The model for the configuration of the distributed firewalling architecture is composed of two components: the activation status of each firewall and their filtering configuration.

For what concerns the activation status, the *active*: $N_F \rightarrow \mathbb{B}$ predicate maps a firewall $n \in N_F$ to true if n is active, to false otherwise. For each $n \in N_F$, the administrator has the faculty to impose that some firewalls must be active in the final solution of the auto-configuration problem. The *active* maps all those firewalls to true. For all the other firewalls, the *active* predicate is left free, i.e., there is no imposition on the Boolean value to which the predicate maps those firewalls. In fact, we want the Boolean value to be established by the solver as output.

For what concerns the filtering configuration of each firewall, it is composed of a set of filtering rules and a default action, that is applied to any packet class that does not match the condition of any specific rule. The default action is modeled with the *whitelist*: $N_F \rightarrow \mathbb{B}$ predicate maps a firewall $n \in N_F$ to true if it is configured in whitelisting mode (where the default action is "deny"), to false if it is configured in blacklisting mode (where the default action is "allow"). The specific filtering rules are modeled with the *rule*: $N_F \times T \rightarrow \mathbb{B}$ predicate, which maps a firewall $n \in N_F$ and a traffic t to true if n is configured with a filtering rule whose condition is matched by the packets of t, to false otherwise. We assume that the actions associated with these specific filtering rules are the opposite of the default action, i.e., if the firewall is in whitelisting mode, the filtering rules have "allow" as action, otherwise they have "deny". As for the *active* predicate, also the *whitelist* and *rule* predicates are left free so that the solver can establish them in a way that optimizes the power consumption objectives.

V. MaxSMT Problem Formulation

This section discusses how the formal models are used to formulate the constraints of the MaxSMT problem.

In particular, hard constraints are defined to enforce the satisfaction of the NSPs (Subsection V-A) and to concretize

the abstract models for the forwarding behavior of the network functions (Subsection V-B). In fact, the NSP satisfaction depends on the forwarding behavior of the network functions, i.e., it depends on which packet classes each network function blocks. Therefore, also that behavior must be represented with hard constraints, so that the automated solver can check if there is any incompatibility with the hard constraints previously described for NSP enforcement.

Instead, soft constraints are employed for the representation of the green optimization objectives related to network sustainability and power consumption (Subsection V-C). Finally, the MaxSMT problem that is thus formulated is automatically resolved by a solver employed by GreenShield, and the values assigned to the open predicates to provide the output to the network administrator (Subsection V-D).

A. Hard constraints related to security enforcement

All input NSPs must be successfully enforced in a correct solution for the firewall auto-configuration problem. Therefore, their satisfaction must be represented with hard constraints. These constraints differ depending on the type of NSP that must be enforced.

An isolation NSP $p \in P$ requires that all the traffic flows of the subset F_p satisfying its condition $p.C$ must not reach their destination. Therefore, for each flow of that subset, there must exist at least an active firewall in its path, successfully blocking the packet class it receives from that flow. This hard constraint is formulated in (1).

$$\forall f \in F_p.\ \exists n \in N_F.\ (n \in \pi(f) \wedge active(n) \wedge deny(n, \tau(f, n))) \quad (1)$$

A reachability NSP $p \in P$ requires that at least a traffic flow of the subset F_p satisfying its condition $p.C$ must reach its destination. Therefore, there must exist at least a flow of that subset such that, if there is an active firewall in its path, that firewall does not block the packet class it receives from that flow. This hard constraint is formulated in (2).

$$\exists f \in F_p.\ \forall n \in N_F.\ (n \in \pi(f) \wedge active(n) \implies \neg deny(n, \tau(f, n))) \quad (2)$$

B. Hard constraints related to the forwarding behavior of the network functions

The forwarding behavior is represented with hard constraints, which vary depending on the specific network function type.

For what concerns the end points in N_E and the middleboxes in N_M, we assume that they never block any packet, because in the service provided by the NF the role of filtering traffic is enforced by firewalls. Therefore, the *deny* predicate is simply forced to map any node of those two subsets to false. This hard constraint is formulated in (3).

$$\forall n \in (N_E \cup N_M).\forall f \in F.(deny(n, \tau(f, n)) = false) \quad (3)$$

For what concerns the firewalls in N_F, they may block some packet classes depending on their filtering configuration. In particular, a firewall $n \in N_F$ blocks a traffic t if and only if the firewall is active and either it is in whitelisting mode without any allowing rule whose condition is matched by t, or it is in blacklisting mode with a denying rule whose condition is matched by t. This hard constraint is formulated in (4).

$$\begin{aligned} \forall n \in N_F.\ (deny(n, \tau(f, n)) &= (active(n) \wedge ((a) \vee (b)))) \\ (a) &= whitelist(n) \wedge \neg rule(n, t) \\ (b) &= \neg whitelist(n) \wedge rule(n, t) \end{aligned} \quad (4)$$

The *deny*, *active*, and *rule* predicates appearing in these constraints are left free, with minor exceptions related to restrictions imposed by the administrator. In case the administrator has specified his desire to make a certain firewall $n \in N_F$ as active, then the *active* predicate is simply forced to map that firewall to true, as shown in the hard constraint formulated in (5).

$$active(n) = true \quad (5)$$

A noteworthy consideration is that the *deny* predicate is shared by both the constraints about forwarding behavior and the constraints about NSP enforcement. Therefore, the solver will have to search for a predicate interpretation that satisfies all those constraints (i.e., for each pair of node and traffic, the solver must establish the Boolean value to which the predicate maps that pair) in order to produce a correct solution.

C. Soft constraints related to the optimization objectives

GreenShield pursues two optimization objectives:

1) it aims to minimize the overall power consumption related to firewall activation;
2) it aims to block the packet classes that must be prevented from reaching their destination as nearest as possible to their source so that fewer middleboxes must process the corresponding traffic flows and can consequently avoid consuming additional power.

These objectives are modeled as soft constraints in the MaxSMT problem. In this way, the solver will try to satisfy them as far as possible, but if it cannot satisfy them, it may still find a correct solution where all hard constraints are satisfied. For the representation of the soft constraints, we will use the $Soft(f, w, g)$ notation, where f is the formula which should be satisfied if possible, w is the weight representing the penalty to be paid if f cannot be satisfied, and g is a string representing the class of soft constraints to which f appears. As two optimization objectives are pursued, two classes will be used in this study, named obj_1 and obj_2, respectively.

For what concerns the first objective, for each firewall $n \in N_F$, a soft constraint is formulated to state that the *active* predicate should map n to false, if possible. That soft constraint is associated with the average power consumption w_n of firewall n as weight. This class of soft constraints is formulated in (6).

$$\forall n \in N_F.\ \ Soft(active(n) = false, w_n, obj_1) \quad (6)$$

If the solver is forced to activate firewall n to satisfy some hard constraints (e.g., the ones about NSP enforcement), w_n represents the penalty that it must pay to propose that solution. Consequently, minimizing the overall power consumption does not necessarily coincide with minimizing the number of activated firewalls, because a firewall may have an activation

Algorithm 1 computation of the weights for the constraint (7)

Input: an isolation policy $p \in P$
Output: the value of all weights $w_{n,f}$

```
for f ∈ F_p do
    w_count ← 1
    for n ∈ π(f) do
        if n ∈ N_F then
            w_{n,f} ← w_count
            w_count ← w_count + 1
```

weight higher than the sum of the weights of multiple other firewalls.

For what concerns the second objective, for each flow $f \in F_p$ satisfying the condition of an isolation policy $p \in P$, a number of soft constraints equal to the number of firewalls in the path $\pi(f)$ is introduced in the MaxSMT problem. Each one of these soft constraints states that the *deny* predicate should map the pair composed of a firewall n and the input traffic $\tau(n, f)$ to false, if possible. The weights $w_{(n,f)}$ associated with those constraints start from 1 for the first firewall encountered in the path $\pi(f)$ and are progressively increased by 1 for each crossed firewall. These weights are computed as shown in Algorithm 1, and the class of soft constraints using them is formulated in (7).

$$\begin{gathered}\forall p \in P |\ p.a = deny.\ \ \forall f \in F_p.\ \ \forall n \in N_F |\ n \in \pi(f).\\ Soft(deny(n, \tau(n, f)) = false, w_{(n,f)}, obj_2)\end{gathered} \tag{7}$$

If the solver can choose which firewall to use to block a certain traffic flow to satisfy a related isolation NSP, it will prefer using a firewall nearer to the flow source, because the penalty to falsify the *deny* predicate is lower when applied to that firewall.

For the resolution of the multi-objective problem composed by the soft constraints of the two classes previously described, we configure the automated MaxSMT solver to follow a lexicographic priority of objectives, and we declare the constraints of class *obj1* before the others to the solver, so that it gives higher priority to satisfy the first ones. The design choice of this relative priority is motivated by the fact that, as it can be seen from the firewall data sheets, the power consumption of a firewall has fluctuations related to the traffic load, but these variations are near to an average value. Therefore, when trying to optimize the overall power consumption, the most impactful choice is determining if a firewall must be activated.

D. Output derivation from the problem resolution

After the MaxSMT problem is formulated with all these constraints, GreenShield invokes the automated solver to search for the optimal correct solution.

If no solution satisfying all hard constraints exists, the solver does not produce any output model, i.e., it cannot provide any meaningful interpretation for the free predicates. GreenShield informs the network administrator about the problem unsatisfiability, so that inputs may possibly be modified accordingly.

Otherwise, the solver proposes the interpretation of the free predicates that corresponds to the optimal solution for the firewall auto-configuration problem. From that interpretation, GreenShield can produce the two outputs.

The first output, i.e., the activation scheme, is derived from the interpretation of the *active* predicate. For each firewall $n \in N_F$, GreenShield checks if the solver has set *active*(n) to true or false. In the former case, that firewall must be activated and configured to provide the requested security. In the latter, that firewall must not be activated because it would be redundant, and this allows to save power consumption.

The second output, i.e., the filtering configuration for each active firewall, is derived from the interpretation of the *whitelist* and *rule* predicates. For each firewall $n \in N_F$ such that *active*(n) = *true*, GreenShield finds out if it is configured in whitelisting or blacklisting mode by looking at the value that the solver decided for *whitelist*(n). Subsequently, it derives the more specific filtering rules by looking for all *rule*(n, t) that the solver mapped to *true*. For those instances, GreenShield computes a firewall rule, whose condition is the matching packet class t and the action is "allow" in case of whitelisting mode, "deny" in case of blacklisting mode.

It may sometimes happen for very large networks (e.g., networks composed of thousands of hosts and middleboxes) that the correspondingly modeled firewall configuration problem is solvable, but an optimized solution cannot be found in acceptable times. In these cases, if the administrator is not interested in optimization, the soft constraints (i.e., equations (6) and (7)) may be excluded from problem formulation. As they have a non-negligible impact on execution time, their removal can speed up the work of the automated solver, and at the same time the potentially found solution is correct, because correctness is independent from optimality.

VI. Implementation and Validation

A prototype implementation of GreenShield has been developed in Java, and it exploits Z3 (version 4.8.8), an open-source off-the-shelf theorem prover by Microsoft Research, as MaxSMT solver [31]. It also offers REST APIs, through which exchanged data can be represented in XML or JSON embedding, and which users can use to request the resolution of the green-oriented firewall configuration problem.

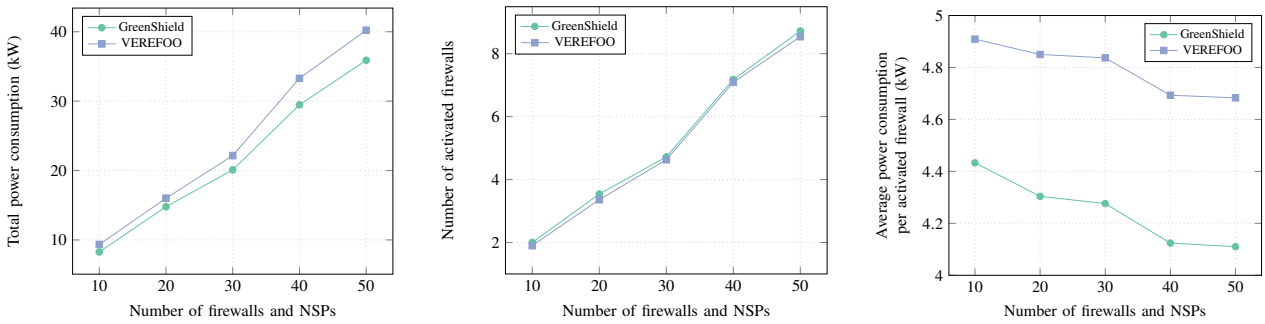
The GreenShield implementation has undergone optimization validation to understand the power consumption saving that it can achieve with respect to the state-of-the-art alternative VEREFOO (Subsection V-C), and performance validation to assess how it scales against parameters such as the numbers of NSPs and firewalls (Subsection VI-B). All the validation tests were carried out on a 4-core Intel i7-6700 3.40 GHz workstation equipped with 32 GB RAM.

A. Optimization validation

In the optimization validation, GreenShield was compared with VEREFOO in assessing how it can better achieve the two green-oriented goals that were included in its core design.

1) Minimization of power consumption due to firewall activation: The first optimization objective is to minimize power consumption due to firewall activation, by activating the least energy-consuming combination of firewall instances

This article has been accepted for publication in IEEE Transactions on Network and Service Management. 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/TNSM.2024.3452150



(a) Total power consumption (b) Firewall activation status (c) Average power consumption per firewall

Fig. 3: Optimization related to power consumption due to firewall activation

	Number of firewalls/NSPs				
	10	20	30	40	50
Total power consumption saving (kW)	1.08	1.22	2.05	3.82	4.33

TABLE VI: Total power consumption savings with GreenShield

	Number of firewalls/NSPs				
	10	20	30	40	50
Average power consumption saving (kW)	0.476	0.545	0.561	0.559	0.573

TABLE VII: Average power consumption savings with GreenShield

that still allows satisfying all the NSPs. For the assessment of this objective, GreenShield and VEREFOO were run on 50 variations of five use cases of progressively increasing size. There, the total power consumption of the activated firewalls, the number of activated firewalls, and the average power consumption per activated firewall were experimentally estimated as the average of their 50 corresponding values measured in the 50 use case variations.

Each use case is characterized by two parameters, establishing its size: the number of firewalls that may be possibly activated, and the number of NSPs that must be enforced. In each use case, these two numbers are assigned with the same value, progressively from 10 to 50, with an incremental factor of 10. The network topologies used for these validation tests are artificially synthesized extensions of the topology already shown in Fig. 1. These extensions have been obtained by attaching a progressively higher number of sub-graphs to them. Similarly, NSPs are variations of the exemplifying ones reported in TABLE IV, and an equality relationship has been maintained between the numbers of isolation and reachability NSPs. The 50 variations created for each use case characterized by a specific number of firewalls and NSPs are automatically synthesized so that in each one everything is kept the same, except for the power consumption value of each firewall, which is randomly selected in the range 2300 and 7000 W. These values were chosen after analyzing the most common power consumption values of real-world commercial firewall implementations.

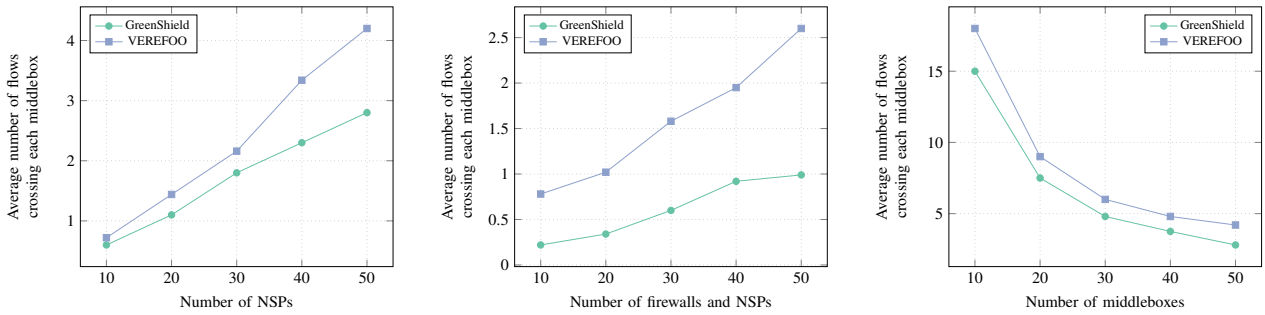
All the findings of this optimization assessment phase are reported in Fig. 3.

Fig. 3a reports the total power consumption of the solutions computed by GreenShield and VEREFOO, averaged on the 50 variations of each use case. The differences between the power consumption values estimated in the output of the two frameworks are precisely detailed in TABLE VI. As seen from the chart and table, GreenShield can provide significant energy savings with respect to VEREFOO, which is in the magnitude order of kWs. Besides, this saving progressively increases with the enlargement of the use case size, thus showing that GreenShield is even more effective in addressing big-sized firewall configuration problems. These results were possible due to the definition of a green-oriented firewall formal model and the soft constraint class (6). Thanks to them, GreenShield analyzes all possible solutions satisfying the input NSPs and selects the one that activates a firewall combination that minimizes power consumption.

Fig. 3b reports the number of firewalls activated in the solutions computed by GreenShield and VEREFOO, again averaged on the 50 variations of each use case. From this point of view, GreenShield activates a slightly higher number of firewalls than the number of firewalls VEREFOO decides to allocate in its solutions. However, this result was expected because a smaller number of firewalls does not imply higher energy savings. In fact, as already discussed since Section I, firewall implementations have highly variable power consumption. Differently from VEREFOO, which lacks any green-oriented feature, GreenShield takes this consideration into account and may opt for activating multiple firewalls instead of a single one, if that has a more significant power consumption than the combination of the other ones.

Fig. 3c reports the average power consumption per activated firewall in the solutions computed by GreenShield and VEREFOO, derived as a ratio between the corresponding values of Fig. 3a and Fig. 3b. TABLE VII complements these findings by showing the actual numerical saving achieved by GreenShield per each firewall. The plots in this chart have a decreasing trend because the slope of the number of activated firewalls (used as the denominator) is steeper than the slope of the total power consumption (used as numerator), as expected.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. 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/TNSM.2024.3452150



(a) Scenario with equinumerous NSP types (b) Scenario with only isolation NSPs (c) Scenario with varying network size

Fig. 4: Optimization related to minimization of traffic processing

Still, these results show again that GreenShield provides power savings even for every single firewall, and this saving increases when the firewall configuration problem gets bigger.

An additional consideration is that the estimated metric in all the charts of Fig. 3 is power consumption, but also energy consumption may be analyzed. The two metrics are closely interconnected, because the consumed energy is computed as the product of the consumed power and the time in which the device (here, the firewall) is active. Consequently, if we contextualize all the findings of 3 into energy consumption, the advantages introduced by GreenShield are even more significant. If we consider the use case with 50 firewalls and 50 NSPs, the energy savings will be 4.33 kW per second. Firewalls are expected to keep active for long time to ensure the required network defense, so the energy savings will steadily increase.

2) Minimization of power consumption due to traffic processing: The second optimization objective is to minimize power consumption due to traffic processing, by making firewalls block undesired flows as nearest as possible to their sources, and thus by reducing the number of traffic flows crossing each network middlebox, firewalls included. Regarding the assessment of this objective, the metric of interest has been the average number of flows crossing each middlebox. In fact, a middlebox, such as a firewall, consumes less power if less traffic crosses it. This metric has been evaluated in three different scenarios.

First, GreenShield and VEREFOO were run again on the same use cases previously discussed for the assessment of the first optimization objective. In those use cases, the number of isolation NSPs was always equal to the number of reachability NSPs. Fig. 4a shows that, in that scenario, GreenShield produces solutions that allow each network middlebox to be crossed by fewer traffic flows than VEREFOO does. The difference between the two tools is more significant for bigger problem sizes: when there are 50 NSPs, in the solution computed by GreenShield each middlebox is crossed by around 1.5 flows on average less than in the one computed by VEREFOO.

Second, GreenShield and VEREFOO were run on use cases based on the same network topologies used for the first scenario, but with different NSPs. Here, only isolation NSPs are requested. This second scenario is motivated by the fact that a flow related to a reachability NSP must cross multiple middleboxes to reach the destination. Therefore, the presence of these flows may make the advantages introduced by Greenshield less evident. In fact, the results derived from these other tests, reported in Fig. 4b, show that Greenshield can produce remarkably optimized solutions, as in all analyzed use cases each middlebox is crossed by on average less than on traffic flow, a result which could not be achieved by VEREFOO, for which more than 2.5 flows were needed for the management of the most complex scenario. This result also means that almost all the firewalls are activated in a position representing the first hop in the network for the communications between end points, thus avoiding that they cross additional intermediate network nodes (or other firewalls, such as the ones near the destinations).

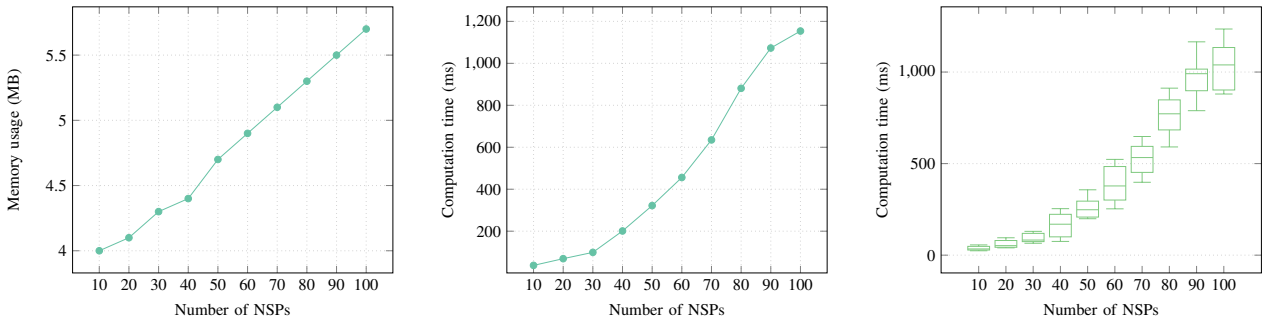
Third, keeping the total number of requested NSPs fixed to 50, with equipartition between the two NSP types, other use cases were used for the optimization validation by varying the network size and, in particular, by progressively increasing the number of middleboxes from 10 to 50, with an incremental factor of 10. For this scenario, the computed results, depicted in Fig. 4c, show that the difference in terms of the average number of flows crossing each middlebox is almost constant, with a value of around 2.5-3 flows. This shows that, even when the network is more extensive and, consequently, the averages become lower, GreenShield can still outperform VEREFOO, maintaining a significant improvement.

All these experimental results confirm that GreenShield successfully allows each intermediate node, firewalls included, to be crossed by a limited number of traffic flows, so that it can consume less power for their processing.

B. Performance validation

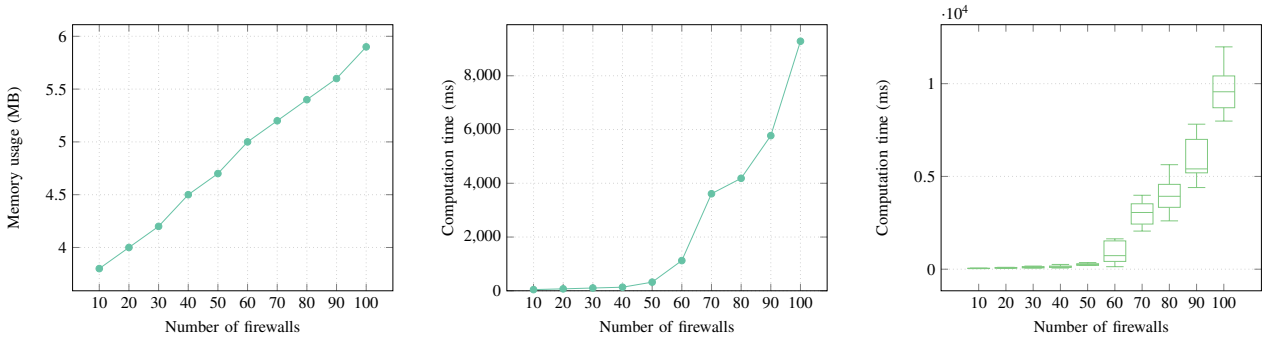
The main objectives of the performance validation of the GreenShield implementation were to understand the impact of two main factors (i.e., the number of possibly activated firewalls and the number of NSPs to be enforced) on the memory usage and computation time, and to evaluate the behavior of this framework when executed on MaxSMT problem instances with increasing size, where all parameters increase in a coordinated way. The network topologies on which GreenShield has been used for these validation tests are again artificially synthesized extensions of the topology already shown in Fig. 1.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. 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/TNSM.2024.3452150



(a) Memory usage (b) Time validation (c) Box plot

Fig. 5: Scalability validation against the number of NSPs



(a) Memory usage (b) Time validation (c) Box plot

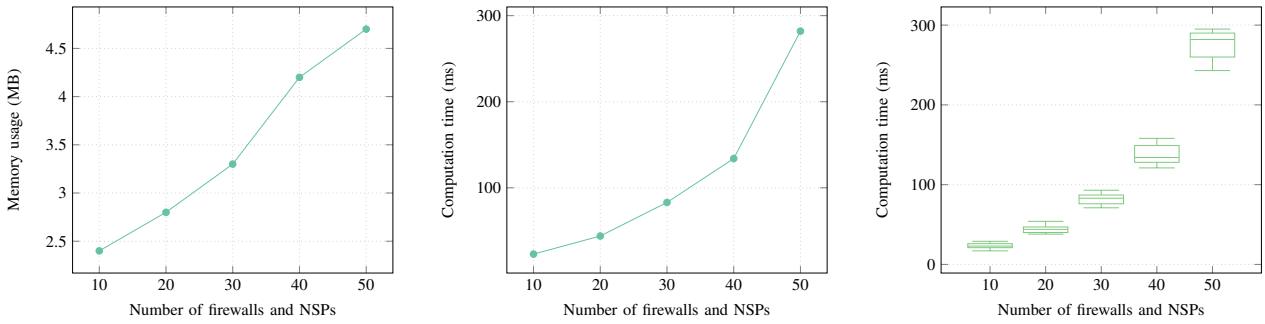
Fig. 6: Scalability validation against the number of firewalls

First, the impact related to the numbers of NSPs and firewalls on the framework behavior has been analyzed, and the results of this analysis are reported in Fig. 5 and Fig. 6, respectively. For the tests aiming to assess the impact of the number of NSPs, the value assigned to that parameter was progressively increased from 10 to 100 with an incremental factor of 10, while the number of firewalls included in the network where those NSPs must be enforced is set constant to 50. On the contrary, for the tests aiming to assess the impact of the number of firewalls, their value was similarly increased from 10 to 100 with an incremental factor of 10, while the number of NSPs to be enforced in their network is set constant to 50. Keeping a parameter fixed while varying the other one was required to understand which one of them has a more significant impact on GreenShield.

Fig. 5a and Fig. 6a report the trend of the peak memory usage as the number of NSPs and the number of firewalls increase, respectively. The highest value measured among all analyzed cases is 5.9 MB, which is low and almost negligible. The trend is also linear with respect to both parameters. These results show that the increase of the NSP and firewall numbers does not impact the behavior of GreenShield, which needs a limited amount of memory to be executed successfully.

Fig. 5b and Fig. 6b report the trend of the computation time as the number of NSPs and the number of firewalls increase, respectively. In these two charts, each plotted value represents the average computed over 50 iterations, where the same network topology and NSPs are kept, and only the IP addresses of the nodes vary. This variation was required, because the time performance of the employed MaxSMT solver is known to vary substantially with the variation of the values of integer constants. As integers are used in the formulation of our MaxSMT problem for IP addresses, varying those addresses in different iterations and computing their average allowed us to minimize the influence of that variability. As 50 iterations were executed to compute the average computation time, for completeness, we report the achieved results in the form of box plots in Fig. 5c and 6c. The box plots depicted in those figures graphically show the minimum, 5th percentile, median, 95th percentile, and maximum for the computed results.

A first noteworthy consideration that can be drawn from the numerical results shown in those charts is that, in the worst case that was considered, GreenShield takes less than 10 seconds to compute the solution to the firewall configuration problem. Specifically, the scenario with 100 NSPs and 50 firewalls requires 1.1s, whereas the scenario with 100 firewalls and 50 NSPs requires 9.3s. In such short times, GreenShield does not simply find a solution for the problem, but it produces the optimal one with respect to the two pursued optimization objectives related to network sustainability. Thanks to the MaxSMT formulation of the auto-configuration problem, the solution can also be considered correct by construction. In comparison, network administrators would take much more time to write a possible configuration for a distributed firewall, they may introduce errors, and they may struggle to define the configuration that minimizes power consumption. Therefore, GreenShield can go beyond all limitations of traditional configuration approaches.



(a) Memory usage (b) Time validation (c) Box plot

Fig. 7: Scalability validation

A second consideration is that the number of firewalls in the network has a more significant impact than the number of NSPs. This can be derived from two aspects noticeable in the charts. On the one hand, solving a MaxSMT problem related to 100 firewalls and 50 NSPs requires around seven times more seconds than a problem characterized by inverted numbers of those parameters. On the other hand, the growth of the plot in Fig. 6b is more pronounced than the one in 5b. However, the cases where a distributed firewall has from 50 to 100 instances just represent worst cases, used in the validation analysis to stress the Greenshield framework to its limits. Thanks to those tests, we were able to confirm the feasibility of the proposed approach for networks of such size.

After examining the separate impact of the two parameters, we executed other performance tests to see the trends of memory usage and computation time when the numbers of NSPs and firewalls increase progressively by the same factor. Fig. 7 reports the results achieved by incrementing the value assigned in parallel to those parameters from 10 to 50, by an addictive factor of 10. Fig. 7a confirms that the memory usage is negligible, even when GreenShield is used to enforce 50 NSPs in a network with 50 possible firewalls. Similarly, Fig. 7b and Fig. 7c confirm that GreenShield can solve firewall configuration problems of significant sizes in limited time (e.g., less than half a second for enforcing 50 NSPs in a network with 50 firewalls).

In conclusion, even if GreenShield is mainly designed for green-oriented optimization, its scalability performance is still much better than what human administrators may achieve configuring firewalls manually.

VII. Conclusions and Future Work

This paper presented GreenShield, a novel methodology combining automation, formal verification and optimization for the automatic configuration of distributed firewalls in virtual networks. This approach pursues green objectives to reduce the impact that firewalls activation and filtering behavior have on the overall network power consumption. A prototype implementation has been developed following the design principles of GreenShield, and its validation showed that it can successfully finds solutions that are more optimized in terms of power consumption than a state-of-the-art counterpart approach of the literature.

As future work, we will investigate how to simplify the duty of the user to provide the input about firewall placement. A possible way to simplify it would be to make the administrator only identify the positions where firewalls may be potentially activated, and then for each one of these positions a sequence of three firewalls with different power consumption levels is modeled in the firewall configuration problem. However, modeling this sequence would increase the solution space, so it may decrease the execution time of the automated solver. Therefore, a full investigation of its impact is needed. Another future work related to firewall placement is to research if moving non-activated firewalls to replace activated firewalls may help to decrease the overall power consumption in some particular cases. Moreover, we plan to extend this study to other network security functions, such as VPN gateways, by investigating how their behavior impacts power consumption, and to extend GreenShield to optimize their automatic configuration. Finally, we will start to investigate another green-oriented problem, which is about the impact of routing operations to energy consumption.

References

[1] K. Wang, X. Hu, H. Li, P. Li, D. Zeng, and S. Guo, "A survey on energy internet communications for sustainability," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 3, pp. 231–254, 2017.

[2] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[3] E. S. Al-Shaer and H. H. Hamed, "Modeling and management of firewall policies," *IEEE Trans. Netw. Serv. Manag.*, vol. 1, no. 1, pp. 2–10, 2004.

[4] D. Bringhenti, G. Marchetto, R. Sisto, and F. Valenza, "Automation for network security configuration: State of the art and research trends," *ACM Comput. Surv.*, vol. 56, no. 3, pp. 57:1–57:37, 2024.

[5] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "Automated firewall configuration in virtual networks," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 2, pp. 1559–1576, 2023.

[6] A. P. Bianzino, C. Chaudet, D. Rossi, and J. Rougier, "A survey of green networking research," *IEEE Commun. Surv. Tutorials*, vol. 14, no. 1, pp. 3–20, 2012.

[7] A. C. Riekstin, G. C. Januario, B. B. Rodrigues, V. T. Nascimento, T. C. M. de Brito Carvalho, and C. Meirosu, "A survey of policy refinement methods as a support for sustainable networks," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 222–235, 2016.

[8] T. C. M. B. Carvalho, A. C. Riekstin, M. Amaral, C. H. A. Costa, G. C. Januario, C. K. Dominicini, and C. Meirosu, "Towards sustainable networks - energy efficiency policy from business to device instance levels," in *Proc. of the 14th International Conference on Enterprise Information Systems, Volume 3, Wroclaw, Poland, 28 June - 1 July, 2012*. SciTePress, 2012, pp. 238–243.

[9] Y. B. Udupi, A. Sahai, and S. Singhal, "A classification-based approach to policy refinement," in *Proc of. IM 2007, 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany, 21-25 May 2007*. IEEE, 2007, pp. 785–788.

[10] R. Craven, J. Lobo, E. Lupu, A. Russo, and M. Sloman, "Policy refinement: Decomposition and operationalization for dynamic domains," in *Proc. of the 7th International Conference on Network and Service Management, CNSM 2011, Paris, France, October 24-28, 2011*. IEEE, pp. 1–9.

[11] A. Merlo, M. Migliardi, and L. Caviglione, "A survey on energy-aware security mechanisms," *Pervasive and Mobile Computing*, vol. 24, pp. 77–90, 2015.

[12] U. Chauhan, D. Sharma, S. Mohril, and G. P. Singh, "A survey on energy-efficient networking and its improved security features," pp. 11–17, 2021.

[13] A. Merlo, M. Migliardi, and E. Spadacini, "Balancing delays and energy consumption in ips-enabled networks," in *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2016, pp. 267–272.

[14] O. Akin, U. C. Gulmez, O. Sazak, O. U. Yagmur, and P. Angin, "Greenslice: An energy-efficient secure network slicing framework." *J. Internet Serv. Inf. Secur.*, vol. 12, no. 1, pp. 57–71, 2022.

[15] S. Rajasoundaran, S. Sivakumar, S. Devaraju, M. J. Pasha, and J. Lloret, "A deep experimental analysis of energy-proficient firewall policies and security practices for resource limited wireless networks," *Security and Privacy*, p. e450, 2024.

[16] N. Schnepf, R. Badonnel, A. Lahmadi, and S. Merz, "Rule-based synthesis of chains of security functions for software-defined networks," *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.*, vol. 76, 2018.

[17] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in network service chaining," in *Proc. of IEEE SDN for Future Networks and Services, SDN4FNS 2013, Trento, Italy, November 11-13, 2013*, 2013, pp. 1–7.

[18] M. Yoon, S. Chen, and Z. Zhang, "Minimizing the maximum firewall rule set in a network with multiple firewalls," *IEEE Trans. Computers*, vol. 59, no. 2, 2010.

[19] M. A. Rahman and E. Al-Shaer, "Automated synthesis of distributed network access controls: A formal framework with refinement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, 2017.

[20] Y. Bartal, A. J. Mayer, K. Nissim, and A. Wool, "*Firmato*: A novel firewall management toolkit," *ACM Trans. Comput. Syst.*, vol. 22, no. 4, pp. 381–420, 2004.

[21] F. Cuppens, N. Cuppens-Boulahia, T. Sans, and A. Miège, "A formal approach to specify and deploy a network security policy," in *Proc. of Formal Aspects in Security and Trust: Second IFIP TC1 WG1.7 Workshop on Formal Aspects in Security and Trust (FAST), an event of the 18th IFIP World Computer Congress, August 22-27, 2004, Toulouse, France*, vol. 173, 2004, pp. 203–218.

[22] P. Verma and A. Prakash, "FACE: A firewall analysis and configuration engine," in *Proc. of the 2005 IEEE/IPSJ International Symposium on Applications and the Internet (SAINT 2005), 31 January - 4 February 2005, Trento, Italy*. IEEE Computer Society, 2005, pp. 74–81.

[23] N. B. Youssef and A. Bouhoula, "A fully automatic approach for fixing firewall misconfigurations," in *Proc. of the 11th IEEE International Conference on Computer and Information Technology, CIT 2011, Pafos, Cyprus, 31 August-2 September 2011*, 2011, pp. 461–466.

[24] K. Adi, L. Hamza, and L. Pene, "Automatic security policy enforcement in computer systems," *Comput. Secur.*, vol. 73, pp. 156–171, 2018.

[25] D. Ranathunga, M. Roughan, P. Kernick, and N. Falkner, "The mathematical foundations for mapping policies to network devices," in *Proc. of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26-28, 2016*, 2016, pp. 197–206.

[26] A. El-Hassany, P. Tsankov, L. Vanbever, and M. T. Vechev, "Netcomplete: Practical network-wide configuration synthesis with autocompletion," in *Proc. of the 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018*, S. Banerjee and S. Seshan, Eds., 2018, pp. 579–594.

[27] H. Yang and S. S. Lam, "Real-time verification of network properties using atomic predicates," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 887–900, 2016.

[28] ——, "Scalable verification of networks with packet transformers using atomic predicates," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2900–2915, 2017.

[29] P. Zhang, X. Liu, H. Yang, N. Kang, Z. Gu, and H. Li, "Apkeep: Realtime verification for real networks," in *Proc. of the 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, 2020, pp. 241–255.

[30] D. Bringhenti, S. Bussa, R. Sisto, and F. Valenza, "A two-fold traffic flow model for network security management," *IEEE Trans. Netw. Serv. Manag.*, 2024.

[31] L. M. de Moura and N. S. Bjørner, "Z3: an efficient SMT solver," in *Proc. of the 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*, ser. Lecture Notes in Computer Science, vol. 4963, 2008, pp. 337–340.

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 an Assistant Professor with time contract. His research interests include novel networking technologies, automatic orchestration and configuration of security functions in virtualized networks, formal verification of network security policies.

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.