Improving Web Protection in Virtual Networks with Automatic WAF Configuration

Improving Web Protection in Virtual Networks with Automatic WAF Configuration

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

***Abstract*—The higher complexity introduced by virtualization in computer networks has made web-based threats more impactful, due to a larger attack surface and increased function heterogeneity. The traditional manual approaches for the configuration of Web Application Firewalls are no longer feasible in this environment, as they would be excessively time-consuming, error-prone, and unoptimized. Even if configuration automation may represent a solution to the problem, it has been investigated in the literature only for other firewall types, mainly packet filters. In order to fill this gap, this paper proposes a novel configuration approach for distributed Web Application Firewalls, which combines security automation, formal verification, and optimization by leveraging policy-based management and formulating the problem as a Maximum Satisfiability Modulo Theories instance. This approach can automatically establish both filtering rules defined over specific HTTP attributes or based on the OWASP Core Rule Set, thus providing high expressiveness and effectiveness in mitigating different threats. The scalability validation conducted on the framework developed to implement this method showed that it can solve large configuration problems in a few seconds.**

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

I. Introduction

In recent years, the complexity of computer networks has dramatically increased, as virtualization paradigms such as Network Function Virtualization (NFV) and Software-Defined Networking (SDN) have enabled the creation of larger and more heterogeneous network architectures. As a consequence, attackers can exploit an expanded attack surface to intrude vulnerable assets and compromise critical services by leveraging weakly isolated network slices and virtual functions. In particular, web-based attacks remain among the most common and impactful threats, since modern web applications expose large, constantly evolving interfaces (e.g., REST APIs, microservices endpoints, and third-party integrations) and are often deployed at scale with short release cycles [1].

Distributed Web Application Firewalls (WAFs) represent a primary security control against application-layer attacks such as SQL injection, Cross-Site Scripting (XSS), command injection, and HTTP protocol manipulation. In virtualized environments, security functions such as Squid proxies, ModSecurity-based WAFs, and NGINX/OpenResty instances can be deployed as Virtual Network Functions (VNFs) and dynamically instantiated to enforce security policies close to protected assets. Compared to packet filters such as iptables or nftables, WAFs operate at the application layer and can inspect HTTP payloads, parse request parameters, and apply semantic validation rules, thus enabling fine-grained protection against web-specific threats.

Nevertheless, writing a correct distributed WAF configuration is not a trivial task. On the one hand, configuring a distributed WAF entails two sub-tasks: allocation scheme establishment, i.e., defining where the different instances should be positioned in the network, and filtering rule generation. These two operations should preferably be carried out jointly to provide effective and optimized protection. On the other hand, defining effective filtering rules requires deep security expertise and a thorough understanding of both the protected application's logic and legitimate traffic patterns. Rule generation and deployment also often need to be performed under strict time constraints, for instance as a reaction to an ongoing or newly discovered attack.

Unfortunately, traditional approaches to security configuration are still largely manual and, consequently, become time-consuming, suboptimal, and error-prone when applied to modern, complex, and highly dynamic virtual networks [2], [3]. Furthermore, although automatic firewall configuration has been extensively studied in the literature, most existing approaches focus on packet filters operating at the network and transport layers, generating rules based on header fields such as IP addresses, ports, and protocols. These techniques cannot be directly applied to WAFs, which require semantic inspection of HTTP requests, parameters, cookies, and payload content, as well as awareness of application-specific behavior. WAF deployments may also rely on generic rule sets such as the OWASP Core Rule Set (CRS), which provide baseline protection against well-known vulnerabilities but remain application-agnostic and require significant manual tuning to balance security and false positives [4].

In order to overcome these limitations, this paper proposes an automatic firewall configuration approach, specifically tailored to WAFs, addressing both allocation scheme and filtering rule generation sub-tasks simultaneously. The methodology followed by this approach leverages policy-based management as a founding principle, so that the users are not expected to know the low-level languages for WAF configuration, but can express their desired security requirements (e.g., blocking a certain web domain or attack) with high-level sentences named policies. Moreover, it enhances automation with two relevant security-oriented features, i.e., formal verification and optimization of the produced configuration. This achievement is reached by formulating and solving the WAF configuration

problem with a technique rooted in constraint programming, i.e., Maximum Satisfiability Modulo Theories (MaxSMT). A prototype framework, based on this formal approach, has then been developed in order to evaluate experimentally its feasibility and scalability, and prove the achieved advancements in the literature.

The remainder of this paper is structured as follows. Section II dissects related work on firewall configuration automation. Section III describes the proposed approach tailored to WAFs, including formal details about the MaxSMT problem. Section IV discusses how the framework implementing the approach has been validated. Section V concludes the paper and outlines future work.

II. Related Work

The literature on network security configuration automation has focused on a large variety of controls, from VPN gateways [5] to honeynets [6]. However, concerning firewalling, existing studies only address the two configuration sub-tasks (allocation and rule generation) for packet filters, i.e., functions that can only use the IP 5-tuple fields to make filtering decisions.

The packet filter allocation task has been investigated less extensively, and mainly in the context of the more generic problem of security service design. Studies such as [7]–[10] define how different security middleboxes, including packet filters, should be combined to enforce multiple requirement types. Unlike them, ConfigSynth [11] solves the allocation problem in the same way considered by this paper, i.e., by establishing how packet filters, in addition to VPN gateways and intrusion detection systems, should be placed in an already defined network service. However, this method cannot guarantee that all the requested requirements are satisfied, because they are modeled as relaxable objectives in finding the optimum trade-off among the security level, the allocation cost and usability.

The packet filter rule generation task has represented a cornerstone in the security automation literature. This problem started to be analyzed even before the advent of network virtualization. After initial studies focused only on centralized packet filters [12]–[14], others extended the focus to distributed versions [15], [16], sometimes providing formal correctness assurance [17], [18]. More recently, research on firewall rule generation has also shifted toward improving usability and exploring the integration of emerging technologies, including artificial intelligence techniques [19]–[21], even if formal correctness and security optimization are not guaranteed anymore.

Finally, few studies address both sub-tasks simultaneously [22]–[27]. The methods described in [22]–[24] are inherently limited by their exclusive support of function chains instead of branched network topologies, which represent the most common scenario nowadays. Instead, the approach in [25] does not consider the possible presence of middleboxes that can alter the traffic, albeit they represent one of the most crucial reasons of heterogeneity in modern networks. The most feature-complete approach is VEREFOO [26], [27], as it combines automation, formal verification and security for the complete distributed packet filter configuration, and was proved to be sufficiently efficient for the management of virtual networks.

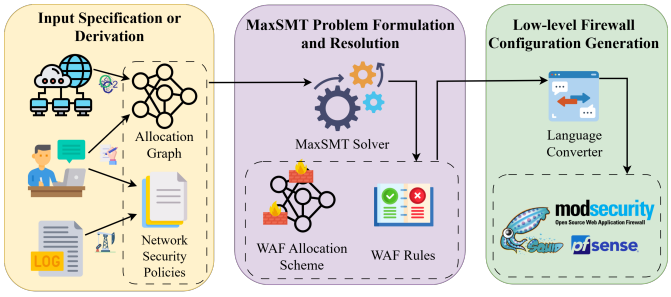

Fig. 1: The workflow of the proposed approach

In this context, the approach proposed in this paper adopts the VEREFOO methodology as a starting point and extends it to support the specific features of WAFs, as it was already the state of the art for packet filters. In doing so, it is able to fill the existing gap in the literature, i.e., the lack of automatic configuration techniques oriented to web security requirements, while preserving the features of formal verification and optimization.

III. The Proposed Approach

The approach proposed in this paper is based on three main phases, as shown in Fig. 1:

1) input specification or derivation;
2) MaxSMT problem formulation and resolution;
3) low-level WAF configuration generation.

A. Input Specification or Derivation

The methodology of this paper requires two main inputs: (i) a virtual network description, named Allocation Graph (AG); (ii) a set of Network Security Policies (NSPs), describing the web security requirements to be enforced in the network.

1) Allocation Graph: The logical topology of the virtual network is modeled as an AG, which is a directed graph defined as $G = (N, L)$. N is the vertex set including network nodes of different types, i.e., $N = N_E \cup N_M \cup N_A$. N_E is the set of endpoints, which may be single hosts starting or receiving a communication, or subnetworks representing host groups. N_M is the set of middleboxes, such as network address translators and load balancers, firewalls excluded, as allocating and configuring them is the objective itself of this proposal. N_A is a set of special nodes, named Allocation Places (AP), which represent the possible positions in the logical topology where a WAF may be allocated, if required. Instead, L is the edge set, including all the links putting nodes into communication.

This representation does not take into account the embedding of the virtual topology into the physical infrastructure, as that problem, named Virtual Network Embedding (VNE), is complementary to the one addressed in this paper, and has already been exhaustively investigated in the literature

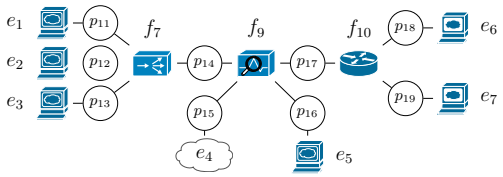

Fig. 2: Allocation Graph example

[28]. Moreover, it may be specified by a human administrator knowing all the details of their managed network, or derived from the output of context discovery protocols, based on languages such as OpenC2, which can recursively query context providers that describe services, relationships, and network functions in order to build the whole graph automatically [29].

An AG example is depicted in Fig. 2. The nodes with identifier starting with p are the APs. The APs near web clients, i.e., e_6 and e_7, may host personal virtual WAFs that can be installed on those endpoints. Instead, the other ones would host virtual WAF middleboxes.

2) Network Security Policies: The NSP set expresses connectivity requirements to be enforced in the AG, including web security ones. Given P the set of all required NSPs, $p \in P$ is formalized as the tuple $p = (a, c)$, where $p.a$ is the policy action, while $p.c$ is the policy condition identifying the traffic to which the action must be applied. Note that the operator "." is here used to identify a specific element of a tuple.

Three NSP types are envisioned in this study, i.e., $P = P_D \cup P_O \cup P_A$.

- P_D is the set of *HTTP attribute-based isolation* policies, expressing which web communications should be blocked in the network, based on features of the HTTP header. For $p \in P_D$, $p.a =$ *deny*, whereas $p.c = (IPSrc, IPDst, portSrc, portDst, trnProto, HTTPMethod, URL, Domain)$. The first five condition values represent the IP 5-tuple of the matching packets, while the other three can be used to identify the HTTP method used for the interaction, and the possibly contacted URL or domain. Each condition field can be assigned the wildcard value $*$ to specify that any value is accepted as matching for that specific field.
- P_O is the set of *OWASP CRS-based isolation* policies, expressing which web communications should be blocked based on the specific application-layer attacks identified by the OWASP CRS. For $p \in P_O$, $p.a =$ *deny* again, whereas $p.c = (IPSrc, IPDst, CRSRuleID)$ identifies the two endpoints among which the specific attack denoted by the ID of the OWASP CRS rule should be blocked.
- P_A is the set of *reachability policies*, expressing the communications that must be allowed so as to provide requested connectivity among selected endpoints. For $p \in P_A$, $p.a =$ *allow*, whereas $p.c = (IPSrc, IPDst, portSrc, portDst, trnProto, HTTPMethod, URL, Domain)$, as it was for the HTTP attribute-based isolation policies.

As for the AG specification, NSPs may be specified manually by a human, or derived automatically. For example, isolation policies can be extracted by parsing the log files produced by intrusion detection systems and associating alerts with the corresponding URL, domain or OWASP CRS rules.

TABLE I reports some examples for each NSP type.

TABLE I: Network Security Policy examples

HTTP attribute-based isolation policies

Action	IPSrc	IPDst	portSrc	portDst	trnProto	HTTPMethod	URL	Domain
deny	192.0.2.*	198.51.100.10	*	80	TCP	POST	/admin	internal.example
deny	198.51.100.*	192.0.2.20	*	443	TCP	PUT	/api/config	management.example

OWASP CRS-based isolation policies

Action	IPSrc	IPDst	CRSRuleID
deny	192.0.2.*	198.51.100.*	942100 (REQUEST-942-APPLICATION-ATTACK-SQLI)
deny	198.51.100.*	192.0.2.*	941130 (REQUEST-941-APPLICATION-ATTACK-XSS)

Reachability policies

Action	IPSrc	IPDst	portSrc	portDst	trnProto	HTTPMethod	URL	Domain
allow	192.0.2.*	203.0.113.30	*	443	TCP	GET	/public	www.service.example
allow	198.51.100.*	203.0.113.40	*	80	TCP	GET	*	monitoring.example

B. MaxSMT Problem Formulation and Resolution

After being fed with all these inputs, the proposed approach formulates the WAF configuration problem as a partial weighted MaxSMT problem. Unlike traditional SMT, it is characterized by constraints of two types: hard constraints to be always satisfied in any correct solution, soft constraints to be satisfied only as far as possible. Each soft constraint is also assigned a weight, so that the goal is to maximize the sum of the weights assigned only to the satisfied soft constraints. Such formulation ensures a-priori formal correctness assurance, as long as all problem models and clauses capture all the information possibly impacting the solution correctness. It also provides optimization, as soft constraints can be used to embed optimality goals.

All these constraints are defined upon the models previously introduced, and also on functions and predicates defined on them. The most interesting aspect is that certain variables, functions and predicates are intentionally left unspecified, meaning their interpretation is determined by the MaxSMT solver employed to address the configuration problem. These elements provide an effective way to describe both the WAF allocation strategy and the rule configuration, since their values are not predefined but instead emerge as part of the solution generated by the solver. Based on this perspective, the allocation strategy relies on a predicate named *allocated*, which is associated with the APs in the AG. When applied to a specific AP $n \in N_A$, this predicate evaluates to true if a firewall is placed at that AP in the computed solution, and to false otherwise. In a similar manner, for each AP $n \in N_A$ where a WAF may be deployed, a predicate called *configured* is defined over the set of potential configuration rules R_n, named *placeholder rules*, which have the same model structure as NSP in terms of action and condition, but include free variables unlike them. Consequently, when the solver assigns a rule to a WAF, the *configured* predicate evaluates to true for that rule. At the same time, the solver can thus determine concrete values for the rule's variables.

TABLE II: Summary of MaxSMT Constraint Categories

Type	Purpose	Description
Hard	Flow propagation	They model traffic propagation across the AG depending on its topology and composing middleboxes.
Hard	Network function behavior	They model the possible traffic transformation by the middleboxes.
Hard	WAF semantics	They enforce that an AP acts as forwarder if $\neg allocated(n)$, and as filtering node otherwise.
Hard	P_D enforcement	They ensure that HTTP attribute-based deny policies are satisfied along each matching flow.
Hard	P_O enforcement	They guarantee blocking of flows matching a specific OWASP CRS rule.
Hard	P_A enforcement	They prevent unintended blocking of required reachability policies.
Soft	Allocation minimization	They encode the goal to reduce deployed WAF instances.
Soft	Rule minimization	They encode the goal to reduce rule sets.

The hard and soft constraints designed for this MaxSMT problem are concisely reported in TABLE II, and explained in more detail in the following.

Hard constraints encode constraints that must always be satisfied. Specifically, they model: (i) the way traffic flows specified by the NSPs cross the network, based on the interconnection among the elements composing the AG; (ii) the behavior of service functions, which may alter traffic; (iii) the behavior of allocable WAFs, whose constraints are enforced only if they are actually allocated, otherwise the AP acts as a simple forwarder; and (iv) the enforcement of all requested NSPs. Due to space limitation, only a representative example is reported here in (1), i.e., the one about the enforcement of OWASP CRS-based isolation policies.

$$\begin{gathered}\forall p \in P_O.\, \forall f \in F_p.\, \exists n \in N_A.\, (n \in \pi(f) \wedge \\ \textit{allocated}(n) \wedge \exists r \in R_n.(\textit{configured}(r) \wedge r.a = \text{deny} \wedge \\ p.c.\textit{IPSrc} \subseteq r.c.\textit{IPSrc} \wedge p.c.\textit{IPDst} \subseteq r.c.\textit{IPDst} \wedge \\ p.c.\textit{CRSRuleID} = r.c.\textit{CRSRuleID}))\end{gathered} \tag{1}$$

This formula includes additional formalism: F_p is the set of traffic flows that match the condition of p, and π is a function that maps a traffic flow to the list of crossed nodes. Thanks to these elements, the formula states that, for each traffic flow satisfying the condition of policy $p \in P_O$, there must be an AP $n \in N_A$ with an allocated WAF, characterized by a configured rule satisfying the policy itself.

Soft constraints encode two optimization goals: (i) minimization of the WAF allocation scheme, so as to reduce resource consumption for the activation of the softwarized instances on physical machines; (ii) minimization of the rule set generated for each allocated WAF, with the aim of speeding up their filtering operations.

The first goal is achieved by expressing that, for every AP $n \in N_A$, the predicate *allocated(n)* should preferably evaluate to false. This concept is formalized in (2) using the notation $Soft(c, w)$, where c denotes a soft constraint and w its associated weight. Clearly, a solution in which no WAF is placed at all is not feasible, as no isolation policy would then be enforced. However, since a MaxSMT solver seeks to maximize the total weight of satisfied soft clauses, it will attempt to satisfy as many of them as possible, thus minimizing the number of allocated WAFs, while still ensuring that all hard constraints are satisfied.

$$Soft(\neg \textit{allocated}(n), w_n) \tag{2}$$

Instead, the second goal is achieved by expressing that, for every placeholder rule $r \in R_n$ for any WAF possibly allocated in $n \in N_A$, the predicate *configured(r)* should preferably evaluate to false. This formulation, reported in (3), means that the optimal solution would be the one in which no rules are installed in the distributed WAF.

$$Soft(\neg \textit{configured}(r), w_r) \tag{3}$$

The weights w_r assigned to the soft constraints related to rule generation are lower than the weights w_n assigned to the clauses related to allocation, as shown in (4), so that the latter objective has higher priority than the former. This order is established because the impact in terms of resource consumption of activating a new WAF is higher than just deploying multiple rules in the same firewall instance.

$$\forall n \in N_A.\ \sum_{r \in R_n} (w_r) < w_n \tag{4}$$

After all the hard and soft constraints are formulated, a MaxSMT solver is used to analyze the solution space and search for the solution that satisfies all the hard constraints, while aiming at maximizing the total weight of the satisfied soft constraints. In this optimal solution, the interpretation computed for the predicate *allocated* represents the output WAF allocation scheme, as it allows to understand in which APs of the logical topology WAFs must be allocated. Instead, the interpretation computed for the predicate *configured* and the values of the free variables composing the placeholder rules provide the information required for establishing the output WAF rule sets.

The two outputs, i.e., the WAF allocation scheme and rules, initially expressed in the language of the employed MaxSMT solver (e.g., SMT-LIB), are then exposed in a medium-level language (e.g., with JSON or XML-based files, reporting the information in a structured format), easily comprehensible also for human administrators, and at the same time independent from the technical terms required for the specific low-level configuration of software products implementing WAF functionalities.

C. Low-level WAF Configuration Generation

As the structured format of the two outputs of the previous phase is independent from any specific vendor technology,

it cannot be directly deployed in real infrastructures. As a consequence, without an additional transformation step, the overall process would not be completely automatic and would still require manual intervention by a security administrator to translate the generated configuration into the concrete syntax required by the selected WAF products and virtualization platforms.

For this reason, the last phase of the proposed approach consists of a language conversion module, which automatically translates the medium-level outputs into low-level configuration artifacts ready to be deployed. This module has a twofold objective: (i) to materialize the allocation scheme into executable deployment scripts for virtualized environments; (ii) to convert the structured rule representation into the specific configuration syntax required by the adopted WAF technologies (e.g., ModSecurity, Squid, pfSense).

Concerning the allocation scheme, the conversion module generates configuration files and scripts tailored to virtualized infrastructures, such as Docker-based environments. In particular, for each AP where the predicate *allocated* evaluates to true in the optimal solution, the system produces the necessary artifacts to instantiate a corresponding WAF container. These artifacts may include Dockerfiles, docker-compose descriptors, and auxiliary shell scripts that orchestrators can automatically execute.

Regarding the filtering rules, the conversion module translates the medium-level XML/JSON representation into the concrete syntax required by the target WAF implementation. Since different products adopt heterogeneous configuration languages, this step is technology-dependent but conceptually straightforward: each placeholder rule instantiated by the solver is mapped to the corresponding low-level directive. For example, a WAF rule represented in JSON format as:

```
{
 "action": "deny",
 "IPSrc": "192.0.2.*",
 "IPDst": "198.51.100.10",
 "HTTPMethod": "POST",
 "URL": "/admin"
}
```

can be translated into a ModSecurity rule as follows:

```
SecRule REMOTE_ADDR "@ipMatch 192.0.2.0/24"
"phase:2,deny,status:403,id:10001,chain"
SecRule REQUEST_METHOD "@streq POST" "chain"
SecRule REQUEST_URI "@streq /admin"
```

As a final result, the entire workflow, from high-level policy specification to the activation of concrete WAF instances with their rule sets, becomes fully automatic, ensuring practical applicability in virtualized environments.

IV. Implementation and Validation

A Java-based framework was developed to implement the proposed approach. This implementation interfaces with the MaxSMT solver Z3, which is an open-source state-of-the-art theorem prover by Microsoft Research [30]. The medium-level language chosen to represent both input NSPs and the WAF configuration output by the second phase of the approach is the Medium Security Policy Language (MSPL), a well-known language based on XML that has been used and validated in multiple EU-funded projects such as SECURED[1] and ANASTACIA[2]. The language converter functionality included in this framework can translate WAF MSPL-based rules into low-level configuration settings for Suricata and Squid. However, the design of the framework has been defined to be easily extended to support other WAF products. Moreover, REST APIs were developed for this framework, so that it can interact with human users and virtual network orchestrators.

The framework underwent both effectiveness validation and scalability evaluation. All tests were carried out on a 4-core Intel i7-6700 3.40 GHz workstation, equipped with 32 GB RAM.

First, the framework was validated in a Docker-based environment managed by Docker Compose. The virtual network previously shown in Fig. 2 as AG example was reproduced, and a set of NSPs similar to those in TABLE I were defined. Some of them were also retrieved by the logs of the IDS software installed there, by using a tool developed by us that can parse log files produced by Snort3 and OSSEC3.7, two extensively adopted IDS implementations. The result produced by our framework was then automatically applied to the softwarized WAFs, and their correct behavior was assessed by trying communications that should be blocked or allowed.

Second, scalability tests were carried out, in order to assess the execution time of the framework when solving firewall configuration problems of varying size. These tests were based on use cases characterized by network topologies artificially synthesized as extensions of the one depicted in Fig. 2, and by NSP sets including policies of all three types. The use cases are characterized by increasing values of the two main parameters that may increase the complexity of the MaxSMT problem to be solved, i.e., the numbers of APs and NSPs. The test on each use case was repeated with 100 independent runs, and the average computation time was estimated across them. In those runs, the network topology and the set of NSPs remained unchanged, while only the IP addresses assigned to the nodes were modified. This validation strategy was adopted because the execution time of Z3 is known to be sensitive to changes in integer constants and variables. Since IP addresses are modeled as integers in our MaxSMT formulation, varying them could affect the solver's performance, so averaging the results of 100 runs was required to obtain stable and representative performance measurements.

In greater detail, the performance was evaluated by increasing the number of APs while keeping the NSP set fixed at 10, then by increasing the number of NSPs while keeping the APs fixed at 10, and finally by growing both parameters simultaneously. This approach allowed us to isolate the impact of each parameter and then obtain an overall view of their combined effect on scalability.

[1]https://www.secured-fp7.eu/

[2]http://anastacia-h2020.eu/

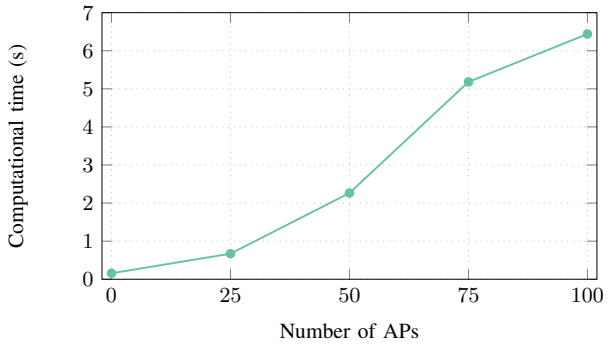

Fig. 3: Results of scalability tests for APs

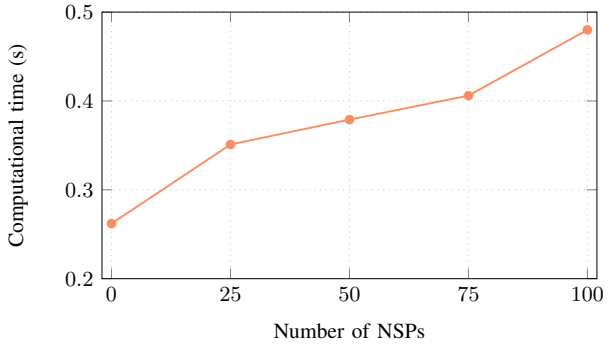

Fig. 4: Results of scalability tests for NSPs

Fig. 3 reports the computation time as the number of APs increases, while keeping the number of NSPs fixed to 10. The results show a gradual growth in execution time, which remains within a few seconds even for 100 APs. This trend confirms that the allocation dimension impacts the complexity of the MaxSMT problem, but the framework is able to scale efficiently with respect to the number of potential firewall placement points.

On the contrary, Fig. 4 illustrates the impact of increasing the number of NSPs while maintaining the number of APs constant at 10. In this case, the growth in computation time is even more moderate, remaining below one second across all tested configurations. This behavior indicates that the solver handles the enlargement of the policy set effectively, and that the constraint structure introduced by additional NSPs does not significantly degrade performance.

Finally, Fig. 5 shows the results obtained when both the number of APs and NSPs are increased proportionally. As expected, this scenario leads to a more evident rise in execution time, since both the allocation and rule-generation dimensions contribute simultaneously to the problem complexity. Nevertheless, the computation time remains bounded within a few seconds even for the largest tested instances, demonstrating the overall scalability and practical applicability of the developed framework.

Overall, the positive results achieved from the effectiveness and scalability validation tests demonstrate that the proposed approach can produce formally correct and optimized con-

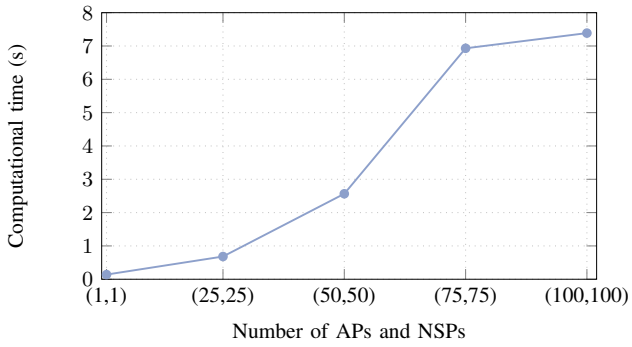

Fig. 5: Results of scalability tests for APs and NSPs proportionally increased

figurations within a few seconds. On the one hand, this achievement would be highly difficult to reach through manual configuration in complex virtualized environments. On the other hand, it was reached despite the theoretical worst-case computational complexity of MaxSMT problem resolution, thanks to the Z3 algorithms which are polynomial in the average case, and to a careful design of the formal models, so as to balance adherence to reality and impact to performance. In fact, the designed models capture all the characteristics of their real counterparts that could impact the solution's correctness, while excluding redundant details which would have been useless, so that they could be more lightweight and lead to higher scalability.

V. Conclusion and Future Work

This paper presented an automatic approach for WAF configuration, integrating formal verification and optimization to replace traditional manual strategies, which are error-prone and time-consuming. A partial weighted MaxSMT problem formulation was designed to jointly address the allocation of distributed WAF instances and the generation of their filtering rules, ensuring correctness by construction while minimizing resource usage. The experimental validation demonstrated both the effectiveness and the scalability of the proposed framework. The results show that optimized configurations can be computed within a few seconds even for large problem instances, whereas a manual approach would struggle to achieve the same level of correctness and optimality within comparable time constraints.

Future work will investigate the integration of additional classes of application-layer security controls, such as API gateways and advanced intrusion prevention systems, within the same formal framework. Another promising direction concerns the refinement of the optimization model, by introducing more expressive cost functions that capture performance overhead, latency impact, and resource consumption in heterogeneous virtual infrastructures. Finally, an extension of the policy specification language is planned, so as to support richer semantic constructs, enabling administrators to express complex security intents in an even more abstract and user-friendly manner.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101168144 (MIRANDA). The views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Cybersecurity Competence Centre. Neither the European Union nor the granting authority can be held responsible.

REFERENCES

[1] A. Alhuzali, R. Gjomemo, B. Eshete, and V. N. Venkatakrishnan, "NAVEX: precise and scalable exploit generation for dynamic web applications," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. USENIX Association, 2018, pp. 377–392. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/alhuzali

[2] 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. [Online]. Available: https://doi.org/10.1145/3616401

[3] D. Bringhenti and F. Valenza, "Greenshield-e2c: A sustainable energy-aware firewall configuration mechanism for edge-to-cloud continuum," *Comput. Networks*, vol. 282, p. 112279, 2026. [Online]. Available: https://doi.org/10.1016/j.comnet.2026.112279

[4] A. Mk, K. S. S. Bala, S. S. T. Sonti, and J. Kp, "An empirical study on the evaluation and enhancement of OWASP CRS (core rule set) in modsecurity," *Comput. Secur.*, vol. 160, p. 104714, 2026. [Online]. Available: https://doi.org/10.1016/j.cose.2025.104714

[5] D. Bringhenti, R. Sisto, and F. Valenza, "Automating VPN configuration in computer networks," *IEEE Trans. Dependable Secur. Comput.*, vol. 22, no. 1, pp. 561–578, 2025. [Online]. Available: https://doi.org/10.1109/TDSC.2024.3409073

[6] A. M. Zarca, J. B. Bernabé, A. F. Skarmeta, and J. M. A. Calero, "Virtual iot honeynets to mitigate cyberattacks in sdn/nfv-enabled iot networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1262–1277, 2020. [Online]. Available: https://doi.org/10.1109/JSAC.2020.2986621

[7] E. J. Scheid, C. C. Machado, M. F. Franco, R. L. dos Santos, R. J. Pfitscher, A. E. S. Filho, and L. Z. Granville, "Inspire: Integrated nfv-based intent refinement environment," in *Proc. of the IFIP/IEEE Symp. on Integrated Network and Service Management*, 2017. [Online]. Available: https://doi.org/10.23919/INM.2017.7987279

[8] Z. Hao, Z. Lin, and R. Li, "A SDN/NFV security protection architecture with a function composition algorithm based on trie," in *The 2nd International Conference on Computer Science and Application Engineering, Hohhot, China, October 22-24, 2018*. ACM, 2018, pp. 176:1–176:8. [Online]. Available: https://doi.org/10.1145/3207677.3277992

[9] Y. Liu, Y. Lu, W. Qiao, and X. Chen, "A dynamic composition mechanism of security service chaining oriented to sdn/nfv-enabled networks," *IEEE Access*, vol. 6, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2870601

[10] A. S. Sendi, Y. Jarraya, M. Pourzandi, and M. Cheriet, "Efficient provisioning of security service function chaining using network security defense patterns," *IEEE Trans. Services Comput.*, vol. 12, no. 4, 2019. [Online]. Available: https://doi.org/10.1109/TSC.2016.2616867

[11] 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. [Online]. Available: https://doi.org/10.1109/TPDS.2016.2585108

[12] Y. Bartal, A. J. Mayer, K. Nissim, and A. Wool, "*Firmato*: A novel firewall management toolkit," *ACM Trans. Comput. Syst.*, vol. 22, no. 4, 2004. [Online]. Available: https://doi.org/10.1145/1035582.1035583

[13] A. K. Bandara, A. C. Kakas, E. C. Lupu, and A. Russo, "Using argumentation logic for firewall configuration management," in *Proc. of the 11th IFIP/IEEE Intern. Symp. on Integrated Network Management*, 2009. [Online]. Available: https://doi.org/10.1109/INM.2009.5188808

[14] P. Adão, C. Bozzato, G. D. Rossi, R. Focardi, and F. L. Luccio, "Mignis: A semantic based tool for firewall configuration," in *Proc. of the IEEE 27th Computer Security Foundations Symp.*, 2014. [Online]. Available: https://doi.org/10.1109/CSF.2014.32

[15] P. Verma and A. Prakash, "FACE: A firewall analysis and configuration engine," in *Proc. of the IEEE/IPSJ Intern. Symp. on Applications and the Internet (SAINT05)*, 2005. [Online]. Available: https://doi.org/10.1109/SAINT.2005.28

[16] A. Sahu, P. Wlazlo, N. Gaudet, A. Goulart, E. Rogers, and K. Davis, "Generation of firewall configurations for a large scale synthetic power system," in *Proc. of the IEEE Texas Power and Energy Conference*. IEEE, 2022, pp. 1–6. [Online]. Available: https://doi.org/10.1109/TPEC54980.2022.9750776

[17] J. Govaerts, A. K. Bandara, and K. Curran, "A formal logic approach to firewall packet filtering analysis and generation," *Artif. Intell. Rev.*, vol. 29, no. 3-4, 2008. [Online]. Available: https://doi.org/10.1007/s10462-009-9147-0

[18] D. Ranathunga, M. Roughan, P. Kernick, and N. Falkner, "The mathematical foundations for mapping policies to network devices," in *Proc. of the 13th Intern. Joint Conf. on e-Business and Telecommunications*, 2016. [Online]. Available: https://doi.org/10.5220/0005946201970206

[19] N. Wintering, E. Lanfer, and N. Aschenbruck, "Automating network perimeter threat prevention for decentralized network administration," in *20th International Conference on Network and Service Manag., Prague, Czech Republic, October 28-31, 2024*. IEEE, 2024, pp. 1–7. [Online]. Available: https://doi.org/10.23919/CNSM62983.2024.10814436

[20] W. Zahwa, A. Lahmadi, M. Rusinowitch, and M. Ayadi, "In-network ACL rules placement using deep reinforcement learning," in *IEEE International Mediterranean Conference on Communications and Networking, MeditCom 2024, Madrid, Spain, July 8-11, 2024*. IEEE, 2024, pp. 341–346. [Online]. Available: https://doi.org/10.1109/MeditCom61057.2024.10621188

[21] M. Jiménez-Lázaro, J. Berrocal, and J. Galán-Jiménez, "Deep reinforcement learning based method for the rule placement problem in software-defined networks," in *2022 IEEE/IFIP Network Operations and Management Symposium, NOMS 2022, Budapest, Hungary, April 25-29, 2022*. IEEE, 2022, pp. 1–4. [Online]. Available: https://doi.org/10.1109/NOMS54207.2022.9789906

[22] N. Schnepf, R. Badonnel, A. Lahmadi, and S. Merz, "Rule-based synthesis of chains of security functions for software-defined networks," *ECEASST*, vol. 76, 2018. [Online]. Available: https://doi.org/10.14279/tuj.eceasst.76.1075

[23] C. Basile, F. Valenza, A. Lioy, D. R. Lopez, and A. P. Perales, "Adding support for automatic enforcement of security policies in NFV networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, 2019. [Online]. Available: https://doi.org/10.1109/TNET.2019.2895278

[24] C. Yang, X. Mi, Y. Ouyang, R. Dong, J. Guo, and M. Guizani, "SMART intent-driven network management," *IEEE Commun. Mag.*, vol. 61, no. 1, pp. 106–112, 2023. [Online]. Available: https://doi.org/10.1109/MCOM.002.2200119

[25] 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, pp. 218–230, 2010. [Online]. Available: https://doi.org/10.1109/TC.2009.172

[26] 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. [Online]. Available: https://doi.org/10.1109/TDSC.2022.3160293

[27] D. Bringhenti, S. Bussa, R. Sisto, and F. Valenza, "A two-fold traffic flow model for network security management," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 4, pp. 3740–3758, 2024. [Online]. Available: https://doi.org/10.1109/TNSM.2024.3407159

[28] S. Wu, N. Chen, A. Xiao, P. Zhang, C. Jiang, and W. Zhang, "Ai-empowered virtual network embedding: A comprehensive survey," *IEEE Commun. Surv. Tutorials*, vol. 27, no. 2, pp. 1395–1426, 2025. [Online]. Available: https://doi.org/10.1109/COMST.2024.3424533

[29] S. Tanzarella and M. Repetto, "Context discovery for digital service chain with openc2," in *11th IEEE International Conference on Network Softwarization, NetSoft 2025, Budapest, Hungary, June 23-27, 2025*. IEEE, 2025, pp. 579–584. [Online]. Available: https://doi.org/10.1109/NetSoft64993.2025.11080629

[30] L. M. de Moura and N. S. Bjørner, "Z3: an efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, Budapest, Hungary, March 29-April 6, 2008*, ser. Lecture Notes in Computer Science, vol. 4963. Springer, 2008, pp. 337–340. [Online]. Available: https://doi.org/10.1007/978-3-540-78800-3_24