Toward the Optimization of Automated VPN Configuration

Toward the Optimization of Automated VPN Configuration

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

***Abstract*—In recent years, VPNs have become one of the most essential security mechanisms, allowing the users to safely communicate over untrusted networks. As research in security automation advances, the literature has introduced various approaches for automating the configuration of security functions and addressing the growing challenges faced by security administrators, though only a limited number specifically address VPNs. An effective constraint programming-based approach in this field is VEREFOO, which leverages formal methods to automatically and optimally configure VPNs while ensuring formal correctness by construction. However, VEREFOO was not designed to minimize memory consumption and performance overhead, despite their relevance in both enterprise and commercial modern virtual networks. In this paper, the optimization aspect of the VEREFOO approach is enhanced and expanded on both of these new fronts. Specifically, new optimization strategies are designed to provide minimization of the configured rules and maximization of constraints generation efficiency. This optimized approach has been implemented as a framework and validated on a realistic use case to assess optimization improvements across multiple aspects.**

***Index Terms*—security automation, VPN, optimization**

I. Introduction

In recent years, the exponential rise in cyberattacks such as hijacking has made the Virtual Private Network (VPN) technology a fundamental component of network architectures, as it plays a crucial role in safeguarding user privacy and enabling secure communication over untrusted networks. Nowadays various VPN models exist, each enabling different security properties for network traffic, such as securing communication with an end-to-end VPN, encapsulating traffic in a tunnel with a site-to-site VPN, or providing secure access to services with a remote-access VPN [1]. The key architectural element common to all these VPN models is the Communication Protection System (CPS). Located at the VPN borders, it serves as the node or module actually responsible for applying (or removing) protection on the traffic.

The configuration and network allocation of Cyber-Physical Systems (CPSs) are critically important tasks, as they determine whether the desired security properties are enforced. If these tasks are not performed correctly and effectively, there is a risk of compromising traffic confidentiality and integrity, as well as significantly impacting network performance [2]. Historically, security administrators manually configured critical systems, a practice that remained feasible until recent years when new challenges emerged. In particular, the increasing heterogeneity of security controls—encompassing systems such as VPNs, firewalls, and NATs—along with the expanding size and complexity of network topologies, have rendered manual configuration progressively more difficult. These factors, combined with the inherent limitations of human administrators, contribute to making the manual management of VPNs highly error-prone and likely to result in sub-optimal security solutions, while attackers continuously become more skilled and aware of potential vulnerabilities to exploit. In response to these challenges, significant efforts have been dedicated to the automation of cybersecurity configuration tasks, with the aim of reducing human intervention and, consequently, the risk of anomalies arising from misconfigurations of security services. Automation also enables a more effective pursuit of optimization objectives, including enhancements in network and computing performance, lower memory consumption, and reduced computational overhead. Several approaches for the automatic configuration of network security functions have been proposed, particularly following the advent of innovative paradigms such as Software-Defined Networking (SDN) and Network Function Virtualization (NFV), which have profoundly transformed the networking field. For instance, the literature is rich with research concerning firewall configuration, where automation, optimization, and formal correctness are often successfully integrated [3] [4]. However, only a few studies focus specifically on VPN configuration, and even fewer attempt to combine optimization and formal verification within a unified framework.

One of the most promising approaches in this field is represented by VEREFOO (VErified REFinement and Optimized Orchestration) [5], an approach based on formal methods and rooted in constraint programming. This approach allocates and configures CPSs within the network in a fully automated and provably correct manner, ensuring compliance with the security policies defined by the administrator while simultaneously optimizing the number of CPSs deployed. In order to achieve this, it models the configuration problem as a Maximum Satisfiability Modulo Theories (MaxSMT) problem. However, VEREFOO still has severe shortcomings in terms of VPN configuration optimization. In fact, the only optimization goal that it pursues is the minimization of the number of allocated CPSs, but it is not enough to support

the requirements of modern networks related to memory consumption and computational overhead.

To address the limitations of the literature, this paper introduces new optimization strategies integrated into the VEREFOO approach to enhance automated VPN configuration in virtual networks along two dimensions: minimizing both the number of security rules configured in each CPS and the total number of constraints generated for the auto-configuration problem, resulting in maximized CPSs memory consumption efficiency and minimized performance overhead. The first objective is achieved by (i) generating a Security Association (SA) per Security Requirement (SR) rather than per traffic flow; (ii) avoiding rule configuration in a CPS unless strictly necessary; and (iii) promoting the use of wildcard values to cover the widest possible set of flows with a single rule. The second objective focuses on simplifying the MaxSMT problem formulation to reduce computational overhead and improve scalability, while preserving solution completeness.

The remainder of this paper is structured as follows. Section II discusses related work. Section III illustrates how the number of security rules configured has been optimized, and how the MaxSMT problem formulation has been streamlined. Section IV describes the implementation and validation of the contributions. Section V draws conclusions and discusses future work.

II. Related Work

The initial efforts spent by the research community on the automation of VPN configuration are represented by a first group of studies composed by [6], [7], [8] and [9]. In particular, [6] is one of the earliest papers to identify the limitations of manual VPN configuration and to recognize the significance of abstracting the VPN technology. There, the authors propose a VPN design aimed at satisfying the specifications provided by the customer, while simultaneously maximizing resource availability for the VPN Service Provider. However, their proposed approach simply consists in a brute-force algorithm that, as expected, performs poorly in practice and cannot be effectively applied to complex networks. The next studies of this initial group, i.e., [7], [8] and [9], evolve that early-stage research, trying to overcome some of its limitations concerning VPN configuration automation. Specifically, [7] introduces two algorithms, named Bundle Approach and Direct Approach, which represent possible trade-offs between configuration completeness and efficiency. In fact, the former guarantees completeness and correctness but it has limited scalability due to the computational overhead involved in combining traffic into disjoint bundles. Instead, the latter compromises on those two features in order to offer the scalability and efficiency lacking in the first approach. [8] continues this research line proposing a third and alternative algorithm, the Ordered-Split Approach, aimed at addressing the redundancy and quality issues affecting them, while [9] addresses the challenge of distributed policy management through an ad-hoc framework. Even if these studies represent the initial milestones of this research line, their proposed approaches are severely limited, as they are restricted to traditional computer networks and lack a substantial focus on optimization.

A next group of studies proposes solutions to the VPN configuration automation problem in non-traditional computer networks, such as the ones based on SDN and NFV. Despite the increasing importance of these cutting-edge technologies, a limited number of papers belongs to this class, i.e., [10], [11] and [12]. In greater detail, the paper [10] proposes a method that integrates SDN with IPsec to streamline the configuration and management of IPsec VPNs by using OpenFlow switches and controllers. The study described in [11] adopts a similar philosophy, presenting a solution for managing IPsec Security Associations using SDN, with the novel distinction of handling both IKE and IKE-less cases and reducing the complexity at the network resource level by centralizing key management within the SDN controller. Instead, [12] proposes a different approach, where the concepts of DevOps are applied to the resolution of the automated VPN configuration problem in an SDN-based environment, all while emphasizing emphasizes enhanced usability and manageability. Although applicable to modern next-generation networks, this group of papers still overlooks the optimization aspect and lacks the soundness and stability provided by a formal methods-based approach.

Finally, an approach that can be applied to virtual networks and successfully integrates automation, formal verification, and optimization is VEREFOO [13] [5]. This approach represents a significant contribution to the literature because, formulating the auto-configuration problem as MaxSMT, it can incorporate some optimization goals directly into the problem definition while ensuring correctness by construction of the produced solution. However, these goals are exclusively limited to minimizing the number of CPSs allocated in the network to maximize throughput and efficiency.

In view of this analysis of the related work, the optimization strategies proposed by this paper represent a step ahead in the literature on VPN configuration automation. In fact, they are suitable also for virtual networks, differently from the studies analyzed in the first group. Moreover, differently from the studies of the second class and VEREFOO, they aim at reducing the number of configured rules and the complexity of the problem formulation, with the significant objectives to minimize memory consumption and computational overhead.

III. Methodology and Approach

This paper enriches the methodology of VEREFOO, briefly recalled in Section III-A, embedding new optimization strategies that help to reduce resource consumption in computer networks and improve the operational performance of the CPSs. Specifically, these strategies aim at (i) minimizing the number of rules of each allocated CPS to make their configuration optimal, and (ii) reducing the number of constraints needed to define the MaxSMT auto-configuration problem, thereby streamlining the formal model and improving the scalability of the approach. Further details about how these two objectives are reached are discussed in Sections III-B and III-C, respectively.

A. Overview on VEREFOO approach

The approach pursued by the VEREFOO methodology is based on formal methods. In fact, it leverages constraint programming, so as to ensure formal correctness by construction and eliminate the need for a-posteriori formal verification of the solution produced.

First of all, the VEREFOO approach takes as input a Service Graph (SG), representing the network topology to be managed, and a set of Security Requirements (SRs), expressed in a medium-level language (i.e., a policy language that is devoid of vendor-dependent and implementation-dependent characteristics), which the security administrator would like to enforce. From this point onward, no human intervention is required anymore, as the methodology autonomously models the VPN architecture configuration problem as a MaxSMT problem, which is then solved by an SMT solver. Finally, it produces (i) an Allocation Scheme, which is the input SG enriched with information on the placement of CPSs, and (ii) the list of Security Associations (SAs) configured for each CPS to enforce the specified communication protection requirements.

The MaxSMT formulation was chosen over the traditional SMT one because it enables the integration of optimization objectives directly into the formal model through soft constraints. Unlike hard constraints, which must be fully satisfied to produce a valid solution, soft constraints are not strictly required to be met but are assigned weights, and the solver selects the solution that maximizes the sum of the weights of the satisfied soft constraints. This way, by expressing the optimization goals as soft constraints and by modeling the required security properties as hard constraints, the solution produced is guaranteed to be both formally correct and optimal.

All these constraints are based on formal models of the components involved in the approach. For the sake of conciseness, in the following, only the models related to traffic flows, SRs and CPS rules are recalled, because they are key concepts to understand the contributions of this paper. All other models can be found in [5].

A traffic flow *t* represents a class of packets and is formally defined as a tuple $t = (p^i, h^i, h^a)$, where p^i represents the internal payload, h^i denotes the original internal header, and h^a accounts for any additional header that a CPS may introduce. Each header is a conjunction of five predicates, one for each field of the IP 5-tuple *(IPsrc, IPdst, pSrc, pDst, tProto)*, where the *IPsrc* and *IPdst* predicates define source and destination IP conditions, *pSrc* and *pDst* predicates specify source and destination port conditions, and *tProto* indicates the transport-layer protocol (TCP or UDP). All possible traffic flows and their corresponding paths are computed for each given SR to which they are associated.

The SRs represent the communication protection proeprties to be enforced in the network. An SR is modeled as a tuple $s_r = (C, A^c, A^i, A^a, V, S, W)$. On the one hand, the model defines the policy conditions C, identifying the relevant traffic flows and expressed using the previously explained IP 5-tuple notation. On the other hand, it includes specifications (A^c, A^i, A^a) on the cipher algorithms to be used to enforce respecitvely confidentiality, integrity and authentication properties on the identified traffic (e.g., AES-128-CBC, HMAC-SHA-256, RSA, etc.), the required VPN protocols (V, with IPsec and TLS being the available options), the enforcement modes (S), and the trustworthiness and inspection requirements (W) of nodes and links for the identified communication.

The CPS rules define the actions a CPS must perform on specific packet classes. They are also called "placeholder rules" because they are represented by a set of free variables whose values are determined by the MaxSMT solver, meaning their actual usage in the final output is not known a-priori. A placeholder rule is modeled as a tuple *r = (C, a^c, a^i, a^a, v, S, m, act)*, where *C* defines the traffic conditions using the IP 5-tuple, a^c, a^i, and a^a specify the algorithms for confidentiality, integrity, and authentication, *v* represents the used VPN protocol, *S* indicates how the CPS enforces these properties, *m* determines whether tunneling is required, and *act* defines whether the CPS applies or removes the protection. When a rule is actually configured by the solver, it is translated in the final output into an SA, represented using the same medium-level language as the SRs.

B. Optimization of rule configuration

The first optimization objective introduced by this study is the minimization of the number of configured rules in the allocated CPS. This goal was achieved by following two complementary approaches, related to hard and soft constraints, respectively.

For what concerns hard constraints for rule configuration, they are used in the MaxSMT problem defined in VEREFOO to ensure that, if a CPS is assigned the task of adding or removing protection to or from a packet class, a corresponding rule is configured to apply or remove protection on the traffic flow. However, these constraints used to be enforced at a flow granularity, meaning that a separate rule was configured for each traffic flow crossing the allocated CPS, regardless of whether these flows were associated with the same SR. Consequently, such an approach led to the configuration of more rules than strictly necessary.

In order to avoid this problem, we decided to shift the granularity of these hard constraints from the traffic flow level to the SR level, they now ensure that the solver configures only one rule per SR in each allocated CPS. This change significantly reduces the number of SAs in the final output, especially in network topologies where the CPSs handle multiple traffic flows. In order to aid in understanding this improvement, Fig. 1 shows an example useful to assess the impact of the proposed optimization. In this example, there are two CPSs, which must handle five traffic flows derived from two SRs: CPS_1 must handle Flows 1, 2, and 3 derived from SR_1, while CPS_2 must handle Flow 3 from SR_1, Flows 4 and 5 from SR_2. The output of the original VEREFOO approach,

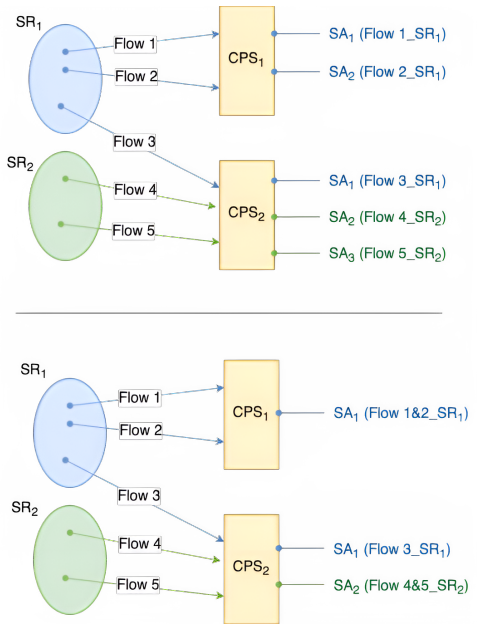

Fig. 1. Hard constraint granularity at flow (above) and SR (bottom) level

illustrated in the top section of Fig. 1, would have been that CPS_1 and CPS_2 are configured with a distinct SA for each traffic flow associated with SR_1 and SR_1. Instead, within the improved approach, a significant rule reduction is achieved, because CPS1 now configures a single SA to satisfy SR_1, while CPS_2 configures only two SAs, i.e., one for Flow 3 under SR_1 and another for SR_2, covering the remaining two flows, as illustrated in the bottom section of the figure.

For what concerns soft constraints, they represent a main strength of MaxSMT-based approaches, as they can straightly incorporate optimization objectives. Leveraging this feature, the solver can be guided to configure only the strictly necessary rules to enforce the required security properties and satisfy the SRs. To this end, two new sets of soft constraints have been introduced in our improved methodology, designed to work both in synergy with each other and alongside the existing hard constraints.

A set of soft constraints, introduced to reduce the number of configured rules, aims to prevent the solver from configuring rules in a CPS if not strictly necessary. To do so, the solver is guided not to configure rules at all (when possible, compatibly with the requested SRs) through the following soft constraint class:

$$\forall n \in N^A.\ \forall r \in R_n.\ \mathit{Soft}(\neg \mathit{configured}(r), w_{ncr}) \qquad (1)$$

where N^A is the set of the nodes where CPSs may be allocated, named Allocation Places, R_n is the set of placeholder rules of node $n \in N_A$, *configured* is the predicate that models the configuration decision of a rule by taking a boolean value reflecting the usage of that placeholder rule (*true* if r is actually used, false otherwhise), and w_{ncr} represents the weight associated with the soft constraint. Since it is a soft constraint, it does not conflict with the previously discussed hard constraints, ensuring that the solution space remains unaltered by its introduction. Instead, the solver can thus prioritize solutions that configure the fewest rules, maximizing the number of these soft clauses as much as possible.

Another set of soft constraints employed to further reduce the number of required rules is based on maximizing their applicability to the largest possible set of traffic managed by a CPS. Essentially, if a single rule can cover multiple traffic flows associated with different SRs, the total number of rules required to satisfy all the desired SRs is reduced, synergically working with the previous soft constraint set. In order to do so, a soft constraint expressing the preference of using wildcard values for the configuration of rules has been introduced. The wildcard is a special value indicating that all possible values are accepted for a specific field in a rule's condition. For example, when applied to IP addresses, a wildcard can indicate that any value is acceptable for part or all of the address, allowing the rule to match a broader range of traffic. Wildcard values have been established for each field of a rule condition, including IP addresses, ports, and the transport layer protocol. Ultimately, the resulting soft constraint is as follows:

$$\forall n \in N^A.\ \forall r \in R_n.\ \mathit{Soft}(r.C = *, w_w) \qquad (2)$$

where w_w represents the weights assigned to this soft constraint and the symbol * is the wildcard.

C. Optimization of constraints generation

The optimizations discussed so far, while significantly reducing the number of rules configured in each allocated CPS, come at the cost of increased computational overhead. In particular, the newly introduced soft constraints heavily impact the performance, since the solver must explore many more possible solutions before finding the most optimal one. To contain the overhead and maintain the scalability of the approach, a second type of optimization has been introduced, aiming at reducing the size of the formal model defined by the generated constraints.

A first contribution toward achieving this objective has been restricting the generation of hard constraints for enforcing security policies to only the nodes capable of hosting a CPS. Previously, the constraints ensuring that (i) traffic remains protected when crossing untrusted nodes or links, (ii) traffic is plain when reaching an inspector node to enable monitoring activities, and (iii) nodes support the required technology and algorithms to perform protection or unprotection actions were applied indiscriminately to all nodes along the traffic path, regardless of their role or actual capabilities. Although theoretically correct, this strategy generated a substantial number of constraints that did not influence the decisions of the solver regarding CPS allocation and configuration. For this reason, these constraints are now being applied only to end-host nodes and VPN gateways, still enabling the configuration of site-to-site, end-to-end, and remote-access VPN models while substantially reducing the total number of generated constraints. This leads to a more streamlined formal model for the solver and improved computational performance.

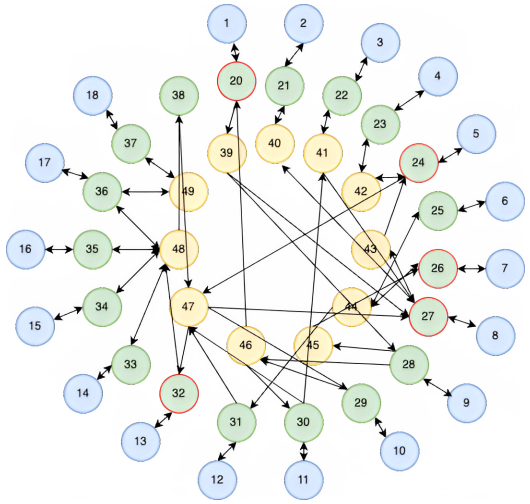

Fig. 2. Use case scenario

A second contribution to reducing the complexity of the auto-configuration problem is to minimize the number of placeholder rules that need to be defined and processed by the solver. Previously, for each Allocation Place, a placeholder rule was defined for each traffic flow associated with every SR. This used to result in a substantial number of placeholder rules being nearly identical, as their applicability conditions – source and destination IP addresses, source and destination ports, and L4 protocol – depend on the SR rather than the specific flow path. In order to reduce the number of placeholder rules that are later subjected to the hard and soft constraints, and thus complicating the MaxSMT problem resolution, a pruning function has been designed to determine the minimum set of placeholder rules required in each Allocation Place. The pruning algorithm is as straightforward as it is effective: for each CPS candidate node, it extracts the SRs associated with all traffic flows crossing that node. The minimum set of placeholder rules to be defined at that node then corresponds to the number of distinct extracted SRs.

IV. Implementation and Validation

The proposed approach has been implemented as a Java-based framework, building upon the original version of VEREFOO. The adopted MaxSMT solver is Z3 [14], an off-the-shelf SMT solver developed by Microsoft Research and known for its efficiency, versatility and free availability. The framework exposes a RESTful interface, with APIs that can be called by both users to get help in configuring their networks or by automated orchestration platforms. The input and output data exchanged through this interface can be represented in XML or JSON embedding.

To concretely demonstrate the discussed improvements, a use case scenario has been crafted, on which the framework has been evaluated through a machine equipped with a 12-core 12th Gen Intel i7-1255U CPU and 40 GB of RAM.

The use case scenario consists of a medium-sized network, realistically representing a typical enterprise infrastructure.

TABLE I
Table of Comparison

Metric	Original	Updated
Placeholder rules defined	60	40
SAs configured	24	8
Total constraints	2968	1400
Constraints on untrusted nodes	64	26
Constraints on inspector nodes	168	44
Constraints on support of required technology	1312	240

Fig. 2 illustrates the Service Graph, depicting 48 nodes divided into endpoints (colored in blue), forwarders (in green), and nodes without a predefined network function (in yellow), with directed arrows indicating their connectivity links. The Service Graph is provided as input to the framework, along with a list of 11 Security Requirements (SRs) specifying the untrusted nodes (i.e., the ones with red border of Fig. 2) where traffic must be protected by deploying CPSs on the network and configuring them with SAs.

The validation is performed on the described use case scenario by executing both the original VEREFOO implementation and the framework that we implemented to embed the new optimization strategies. Significantly different results are achieved across several other metrics: (i) total number of placeholder rules defined, (ii) total number of SAs configured, and (iii) total number of constraints generated. The last metric is further analyzed by breaking it down into (i) constraints ensuring traffic protection when crossing untrusted nodes, (ii) constraints allowing inspector nodes to analyze plain traffic, and (iii) constraints enforcing that nodes support the required technology to apply or remove protection on the traffic. The collected values are shown in TABLE I.

Starting with the number of placeholder rules defined, the impact of the pruning algorithm is evident, reducing the total by a third. Its effectiveness becomes more pronounced as the number of traffic flows associated with the same SR increases. For example, the Allocation Place with IP address 30.0.0.39 (represented by the node with index 39 in Fig. 2) is traversed by seven traffic flows, but since only three distinct SRs actually apply, the updated version of the framework defines just three rules for that node, whereas the original VEREFOO would have defined seven.

Moving to the next metric under evaluation, the optimization of rule configuration through the discussed hard constraints and the introduced soft constraints has led to a significant reduction of SAs configured. For instance, the number of SAs configured for the AP with IP address 30.0.0.40 (represented by the node with index 40 in Fig. 2) has decreased from six to just one, with the configured SA leveraging wildcard values to satisfy the two SRs involving the node.

The final aspect is the total number of generated constraints, whose reduction directly reflects the streamlining efforts concerning the formulation of the MaxSMT problem. Even with the introduction of multiple constraints per Allocation Place to optimize rule configuration, the total number of constraints has dropped by more than half with respect to the value

obtained with the original version of the framework. This is largely due to the significant optimization of constraints related to untrusted nodes, inspector nodes, and, most notably, the enforcement of technology support requirements.

Finally, for what concerns computation time, the updated framework only takes 3.4 s to find the optimal solution for the use case scenario. This achievement was possible thanks to the optimization strategies defined for constraints generation (i.e., the ones presented in Section III-C). In fact, without integrating those strategies aimed at streamlining the MaxSMT problem definition, expanding the VEREFOO framework only with the approaches for rule configuration optimization (i.e., the ones presented in Section III-B) would have led not to find a solution in finite time.

In conclusion, the validation carried out on the use case showcases that the strategies proposed in this paper successfully achieves the objective about memory consumption and CPS computational overhead, while getting performance results in line with the requirements of modern virtual networks.

V. Conclusions and Future Work

The approach presented in this paper enhances VPN configuration automation by introducing novel optimization strategies. Specifically, the proposed optimizations aim to minimize memory consumption and performance overhead in Communication Protection Systems by reducing the number of configured rules required to enforce the specified Security Requirements. Furthermore, to ensure the approach remains efficient and scalable, the formulation of the auto-configuration problem as a MaxSMT problem has been optimized without compromising solution correctness or completeness. The efficacy of the approach has been validated through a realistic use case scenario, which clearly demonstrated the substantial improvements achieved in both optimization aspects.

As future work, the optimized approach could be extended to jointly configure VPNs alongside other security functions, such as firewalls and Intrusion Detection Systems, to achieve full automation of the entire cybersecurity management process while ensuring correctness and optimization. Another promising direction could involve evaluating the discussed optimizations from a power consumption perspective and further refining them to enhance energy efficiency, ultimately making the approach more sustainable and environmentally friendly.

Acknowledgment

This work was supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the EU - NextGenerationEU.

References

[1] J. Li, B. Feng, and H. Zheng, "A survey on VPN: taxonomy, roles, trends and future directions," *Comput. Networks*, vol. 257, p. 110964, 2025. [Online]. Available: https://doi.org/10.1016/j.comnet.2024.110964

[2] H. Abbas, N. Emmanuel, M. F. Amjad, T. Yaqoob, M. Atiquzzaman, Z. Iqbal, N. Shafqat, W. B. Shahid, A. Tanveer, and U. Ashfaq, "Security assessment and evaluation of vpns: A comprehensive survey," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 273:1–273:47, 2023. [Online]. Available: https://doi.org/10.1145/3579162

[3] D. Bringhenti and F. Valenza, "Greenshield: Optimizing firewall configuration for sustainable networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 6, pp. 6909–6923, 2024. [Online]. Available: https://doi.org/10.1109/TNSM.2024.3452150

[4] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "Introducing programmability and automation in the synthesis of virtual firewall rules," in *Proc. of the 6th IEEE Conference on Network Softwarization, NetSoft 2020, Ghent, Belgium, June 29 - July 3, 2020*, 2020, pp. 473–478. [Online]. Available: https://doi.org/10.1109/NetSoft48620.2020.9165434

[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] R. Isaacs, "Lightweight, dynamic and programmable virtual private networks," in *Proc. of the 2000 IEEE Third Conference on Open Architectures and Network Programming*, 2000, pp. 3–12. [Online]. Available: https://doi.org/10.1109/OPNARC.2000.828128

[7] Z. Fu and S. F. Wu, "Automatic generation of ipsec/vpn security policies in an intra-domain environment," in *Proc. of the Operations & Management, 12th International Workshop on Distributed Systems, DSOM 2001, Nancy, France, October 15-17, 2001*, O. Festor and A. Pras, Eds., 2001, pp. 279–290. [Online]. Available: http://www.simpleweb.org/ifip/Conferences/DSOM/2001/DSOM2001/proceedings/S8-3.pdf

[8] Y. Yang, C. U. Martel, and S. F. Wu, "On building the minimum number of tunnels: an ordered-split approach to manage ipsec/vpn policies," in *Proc. of Managing Next Generation Convergence Networks and Services, IEEE/IFIP Network Operations and Management Symposium, NOMS 2004, Seoul, Korea, 19-23 April 2004*, 2004, pp. 277–290. [Online]. Available: https://doi.org/10.1109/NOMS.2004.1317665

[9] Y. Yang, Z. J. Fu, and S. F. Wu, "BANDS: an inter-domain internet security policy management system for ipsec/vpn," in *Proc. of Integrated Network Management VII, Managing It All, IFIP/IEEE Eighth International Symposium on Integrated Network Management (IM 2003), March 24-28, 2003, Colorado Springs, USA*, vol. 246, 2003, pp. 231–244. [Online]. Available: https://doi.org/10.1109/INM.2003.1194183

[10] Y. Li and J. Mao, "Sdn-based access authentication and automatic configuration for ipsec," in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 01, 2015, pp. 996–999. [Online]. Available: https://doi.org/10.1109/ICCSNT.2015.7490904

[11] G. L. Millán, R. M. López, and F. Pereñíguez-Garcia, "Towards a standard sdn-based ipsec management framework," *Comput. Stand. Interfaces*, vol. 66, 2019. [Online]. Available: https://doi.org/10.1016/j.csi.2019.103357

[12] L. Firdaouss, A. Bahnasse, B. Manal, and Y. Ikrame, "Automated VPN configuration using devops," in *Proc. of the 12th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2021) / the 11th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2021), Leuven, Belgium, November 1-4, 2021*, ser. Procedia Computer Science, vol. 198. Elsevier, 2021, pp. 632–637. [Online]. Available: https://doi.org/10.1016/j.procs.2021.12.298

[13] D. Bringhenti, G. Marchetto, R. Sisto, and F. Valenza, "Short paper: Automatic configuration for an optimal channel protection in virtualized networks," in *Proc. of CYSARM@CCS '20: the 2nd Workshop on Cyber-Security Arms Race, Virtual Event, USA, November, 2020*. ACM, 2020, pp. 25–30. [Online]. Available: https://doi.org/10.1145/3411505.3418439

[14] L. M. de Moura and N. S. Bjørner, "Z3: an efficient SMT solver," in *Proc. of Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008*, 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