Towards Security Automation in Virtual Networks

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

***Abstract*—Nowadays virtual computer networks are characterized by high dynamism and complexity. However, these features made the traditional manual approaches for network security management error-prone, unoptimized and time-consuming. This paper discusses the research carried out during my Ph.D. program on network security automation. In particular, it presents an approach based on constraint programming that combines automation, formal verification, and optimization for network security management. This approach has been proved to be general enough by means of multiple applications that have been developed. In particular, this paper describes VEREFOO, a framework for the automatic configuration of security functions, and FATO, a framework for the automatic orchestration of security transients. This methodology is extensively evaluated using different metrics and tests, and it has been compared to state-of-the-art solutions and to the requirements of dynamic virtual networks.**

***Index Terms*—network security, network virtualization, automation, formal verification, optimization**

I. Introduction and Motivation

Computer networks have been undergoing an incessant evolution since the beginning of the last decade. Network softwarization, declined in technologies such as *Software-Defined Networking* (SDN) and *Network Functions Virtualization* (NFV), simplified the network management operations. As a consequence, the size of modern computer networks is constantly increasing, because of the tendency to virtualize every activity and give it access to a network. Besides, the employed functions are becoming more heterogenous among them, and they are usually more complex than the old corresponding physical middleboxes. All these trends are confirmed by the characteristics of modern industrial networks, of the emerging *Internet of Things* (IoT) paradigm [1], and of Time-Sensitive SDN [2].

Nevertheless, the introduction of new advantages is typically accompanied by the presence of inevitable drawbacks. As computer networks are becoming bigger and more complex, new opportunities have arisen for cyber attackers to intrude on them, and the number of breaches dramatically increased. Unfortunately, the aforementioned agility and dynamism could not directly benefit network security management. The main reason is that security management is an activity that traditionally used to be performed manually with a trial-and-error approach. Administrators used to configure *Network Security Functions* (NSFs) such as firewalls or intrusion detection systems according to their initial expectation of possible attacks. If later a cyber attack had occurred, they would have simply modified the behavior of the function that could not block it, so as to avoid a possible repetition. However, such an approach could work only with small-sized networks, where everything was almost static and under the direct control of a human user, and where all the network accesses could be easily known. Instead, softwarized networks have opposite characteristics, i.e., big size, heterogeneity, dynamicity, and complexity [3].

For these reasons, security automation has been proposed as a possible solution to this urgent pending problem. The idea is that, if human users cannot cope with the security management of the whole network by themselves, they can be assisted by automated tools or frameworks, in charge of replacing the traditional manual security operations. In this context, the *Policy-Based Management* (PBM) paradigm [4] is a popular approach for automatic security. In a PBM-based approach, administrators should simply define the security requirements by means of a set of business-level statements, the security policies, which are commonly expressed in natural languages, so as to guarantee high usability and user-friendliness. Then, the policies are automatically transformed into low-level security management operations (e.g., the decision of where NSFs should be allocated in the network, and where, and the generation of their low-level configuration) through an operation named policy refinement.

Even if automation brings over great opportunities for network security, the state-of-the-art approaches that have been proposed in literature still have several shortcomings. In particular, two features that could enhance security automation but that are rarely included are formal verification and optimization. On the one hand, providing formal assurance that the results computed by an automated tool are really correct may be essential for the security of safety-critical systems. On the other hand, optimizing those same results may improve network security efficiency (e.g., if a firewall has only the minimum number of rules that are really required to enforce all the security policies specified by the security manager, then its filtering operations take less time because fewer comparisons between rule conditions and packet fields need to be performed). Despite the relevance of these two features, several state-of-the-art approaches do not exploit them, because their introduction is considered too challenging for keeping good performance.

In light of these motivations, during my Ph.D. program I have faced the challenge of investigating and formulating automated, fast, and provably correct techniques for network security management, with the final aim of improving the dependability and resilience of next-generation computer networks to cyber attacks. In fact, a central objective of the program has been to propose the first security management approach in literature to combine full automation, formal verification and optimization. In the definition of such an approach, a first challenge has been to define formal models of modern virtualized networks, such that they capture all the required information for automated security management, without impacting on performance excessively. Another challenge has been pursuing "security by construction" by means of lightweight correctness-by-construction approaches, where automated solvers can find a solution to the security management problem that does not require a traditional a-posteriori formal verification step, and, at the same time, fulfilling optimality criteria (e.g., to improve the efficiency of the security operations and to minimize resource consumption).

The remainder of this paper is structured as follows. Section II discusses the state of the art of network security automation, highlighting its limitations. Section III proposes a novel approach based on constraint programming to automate network security management. Section IV describes how some applications of this approach have been implemented and validated. Finally, Section V outlines the most relevant outcomes and discusses future work.

II. State of the Art

In literature, there are no approaches that combine automation, formal verification and optimization for network security management operations, such as configuration and orchestration.

Network security configuration comprises two main tasks: security service composition and function rule set definition. However, automatic approaches have usually been investigated for these tasks separately. On the one hand, [5], [6] just investigate how a security service can be designed automatically in an SDN-based network, whereas [7], [8] address that problem for NFV environments. Even if sometimes optimization criteria related to networking are embedded in some of these studies,they are not paired with security-oriented criteria. On the other hand, other studies just deal with establishing a configuration for specific NSF types, e.g., firewalls [9]–[11], VPN gateways [12], SDN switches [13]. There, even if formal verification is paired with automation more often than for security service composition methodologies, optimization is instead usually neglected. The only two state-of-the-art approaches which introduce automation for both tasks are [14], [15], but they have several limitations. Both of them can only design service function chains, even though the topology of modern virtualized computer networks is commonly a ramified graph. Moreover, the approach described in [14] is just designed for Android applications connected to SDN-based networks. Instead, [15] proposes optimization techniques that overlook security-oriented objectives, and that do not provide formal correctness assurance.

Network security orchestration comprises a large number of sub-tasks, such as NSF selection, their deployment, and mitigation of cyber attacks. A relevant problem related to attack mitigation is guaranteeing that connectivity policies (i.e., isolation and reachability policies) are still valid during the reconfiguration of a distributed NSF, required to face an on-going attack. However, approaches for automating the process of orchestrating the reconfiguration transient have been investigated only for distributed SDN switch architectures, e.g., [16]–[18]. However, these studies address security issues concerning the violation of connectivity policies only partially, because the configuration of SDN switches is mainly defined to address networking issues with respect to firewalls. They also overlook the impact that the behavior of other networks or security functions, which are present in the network, may cause to the reconfiguration transient. Moreover, optimization criteria (e.g., maximization of the secure transient states depending on the importance of each connectivity policy) should be enforced as well. Therefore, focusing exclusively on SDN switches is a limitation that should be overcome by addressing the transient management problem for more general distributed packet filtering firewalls. Integrating formal verification and optimization would also enhance the automatic techniques that may be proposed to address this problem.

III. The Proposed Approach

Combining automation, formal verification and optimization for network security management has been possible by pursuing approaches based on mathematical constraint programming. In particular, the security configuration and orchestration problems have been formulated by means of *Maximum Satisfiability Modulo Theories* (MaxSMT) problems. Differently from traditional *Satisfiability* (SAT) problems, the language used for the formulation of MaxSMT problems is the first-order logic, which includes the boolean operations as a specific case, but it can use several other theories, such as theories of real numbers, integers, lists, arrays, bit vectors and many other data structures. As such, a MaxSMT problem is composed of a set of predicates, where each predicate is a binary function defined over non-binary variables. Consequently, its language is much richer than the SAT language, and it allows to express more complex models.

A particular MaxSMT version that has been employed is the partial weighted one. This version is characterized by two kinds of constraints. On the one hand, some clauses named hard constraints always require satisfaction so as to achieve a correct solution to the problem. On the other hand, some clauses named soft constraints are given a weight and they do not strictly require satisfaction. Indeed, when solving a partial weighted MaxSMT problem, the goal is to find an assignment of the variables that satisfies all hard constraints, and that maximizes the sum of the weights assigned to the satisfied soft constraints.

The partial weighted MaxSMT formulation is key to jointly achieve all the three main objectives of full automation, optimization, and formal correctness for network security management. Full automation is achieved because a MaxSMT problem can be solved without human intervention, except for the input specification. Optimization can be achieved by expressing the optimization objectives by means of soft constraints, and formal correctness can be achieved by expressing the formal correctness requirements as hard constraints. Adopting this formal correctness-by-construction approach is beneficial not only because it improves the assurance and confidence that the computed solution is correct, but also because it avoids performing a-posteriori formal verification. Indeed, the solution can already be considered formally correct as far as all problem components are correctly modeled, being fundamental that such models capture all the information that may influence the correctness of the solution. Specifically, such models must capture both the security requirements and the forwarding behavior of the network where they must be enforced. At the same time, the number and complexity of constraints in the MaxSMT problem must be kept limited, in order to make the approach scalable. For all the above reasons, the modeling of the problem components, when formulating the MaxSMT problem, represented a big challenge.

Such approach based on constraint programming has been leveraged during the Ph.D. program to solve multiple problems related to network security management, such as NSF configuration and security transient orchestration. Here, I will present two relevant applications of this approach, respectively named VEREFOO and FATO, to address the two above mentioned problems.

A. VEREFOO

VErified REFinement and Optimized Orchestration (VEREFOO) is the first approach in literature to combine automation, formal verification and optimization to simultaneously solve the allocation and configuration problems for NSFs in virtual computer networks. The VEREFOO approach follows a Policy-Based Management paradigm, and therefore it works as illustrated in Fig. 1.

First, it requires the specification of two inputs: a *Service Graph* (SG) and a set of *Network Security Policies* (NSPs). An SG is the logical topology of a virtual network, i.e., an interconnection of service functions and network nodes providing a complete end-to-end network service. It represents a generalization of a Service Function Chain because the functions can be organized within a complex architecture where the traffic can flow through alternative paths. The SG provided by the user is automatically processed to create an internal representation called *Allocation Graph* (AG). For each link between any pair of network nodes or functions, a placeholder element, called *Allocation Place* (AP), is generated. Each AP represents a possible position for the allocation of an NSF instance. Instead, the NSPs describe the security requirements that must be enforced in the network. The user of the VEREFOO approach can specify them with a medium-level language, which abstracts from the vendor-dependent characteristics of the NSF implementations.

After receiving the AG and the NSPs, VEREFOO builds a MaxSMT problem representing the security allocation and configuration problem to be automatically solved. On the one hand, hard constraints are used to formally express the behavior of network functions composing the SG, the way traffic

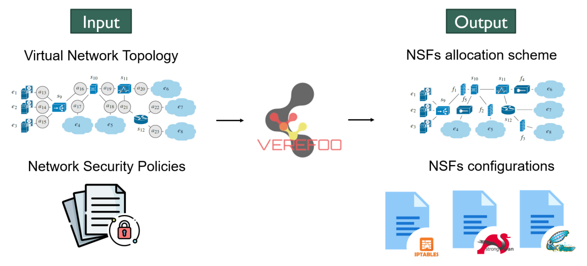


Fig. 1: The VEREFOO approach

flows can cross the network, and the required satisfaction of all NSPs. On the other hand, soft constraints are used to express two main optimization objectives, i.e., the minimization of the number of allocated NSFs and of configured rules. Then, an automated MaxSMT solver is fed with all the constraints, and it searches for an optimal correct solution.

In case of positive outcome, the provided result is composed of (i) the allocation scheme of the NSFs in the input SG; (ii) the configuration of each allocated NSF. The NSF allocation scheme specifies the APs where each NSF has to be allocated, as it can be seen in the example reported in Fig. 1. The configuration of each allocated NSF specifies its configuration rules (e.g., the filtering rules for a firewalls, or the communication protection rules for a VPN gateway). The allocation scheme contains the minimum number of NSFs required to enforce all NSPs, so minimizing resource consumption, while the configuration of each allocated NSF contains the minimum number of configured rules, thus minimizing the amount of memory needed to store them and maximizing the NSF performance. The allocation scheme is only generated at the logical abstraction level represented by the SG, as the output solution can be later deployed automatically into the virtual network by means of existing technologies. Instead, if no solution to the problem can be found, a non-enforceability report is generated for the user, who can try to guess why it has not been possible to enforce the NSPs.

The VEREFOO approach is designed to be a general method, which can be applied to any NSF type. As examples, it has been successfully applied for the configuration of packet filtering firewalls [19], [20], VPN gateways [21], SDN switches [22], and smart home devices [23]. Each application tackles with specific problems related to the corresponding environment, e.g., the management of blacklisting and whitelisting approaches for packet filters, the existence of multiple technologies and protocols for VPNs, and bandwidth and latency requirements in IoT-based SDN networks. Moreover, a peculiar application of VEREFOO is to employ this approach just to formally verify an already existing network security configuration [24].

B. FATO

FirewAll Transients Optimizer (FATO) is the first approach in literature to combine automation, formal verification and optimization to orchestrate a reconfiguration transient for virtual distributed packet filtering firewalls [25].

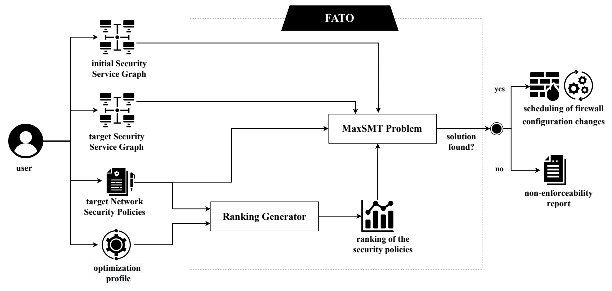
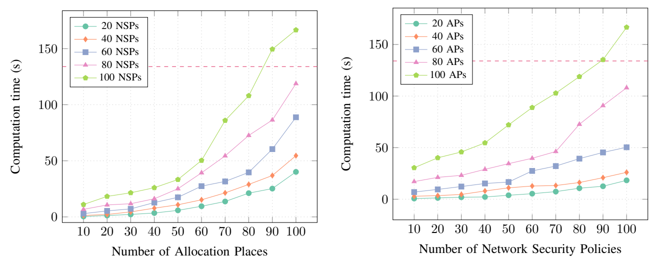

FATO
initial Security Service Graph
target Security Service Graph
target Network Security Policies
optimization profile
user
MaxSMT Problem
Ranking Generator
ranking of the security policies
solution found?
yes
no
scheduling of firewall configuration changes
non-enforceability report

Fig. 2: The FATO approach



(a) Scalability versus APs (b) Scalability versus NSPs

Fig. 3: VEREFOO Time Scalability

When a new firewall configuration is computed, it differs from the initial configuration for at least one of the two management aspects: the allocation scheme might have been changed (e.g., a new firewall instance has been introduced, or an existing one has been removed), or the firewall rules might have been adjusted to be compliant with new connectivity policies. Therefore, the security service must be updated accordingly, by applying a series of operations of different types: deployment of a new virtual firewall, removal of an existing firewall, update of the filtering rules of a firewall, deviation of a traffic flow. The firewall reconfiguration transient consists of a specific ordering of these operations, so that the global configuration is changed from the initial state to the target one. FATO aims to compute automatically the scheduling of these operations that minimizes the number of intermediate unsecure states where some security policies are not satisfied.

In order to accomplish this objective, the FATO approach works as illustrated in Fig. 2.

First, it requires the specifications of four inputs. The first and second inputs are the initial and target security SGs, which respectively include the description of the initial configuration of the distributed firewall (i.e., the start state of the reconfiguration transient) and target configuration of the distributed firewall (i.e., the final state of the reconfiguration transient). The second input is a set of target connectivity policies expressing the requirements that must be satisfied by the target configuration, defining which traffic flow must reach their destination, and which ones must instead be blocked. The fourth input is an optimization profile, which represents a compact indication about the relative priority of the connectivity policies. For example, the user may request that the isolation policies must have higher priority than the reachability policies (*security-max* profile) or vice versa (*service-max* profile). They may also specify other optimization objectives in additional to the basic one, e.g., to maximize the number of policies that are satisfied in each intermediate state (*policy-max* profile).

After receiving these inputs, from the specification of the optimization profile, FATO defines a ranking for the input policies (with the exclusion of the persistent policies, because they must be enforced in any intermediate state), as it comes handy for the definition of the optimization problem. Then, the initial and target security SGs with the firewall configurations, the target policies and their ranking are used by FATO to formulate a MaxSMT problem. After solving this optimization problem, FATO identifies the optimal order of reconfiguration changes, in such a way that the optimization fulfills the criteria derived from the ranking. This scheduling can be followed by a human who manages the virtual network, or a state-of-the-art orchestrator can exploit it to perform the required actions.

IV. Implementation and Validation

Both the VEREFOO and FATO approaches have been implemented by means of Java frameworks, which exploit the APIs offered by the open-source Z3 solver by Microsoft Research to formulate and solve the MaxSMT problem. The frameworks are accessible through its REST APIs, so that they can be exploited by external tools as a component of a more complex architecture, or through their GUI for human users. Besides, the code of the application of the VEREFOO approach to firewall configuration is already publicly available at the following link: https://github.com/netgroup-polito/verefoo/tree/Budapest.

A series of validation tests have been carried out on an 8-core Intel Core i7-10700E CPU @ 2.90GHz workstation with 32 GB RAM to assess all the features provided by these "security by construction" approaches: scalability, optimization, formal correctness. Here, due to space limitation, the most relevant results only about time scalability are presented.

1) VEREFOO Time Scalability: The charts in Fig. 3 present the results of tests performed to evaluate the time scalability of the VEREFOO approach, when applied to firewall configuration, versus number of APs and NSRs. For each test case with a given number of APs and NSRs, 100 runs have been executed. Fig. 3a and Fig. 3b show the average computation time of each test case. From these two charts, the most important result is that, even though the MaxSMT problem belongs to the NP-complete class in terms of computational complexity, the computation time does not increase exponentially. According to such results, the framework can manage AGs with up to 100 APs and 100 NSRs in less than 200 seconds. This result can be motivated by three reasons. First, NP-completeness only implies exponential time for the worst case, but the actual time for solving a MaxSMT instance is often less than the worst case time, also depending on which theories are used in the formulas [26]. Second, formal models have been defined so as to capture all the required

Approach	Alloc.	Config.	Formal	Optimal	Scalability
[7]	✓(SG)	X	✓	✓	20FW - 80s
[15]	✓(SFC)	✓	X	X	20FW - 4s
[9]	X	✓	✓	X	No Info
[10]	X	✓	✓	X	5FW - 50s
[14]	✓(SFC)	✓	✓	X	No Info
[8]	✓(SG)	X	X	✓	60FW
VEREFOO	✓(SG)	✓	✓	✓	100FW - 90s

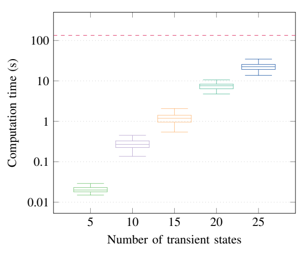
TABLE I: Comparison with most related approaches

aspects, but avoiding excessive complexity in the actual SMT problem to be solved (e.g., avoiding redundancy in variables and constraints, avoiding quantifiers). Leveraging this trade-off between expressiveness and complexity was a key factor that enabled the achievement of such scalability results. Third, state-of-the-art solvers like Z3 employ internal strategies that are quite efficient in exploring the solution space.

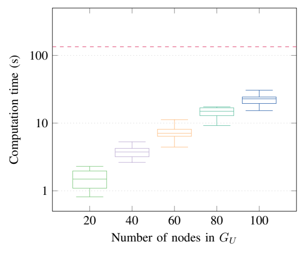
TABLE I shows a comparison of the VEREFOO approach with the most related state-of-the-art approaches available in the literature. The table confirms that no other existing approach jointly computes the firewall allocation scheme and the configuration starting from a provided SG, as the VEREFOO approach does. Also, no prior work achieves all the three features of full automation, optimization, and formal correctness, with the exception of [7], which, however, supports only the automatic generation of the firewall allocation scheme. Therefore, looking at the " Scalability" column, the VEREFOO framework proves to be competitive with respect to the other relevant works in terms of scalability, especially considering the added value of the results achieved.

Besides, in Fig. 3a and 3b, a baseline (red dotted horizontal line) is introduced, in order to have a reference: it is the Deployment Process Delay (DPD) introduced by a well known orchestrator (Open Source MANO) for deployment. DPD is the time the orchestrator takes to deploy and instantiate a VNF within an already booted VM and setup an operational network service. According to [27], this time is 134ms. The figures of these experiments show that the time taken by the framework to automatically allocate and configure firewalls in SGs with up to 100 APs and with up to 80 NSRs does not exceed the DPD, so being acceptable even in highly dynamic situations.

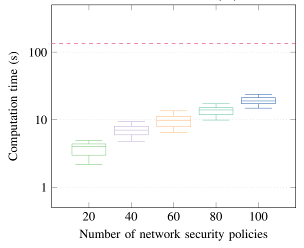
2) FATO Time Scalability: Fig. 4 reports the results of time scalability tests for the FATO approach. First, Fig. 4a analyzes the performance of the implementation when the number of transient states progressively increases. For those tests, each scenario characterized by a certain number of transient states is based on a topology of corresponding size (e.g., when the number of states is 20, in the security SG the number of firewalls subject to configuration changes is 20). The enforcement of a congruent number of network security policies is requested as well. Second, Fig. 4b analyzes how the framework behaves for increasing sizes of the network on which it is applied, while keeping the number of transient states fixed to 20 and the number of policies fixed to 50. Third, Fig. 4c evaluates scalability versus the number of network



(a) Transient states



(b) Network nodes



(c) Security policies

Fig. 4: FATO Time Scalability

security policies that should be enforced in the reconfiguration transient, while keeping the number of transient states fixed to 20 and the network size fixed to 50 nodes. The plotted results show that the proposed approach can successfully manage fairly big networks while checking the satisfaction of a large set of policies. Scalability with respect to network size and policy set cardinality is even better than scalability with respect to the number of transient states. This is due to the fact that the increment of soft constraints in the formulation of the MaxSMT problem is lower.

Time scalability is also in line with the times that are required by state-of-the-art approaches for performing management tasks related to a security reconfiguration. For example, [28] underlines that establishing the embedding scheme of 10 virtual functions on a physical network composed of 50 nodes can be up to 1400 s. [29] experimentally checked that the instantation time of a network security service takes more than 100 seconds, when the service is composed of around 30 virtual functions, for the Virtual Infrastructure Managers of both Open Source MANO and Openstack. Again, [27] states that DPD time related to the deployment of a single virtual function is 134s (reported as baseline). If these numbers are combined, the time introduced by the FATO framework does not represent a high delay, as the scheduling of the reconfiguration changes may be easily computed while another step, such as the service instantiation, is performed.

V. Outcomes and Future Work

This paper presented a possible approach, based on constraint programming, to automate network security management. This approach represents a major novelty in literature, as it is the first one to combine automation, formal verification and optimization. It is also general enough to allow applications related to multiple management operations, such as configuration and orchestration. In particular, VEREFOO has been defined to automatically allocate and configure NSFs in virtual networks, while minimizing their allocation scheme and configuration rule sets. Instead, FATO has been defined to automatically compute the optimal scheduling of the changes occurring in a security reconfiguration transient, so as to minimize the number of unsafe transitory states. In both cases, the formal models that have been defined for representing network components and security policies have been proved to represent a good trade-off between expressiveness and complexity, and this resulted into good performance and scalability of the frameworks implementing such approach.

Nonetheless, the proposed approaches still have limitations that may be overcome as future work. A first limitation is related to how the VEREFOO approach can solve the configuration problem. Currently, the VEREFOO approach can only work on a service graph devoid of network security functions, thus creating the security configuration from scratch even when it is not necessary, e.g., when a distributed firewall is already configured and only some of the user-specified security policies are modified. Therefore, a possible future research direction is the study of an optimized version of the VEREFOO approach, which can manage the reconfiguration of distributed security functions in an optimized way. Instead, another limitation is related to the performance of the MaxSMT formulation. Even if the validation of the VEREFOO and FATO approaches show that they can scales to large networks, it cannot manage the largest networks composed of tens of thousands nodes. Therefore, a heuristic algorithm will be investigated to be used as an alternative strategy.

Finally, other future work is planned to further contribute to the research area of network security automation. On the one hand, the MaxSMT formulation is flexible enough to be extended to support even other NSF types, such as web-application gateways, anti-spam filters, intrusion detection systems, stateful security functions and more. On the other hand, reaction and mitigation strategies can be investigated for a possible integration with this approach and network orchestrators, with the aim of making further steps in the direction of full autonomy in network security management.

References

[1] G. Mei, N. Xu, J. Qin, B. Wang, and P. Qi, "A survey of internet of things (iot) for geohazard prevention: Applications, technologies, and challenges," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4371–4386, 2020.

[2] D. Bringhenti and F. Valenza, "A twofold model for VNF embedding and time-sensitive network flow scheduling," *IEEE Access*, vol. 10, pp. 44 384–44 399, 2022.

[3] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "Sdn security: A survey," in *IEEE SDN for Future Networks and Services, SDN4FNS 2013, Trento, Italy, November 11-13, 2013*. IEEE, 2013, pp. 1–7.

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

[5] A. S. Jacobs, R. J. Pfitscher, R. A. Ferreira, and L. Z. Granville, "Refining network intents for self-driving networks," in *Proc. of the Workshop on Self-Driving Networks (SelfDN18)*, 2018.

[6] N. Schnepf, R. Badonnel, A. Lahmadi, and S. Merz, "Automated factorization of security chains in software-defined networks," in *Proc. of the IFIP/IEEE (INM19)*, 2019.

[7] 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.

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

[9] 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.

[10] 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.

[11] D. Ranathunga, M. Roughan, and H. X. Nguyen, "Verifiable policy-defined networking using metagraphs," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 1, 2022.

[12] L. Firdaouss, A. Bahnasse, B. Manal, and Y. Ikrame, "Automated VPN configuration using devops," in *Proc. of the Inter. Conf. on Emerging Ubiquitous Systems and Pervasive Networks*, 2021.

[13] A. Lara and B. Ramamurthy, "Opensec: Policy-based security using software-defined networking," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 1, 2016.

[14] 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.

[15] 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.

[16] J. Hua, X. Ge, and S. Zhong, "FOUM: A flow-ordered consistent update mechanism for software-defined networking in adversarial settings," in *Proc. of the 35th IEEE Inter. Conf. on Computer Communications, (INFOCOM16)*, 2016, pp. 1–9.

[17] P. Cerný, N. Foster, N. Jagnik, and J. McClurg, "Optimal consistent network updates in polynomial time," in *Proc. of the 30th Inter. Symp. Distributed Computing, DISC16*. Springer, 2016, pp. 114–128.

[18] S. Vissicchio, L. Vanbever, L. Cittadini, G. G. Xie, and O. Bonaventure, "Safe update of hybrid SDN networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1649–1662, 2017.

[19] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "Automated optimal firewall orchestration and configuration in virtualized networks," in *NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, April 20-24, 2020*. IEEE, 2020, pp. 1–7.

[20] ——, "Automated firewall configuration in virtual networks," *IEEE Tran. on Dep. and Sec. Comp.*, pp. 1–18, 2022.

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

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

[23] D. Bringhenti, F. Valenza, and C. Basile, "Toward cybersecurity personalization in smart homes," *IEEE Secur. Priv.*, vol. 20, no. 1, pp. 45–53, 2022.

[24] D. Bringhenti, G. Marchetto, R. Sisto, S. Spinoso, F. Valenza, and J. Yusupov, "Improving the formal verification of reachability policies in virtualized networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 1, pp. 713–728, 2021.

[25] D. Bringhenti and F. Valenza, "Optimizing distributed firewall reconfiguration transients," *Comput. Networks*, vol. 215, p. 109183, 2022.

[26] R. Robere, A. Kolokolova, and V. Ganesh, "The proof complexity of SMT solvers," in *Computer Aided Verification*. Springer International Publishing, 2018.

[27] G. M. Yilma, F. Z. Yousaf, V. Sciancalepore, and X. P. Costa, "Benchmarking open source NFV MANO systems: OSM and ONAP," *Comput. Commun.*, vol. 161, pp. 86–98, 2020.

[28] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Comput. Networks*, vol. 93, pp. 492–505, 2015.

[29] I. Pedone, A. Lioy, and F. Valenza, "Towards an efficient management and orchestration framework for virtual network security functions," *Secur. Commun. Networks*, vol. 2019, pp. 2 425 983:1–2 425 983:11, 2019.