Automation for Network Security Configuration: State of the Art and Research Trends

DANIELE BRINGHENTI, GUIDO MARCHETTO, RICCARDO SISTO, and FULVIO VALENZA, Dipartimento di Automatica e Informatica, Politecnico di Torino, Italy

The size and complexity of modern computer networks are progressively increasing, as a consequence of novel architectural paradigms such as the Internet of Things and network virtualization. Consequently, a manual orchestration and configuration of network security functions is no more feasible in an environment where cyber attacks can dramatically exploit breaches related to any minimum configuration error. A new frontier is then the introduction of automation in network security configuration, i.e., automatically designing the architecture of security services and the configurations of network security functions, such as firewalls, Virtual Private Networks gateways, and so on. This opportunity has been enabled by modern computer networks technologies, such as virtualization. In view of these considerations, the motivations for the introduction of automation in network security configuration are first introduced, along with the key automation enablers. Then, the current state of the art in this context is surveyed, focusing on both the achieved improvements and the current limitations. Finally, possible future trends in the field are illustrated.

CCS Concepts: • **Security and privacy** → **Network security;**

Additional Key Words and Phrases: Network security, network virtualization, policy-based management

ACM Reference format:
Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, and Fulvio Valenza. 2023. Automation for Network Security Configuration: State of the Art and Research Trends. *ACM Comput. Surv.* 56, 3, Article 57 (October 2023), 37 pages.
https://doi.org/10.1145/3616401

1 INTRODUCTION

Modern computer networks have been facing a progressive evolution in the latest years. On the one hand, network size is constantly increasing, due to the digitization of every activity. On the other hand, the heterogeneity of functions and technologies exploited in building networked architectures is increasing. These trends are visible, for example, in modern industrial networks, composed of a huge number of heterogeneous devices [1], and in the emerging **Internet of Things (IoT)** paradigm, based on the idea of connecting any device to the network, so reducing human interaction [2].

The main drawback of this incessant evolution is that the complexity of computer networks has been altogether increasing. Large-scale networks made of heterogeneous devices expose a larger

Authors' address: D. Bringhenti, G. Marchetto, R. Sisto, and Fulvio Valenza, Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Turin, Italy; emails: daniele.bringhenti@polito.it, guido.marchetto@polito.it, riccardo.sisto@polito.it, fulvio.valenza@polito.it.

0360-0300/2023/10-ART57 $15.00
https://doi.org/10.1145/3616401

attack surface, because cyber attackers can intrude on more entry points and interconnections. Besides, the heterogeneity of network devices makes it difficult to identify all their possible vulnerabilities with a larger variety of attack kinds hindering network management [3]. Therefore, a fundamental role is played by network security, which can counterbalance the presence of these vulnerabilities with adequate defense. However, enforcing the desired security properties in modern computer networks is a troublesome task for security managers. The main reason is that the configuration of security functions (e.g., firewalls, anti-spam filters, etc.) is traditionally performed manually, with a trial-and-error approach: Whenever an attack is detected, the configuration is modified accordingly. This work paradigm is not scalable, and it is prone to several errors due to the fallibility of humans.

To address this issue, automation has been recently leveraged by research to improve the state of the art of network security configuration. The main goal is to provide as automatic as possible configuration of security services to minimize human intervention. An automated process can be also combined with optimization techniques to avoid unnecessary resource consumption and, with formal verification, to identify or prevent configuration mistakes [4]. Another benefit introduced by automation is agility, which is essential to provide prompt reaction to security attacks. The shift from manual to automatic configuration has become feasible since the early 2010s thanks to a number of innovations, most notably network softwarization, in its two declinations known as **Network Functions Virtualization (NFV)** [5] and **Software-Defined Networking (SDN)** [6] and **Policy-Based Management (PBM)** [7]. On the one hand, NFV enables allocating virtual functions instead of manually installed physical middleboxes, whereas SDN decouples the network data plane from the control plane, leading to a centralization of the orchestration operations. On the other hand, PBM consists of deriving network configurations from policies describing network requirements.

In light of all these considerations, the main goal of this article is to provide a survey about the state of the art of automation for network security configuration, since a comprehensive synthesis of the research done in this field is not yet available. Although some survey papers related to this field have been published recently, none of them provides good coverage of this specific topic. Herrera et al. [8] provide a comprehensive state of the art of the **NFV Resource Allocation (NFV-RA)** problem, for which we will provide more details in Section 4. However, their survey mostly deals with the automatic virtual function placement on the physical infrastructure, while their report on service composition is limited and mostly focused on networking intents rather than security. Riekstin et al. [9] analyze policy refinement techniques to automatically manage green sustainability-oriented features of datacenter networks, but the security aspect is overlooked in this context, too. Moreover, only sustainable networks are studied, and with a single technique, i.e., policy refinement. Finally, Jabal et al. [10] present an extensive overview of methods for policy analysis, a problem related to PBM but orthogonal to the automation of network security configuration.

Even though a major focus will be on virtualized and cloud-based networks, i.e., the environments that best suit automation, techniques for traditional networks will be investigated as well.

The remainder of this article is structured as follows. Section 2 analyzes the motivations that have stimulated research on automation for network security, and the benefits that can be achieved by pursuing it. Section 3 describes the systematic method that has been followed to carry out the literature survey. Section 4 describes how, in our vision, based on the analyzed literature, fully automated network security service configuration should be organized. Sections 5 and 6 survey the most relevant works about the two main tasks (i.e., service composition and function configuration) for which automation can bring an effective contribution to network security. Section 7 answers the research questions defined in Section 3, and it highlights some future trends and directions that could be followed to make progress in this research area, with the aim to engage the readers in new challenges. Finally, Section 8 draws conclusions.

2 MOTIVATION AND PROBLEM STATEMENT

2.1 Limitations of Manual Network Security Configuration

Network Security Functions (NSFs) are the network functions used to ensure security defense against network attacks. This definition abstracts the concept of Security Controller. If a Security Controller is a middlebox that executes a security function, then an NSF is the function itself that provides security. An NSF is consequently abstract and independent from its implementation, which could be a hardware device (a traditional Security Controller) or a virtual entity. There are different kinds of NSFs. For example, filtering functionalities, such as firewalls, can block unwanted communications, **Virtual Private Networks (VPNs)** can ensure confidentiality and integrity of network communications, and **Intrusion Detection Systems (IDSs)** and **Intrusion Prevention Systems (IPSs)** can respectively detect unwanted network activities and mitigate their negative effects and risks.

The configuration of NSFs has been traditionally performed manually by the security manager, who is a professional figure disjoint from the network administrator in most companies. This person is in charge of collecting the security requirements formulated by network users and enforce them by placing and configuring a set of NSFs. However, configuring NSFs such as firewalls, VPN gateways, and IDSs has always been troublesome and more complex than the configuration of other service functions that provide networking features, like routers, **Network Address Translator (NATs)**, and load balancers. In fact, for NSF configuration, it is necessary to reason about all possible attacks that may occur in the network, not just about connecting services. Besides, manually enforcing the required protection requires expertise in using all the NSF configuration languages, which often are quite different from one another.

As a result, anomalies can likely arise in manually specified security configurations. In literature, an anomaly is defined as an incorrect specification of a network function configuration that an administrator may introduce. Several studies, such as the ones discussed in References [11, 12], extensively analyzed the impact of anomalies related to firewall and VPN configurations on the actual protection these NSFs must guarantee. For example, an unfeasible communication over a VPN occurs when the security manager defines a VPN configuration based on a technology not supported by an end point or a security level too high to be enforced by its available cipher suites. This anomaly is severe, because it completely prevents data exchange due to a hard misconfiguration. Instead a sub-optimization anomaly affects a firewall configuration if all packets matched by a filtering rule are also matched by another rule with higher priority. Even if this is not a hard misconfiguration, it may decrease the efficiency of the security operations, because the firewall takes more time to analyze its rule set when deciding the action to apply to each packet.

The problem of anomalies in the configuration of NSFs is being exacerbated year after year. An analysis of the Data Breach Investigations Reports produced by Verizon from 2013 to 2022[1] leads to two interesting considerations stressing the importance of the security configuration problem. Among the causes of security incidents, both misconfiguration and the macro-category it belongs to, i.e., miscellaneous errors, have a growing trend. Inside this category, the percentage covered by misconfiguration has increased from 0% to 42%, becoming the first cause of breaches within the miscellaneous error category. A similar pattern can be seen in the growing trend of the error

[1]The reports are available at the following link: https://enterprise.verizon.com/resources/reports/dbir/

category itself, whose incidence growed from the 5% of the previous report to the 13% of the last one. Even if these percentages are lower than the ones associated with other incident classes, the absolute number of incidents due to errors, including misconfigurations, is significant. As Verizon reports 23,896 security incidents occurring in 2021, over 1,300 incidents are therefore due to misconfigurations. This high number of related incidents cannot be overlooked when protecting a computer network, without forgetting that many incidents are commonly not declared and consequently not analyzed in the report.

Here are the main reasons why this problem has become so relevant.

Role separation and lack of communication. Security manager and network administrator are separate roles, so lack of communication or knowledge about the other expertise area can easily lead to mistakes in security configuration [13]. For example, if the network administrator does not provide the security manager full information about the network settings, then the latter could make incorrect assumptions when starting to design the security architecture for the network service defined by the former.

Increasing network size. The number of offered network services is constantly increasing, from *Voice-over-IP* to video streaming and from traditional e-mails to in-app communications. The size of the new generation computer networks had to adapt to these needs by becoming bigger. However, the presence of more communication channels increases the possibility of vulnerabilities.

Increasing network complexity. According to the KISS rule, first proposed by the U.S. Navy in 1960, complexity is the worst enemy of security. Indeed, keeping NSFs simple would be fundamental to keep network security configuration easy. Nevertheless, the complexity of NSFs is growing, as reaction to the new emerging attack types: For instance, new kinds of firewalls are produced to work at different levels of the ISO/OSI stack, artificial intelligence algorithms are introduced as intrusion prevention systems, and data loss prevention modules are applied across many network devices. Security configuration correctness is consequently becoming almost impossible to achieve by manual operation: The complexity introduced to provide security becomes a double-edged sword, since it creates new vulnerabilities while trying to stop others.

Increasing network heterogeneity. Modern generation computer networks are characterized by high heterogeneity: Not only the function types are quite different from one another, but differences also arise, because different functions are produced by many different vendors. However, heterogeneous networks are intrinsically more complex than homogeneous ones [14]. For example, if firewalls produced by different companies are installed in a network, then they would require different configurations to set the same filtering policy.

Trial-and-error configuration approaches. The trial-and-error approach that commonly characterizes manual security configuration lets security managers save time in the short term, but in the long term it may lead to ever increasing configuration size and complexity, which in turn favors mistakes such as the introduction of contradictory rules.

Impact of security breaches. In the latest years, cyber attackers have been developing more powerful strategies to intrude information systems. Because of the potential errors due to a manual security configuration, the resulting security breaches have a twofold impact. On the one hand, the financial conditions of the firms affected by a breach are seriously threatened. A multi-faceted analysis carried out in Reference [15] states that also non-breached firms experience significant negative economic impact around the announcement of a breach that is indirectly related to their activity. On the other hand, a breach can also damage non-monetary factors, such as consumer confidence, social trust, and personal safety, as demonstrated in Reference [16] with a visualization technique based on artificial intelligence. Consequently, recent approaches in the literature aim at estimating security costs by taking into account also transparent indirect costs related to security management, such as the method called *Cost Assessment of Personnel Activities in Information Security Management* [17]. From this analysis, manual prevention and mitigation of breaches is clearly becoming impractical.

2.2 Introducing Automation for Network Security Configuration

By definition, automation is a technique that "emphasizes efficiency, productivity, quality, and reliability, focusing on systems that operate autonomously, often in structured environments over extended periods, and on the explicit structuring of such environments" [18]. In an automatic system, the core principle is the minimization of human interventions: After the system receives an external input from a human being or from another system, it should be able to work without requiring other external contributions. Even though design complexity represents a potential criticality for automatic systems, nevertheless the possible benefits equalize and overcome that drawback. Both activity productivity and solution quality typically achieve a great improvement: On the one hand, the human operator is not demanded to perform the whole task but only to make the system properly start and provide assistance or maintenance during the automatic operations; on the other hand, the solution is reached faster and with a better accuracy.

In network security, the introduction of automation represents a possible solution to human errors characterizing manual configuration of security functions. A fundamental requirement for enabling automatic security configuration is agility: Whenever the current state changes, the system must be able to automatically adapt to the new conditions in the shortest possible time, so that no inconsistencies are created. The absence of agility in traditional computer networks represented one of the main reasons why automation had not already been fully introduced in the past in this engineering field. In recent years, the perspective changed thanks to the softwarization of networking, i.e., most notably SDN and NFV, and to the introduction of PBM.

SDN decouples the data plane from the control plane [19], and this decoupling allows us to centralize all the orchestration operations of the control plane in a single architectural element, named the SDN Controller. This element coordinates all the SDN switches of the data plane through protocols, such as OpenFlow [20], which provide an abstraction from the vendor-specific implementations of the forwarding devices. Thanks to these characteristics, SDN introduces several advantages with respect to traditional networking paradigms. First, as the SDN Controller can configure forwarding rules on all the switches of the data plane, it can force network traffic to pass through specific appliances. Second, the controller can dynamically update SDN switch configuration to comply with new security requirements as soon as they emerge. In fact, it can simply install new rules on the switches it manages. Third, it can configure a different security service exploiting the same hardware switches for different users.

NFV virtualizes network functions as software processes named **Virtual Network Functions (VNFs)**, whose possible implementations are traditional **Virtual Machines (VMs)** [21] and Dockers [22]. NFV highly contributes to automatize network security configuration. Every time the service must be reconfigured by introducing a new function or removing an existing one, it is sufficient to start a new software program or to stop a running one, instead of physically managing the appliances. The lifecycle itself of each VNF can be managed automatically, and the failure of a virtual security function can be overcome by executing some programming scripts that would restore it with the same previous configuration. At the same time, the reaction to cyber attacks becomes faster: Instead of having to access the physical appliance, the configuration of the service can be changed more easily by accessing a VM or a Docker, thus saving vital time in blocking an ongoing attack.

The agility and reactivity provided by SDN and NFV enabled the coupling of network security management with PBM, i.e., defining the network security behavior by means of policies. A policy is a definite goal or course or method of action, which can be expressed as a set of rules, to administer, manage, and control access to network resources [23]. The idea is that a network administrator only specifies what security properties the network should fulfill, without defining how, i.e., without defining the configuration of each security function, because this latter task is automatically performed by an assisting tool. An architectural model that can be used for Policy-Based Management has been described in RFC 3060 [23], and it has been later improved by extended models, such as Ponder [24], KAoS [25], or Rei [26].[2] This architecture reflects the whole process through which a policy, after being specified by the user, is processed and finally enforced by the network functions. This process can be structured into three main phases. First, the policies are specified by the user and then automatically analyzed so that any anomaly is found (policy analysis). Second, the user-specified policies are refined into the configuration rules: This task is needed, because the language that is typically exploited by the user is high level to be independent of the technicalities of the functions (policy refinement). Finally, a verification is often performed to check if the result is compliant with the original policies (policy verification).

Policy refinement is the stage that mostly suits network security automation. Network functions, even when belonging to the same type, are typically implemented in different ways, as they are produced by different vendors. In virtualized networks, this issue is exacerbated, because anyone can easily create a VNF by writing a software program. Policy refinement addresses this problem when coupled with the policy continuum [28]. The core idea is the existence of different levels of abstraction for the representation of policies and function configurations. According to the analyses by Basile et al. [29] and by Hermosilla et al. [30], three classes of policy languages may be exploited for a complete representation as follows: (1) **High-level Policy Languages (HPLs)** allow users to express policies in a user-friendly notation, thus easing readability and understandability; (2) **Medium-level Policy Languages (MPLs)** express policies within a structured implementation-independent representation, based on conditions (i.e., the events that must happen so that the policy is triggered) and actions (i.e., the operations that a function must execute whenever all the policy conditions are true); and (3) Low-Level Configurations express policies with the languages specifically required by the network functions that must enforce them. In this policy continuum, policy refinement represents the decision-making process that changes the abstraction level of the policy representation from higher to lower level classes.

According to this discussion, Policy-Based Management is a fundamental component of automated methodologies for the configuration of a network security service. The main reason is that automation always requires input data to perform the operations needed to compute the outcome. User-specified network security policies perfectly play this role, since they describe the behavior that the network must satisfy, thus allowing the automated methodology to establish consistent function configurations. Moreover, thanks to the intermediate abstraction level represented by MPLs, even though anyone can define their own virtual function implementation, the automated methodology that should be created for computing their configuration can be designed without caring about this aspect. Indeed, the final translation from MPLs to low-level configurations can be performed independently from the refinement from HPLs to MPLs. On the one hand, all the information required for security enforcement is already provided by the medium-level representation. On the other hand, this final translation consists in a syntax translation and simply requires the knowledge about the syntax of the languages of the low-level configurations. Therefore, when the problem of automatic configuration is investigated, it is possible to focus on the generation of the medium-level representation.

[2]Further information about policy-based management approaches is reported in Reference [27].

2.3 Advantages of Automatic Network Security Configuration

The trend of introducing automation for network security configuration is motivated by its ability to overcome most of the limitations dissected in Subsection 2.1.

First, automated orchestrators can be used to configure network security without requiring a high level of network security expertise or experience. Expert security managers are few in number and have high costs. Consequently, in many companies, most of the people working in network security have networking expertise, and they are supervised by a restricted number of security experts. If these people use automatic tools, then their lack of expertise is mitigated, thanks to the aid provided by the tools. Of course, they must monitor the tools that perform the automatic operations, but monitoring is much less complex, less error prone, and less time-consuming than the full manual design of a security service.

Second, size and heterogeneity of modern generation computer networks can be better dominated with automation than manually. On the one hand, an automated orchestrator of network security functions can have a complete overview of the whole network architecture by taking global decisions that a human being would have difficulty to manage. On the other hand, heterogeneity can be managed by an abstraction layer between the automated orchestrator and the heterogeneous security functions, so that the configuration that is automatically computed for each one is translated into the correct vendor-dependent commands to set up the specific device. This translation step, if performed manually, requires the complete knowledge of how each parameter must be set for any implementation of the function; in this case, a software process with all the required information can perform this operation faster and more reliably.

Third, differently from a manual configuration, which is based on a trial-and-error approach, an automated orchestration can directly find a correct and optimal solution. Optimization could be exploited, for example, to allocate only the middleboxes that are really needed to provide the service, so that only the required resources are actually installed. Or it could also be exploited to maximize security protection. Achieving the same result manually would be extremely difficult, since correctness itself is hard to achieve manually.

Despite all these benefits that automation could carry over to the network security field, some potential drawbacks could also be highlighted. However, most of them are only apparent and mostly depend on human prejudice. The main problem is clearly not technical but related to the psychological field. In history, automation has always been considered potentially dangerous, because the users of automated tools fail in fully understanding how such tools work and fear they could lead to bigger problems than manual operations. However, this common sense is not justified. First, automation can be exploited to provide a guarantee of correctness by leveraging automated formal verification techniques, while achieving the same guarantee manually is more difficult. Second, any automated tool is developed by a human being, who should provide the full documentation to make others understand how it works and how some potential problems should be managed. Third, the problem is not the "over-automation" but either the design of the automated tools or their supervision [31]. Both these operations rely on human beings, thus proving that, at the end, any drawback that automation can introduce is related to some activities directly or indirectly performed by humans.

Summarizing, the statistics that have been reported in this section come from research studies, which further supports the idea that automation may play a central role in future network security. The challenge that arises is rather how to answer the following questions: (i) Which technologies can be exploited as foundations for automated network security methodologies? and (ii) How can research further deploy this novel path by improving the current state of the art?

3 METHOD FOR LITERATURE SURVEY

This survey has been undertaken as a systematic literature review according to the well-known guidelines proposed in Reference [32]. The objective is to identify and classify the methodologies for automatic synthesis of network security services and automatic configuration of network security functions, from a computer science researcher's point of view. The steps that have been followed for the execution of this review are documented below.

3.1 Research Questions

The research questions addressed by this survey are as follows:

RQ1 (Time distribution): *What is the time distribution of the works about network security configuration automation?*
Research in network security configuration automation has recently started to trend again, thanks to the advent of virtualization in the networking field. However, it is well known that the same topic has been addressed in the past, too. A pair of pilot studies, Firmato [33] and MIRAGE [34], date back to the first decade of the 2000s. Consequently, it is interesting to understand the publication trend of papers on this topic throughout the years.

RQ2 (Enhancing features): *How are automatic methodologies enhancing network security configuration with respect to manual strategies?*
Automation can improve the produced output quality, as it can perform more complex and faster operations than what humans can do. It is expected that the same applies to the network security configuration field. An objective of this literature review is to identify the common enhancements and improvements that have already been achieved by the state-of-the-art automatic approaches for network security configuration with respect to the manual ones. From this analysis, researchers can understand which paths have already been investigated.

RQ3 (Limitations): *What are the limitations of the state of the art in the area of network security configuration automation?*
A crucial objective of this study is to understand the current limitations of the proposed approaches. Even though important steps have been taken to improve the state of the art, not all the problems in this area have been solved, and the existing papers have shortcomings to be addressed. From the identification of these limitations, researchers can intuitively infer emerging challenges and research trends that should be followed in the future to fill the existing gaps.

3.2 Search Process

The search process of conference proceedings and journal papers was carried out in the following databases: SCOPUS, Science@Direct, Wiley InterScience, IEEE Digital Library, ACM Digital Library, SPRINGER, and ISI Web of Knowledge. The following search string has been used in the search engine of the previously listed databases:

computer AND (network OR networking) AND (security OR protection) AND
(automation OR automatic OR automated OR programmability OR programmable) AND
(configuration OR configure OR synthesis OR synthesize OR composition OR compose)

The tool *Publish or Perish* has been used to automate the search process for the supported databases.

The results have been enriched with the snowballing technique, i.e., for each study, its references and the papers citing it have been analyzed. Then, all enriched search results have been merged by fulfilling the following criteria:
(C1) Impurity and duplicates removal: Duplicate results were removed.

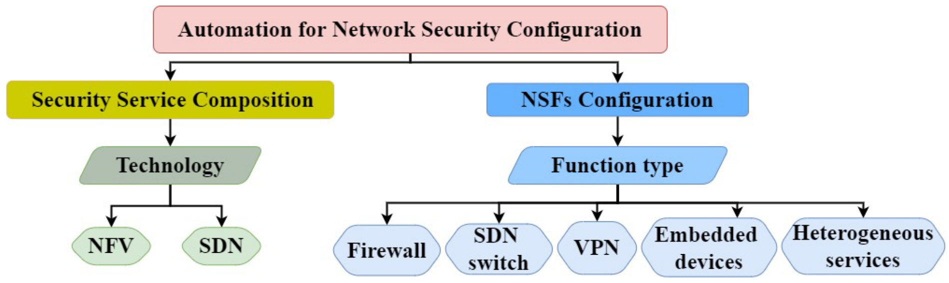


Fig. 1. Taxonomy of studies related to automation for network security configuration.

(C2) Inclusion criteria: Papers were considered if they respected all the following criteria: (1) papers describing methodologies that can be used for the automatic synthesis of a network security service or the automatic configuration of the network security functions; (2) papers published between 1996 and 2023; (3) papers subject to peer review (e.g., journal papers, papers published as part of conference proceedings will be considered, whereas white papers will be discarded); and (4) papers written in English and available in full text.
(C3) Exclusion criteria: Papers were excluded if they fulfilled at least one of the following criteria: (1) papers describing methodologies only for network management automation, without any reference to network security; (2) papers limited to present a formal theory for networking, without any substantial possible application to computer networks; (3) secondary studies (e.g., systematic literature reviews, surveys); and (4) studies in the form of tutorial papers, poster papers, editorials, because they do not provide enough information due to page limitation.
(C4) Combination: If there are multiple papers related to the same study, then a single record is kept for all of them. This action is necessary for ensuring completeness and traceability of results. For example, if a primary study is published in more than one paper (a conference paper, then extended to a journal version), then only one instance will be counted as a primary study. Generally, the journal version will be preferred, since more complete.

Finally, we positively verified that the combined result of the search process includes the following pilot studies (relevant papers for the investigated literature area) [29, 33–36].

3.3 Data Collection and Synthesis to Address the Research Questions

Here we describe how data collection and synthesis have been performed and how we provide responses to the research questions according to the results of those operations.

First, data collection has been performed independently by three authors so that the results could be compared. In merging the results according to the described method, disagreements have been resolved by consensus among the three authors. The fourth author checked how the extraction was performed. At the end of the review process, 98 papers were collected.

Second, the collected data were tabulated according to the taxonomy shown in Figure 1. This taxonomy is the result of patterns identified when analyzing the state-of-the-art literature. In particular, for the studies about security service composition, large differences exist depending on the main technology that is used to introduce automation (SDN or NFV). Instead, approaches for automatic function configuration mainly differ according to the function types for which they have been designed (firewalls, SDN switches, VPN gateways, and embedded devices), while a limited number of them can be applied to heterogeneous security services composed of multiple function types. Each data row includes the characteristics listed in Table 1.

Table 1. Features Extracted for Papers Listed in Tables 2 and 3

Reference	The reference to the paper where the automated methodology is illustrated in detail.
Target	The network type for which the methodology is designed and validated (i.e. traditional, virtual or both).
Fixing	True (✓) if the methodology can automatically fix a security service or NSF configuration, false (X) otherwise.
Scratch	True (✓) if the methodology can automatically create a service or a NSF configuration from scratch.
Correctness	True (✓) if the methodology uses formal proofs, verification techniques or a correctness-by-construction.
Optimality	True (✓) if the methodology can find the optimal solution according to some optimality criteria
Knowledge base	The origin of the input information exploited by the methodology to automatically compute the solution.
Technology	The adopted virtualization paradigm, i.e., SDN or NFV (only for papers about service composition).
Supported NSFs	The NSFs that are supported by the methodology (only for papers about NSFs configuration).
Scalability	A concise indication of the scalability achieved by the methodology (i.e., number of NSFs, requirements or rules).

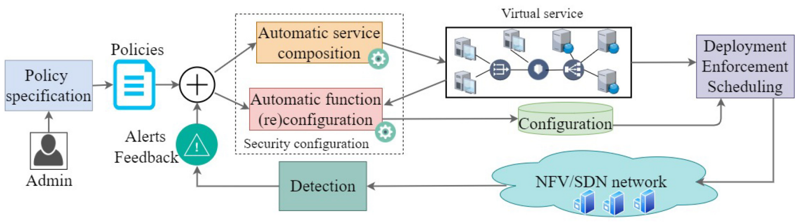


Fig. 2. Full workflow of automated virtual network configuration.

According to such taxonomy and characteristics list, Section 5 and Section 6 summarize the papers collected about automatic service composition and automatic function configuration, respectively. This descriptive synthesis represents the key to understand how the three research questions can be answered.

Finally, the final elaboration of the literature investigation results, jointly with their quantitative analysis, is presented in Section 7. This discussion follows the descriptive synthesis of the collected papers, because, as recommended by the guidelines described in Reference [32], this allows readers to have the knowledge necessary to fully understand the answers.

4 AUTOMATIC NETWORK SECURITY CONFIGURATION WORKFLOW

The process to design and configure a network security service can be organized in different ways. Nonetheless, by analyzing the different approaches, we identified some phases that are common for most of them.

In this section, we present such phases and how they are usually organized in a fully automated process, as found in literature. As network efficiency is also a target to be addressed when configuring network security services, not all the phases of such a process are strictly related to security. Some of them are focused on network parameters or requirements such as latency or bandwidth. Nevertheless, our goal is to provide the readers with an overall picture of the different operations that are performed when configuring virtual network services and then focus attention on those that are specifically related to network security. It is worth noting that some phases of this automated process apply to virtualized environments as well as to traditional networks.

The typical organization of a fully automated workflow for security configuration is graphically represented in Figure 2. It is composed of the following phases.

Policy specification. An automated tool requires external information to compute the consequent output. The main pool of input information is represented by the network security policies, i.e., the security requirements defined by a network administrator. The policy specification phase is when these requirements are specified. For obvious reasons, it requires user intervention.

This step is critical for a number of reasons. First, the user must correctly define all the network security policies that must be enforced in the service; otherwise, the result computed by the automated tool will not be the expected one. Even if we assume automated tools are correct, some mistakes may originate from human faults. As stated in Section 2, the user must comply with all guidelines in input specification and ensure that the specified policies represent the real requirements. For example, if the user incorrectly specifies the characteristics of the traffic to be blocked, then an automatic tool for firewall configuration would define filtering rules that are wrong although adherent to the incorrectly specified policy. Or, if unnecessary policies are given as input, the result may be non-optimal. For all these reasons, policy analysis should be included in this phase to identify and correct errors or sub-optimizations in the definition of policies, and avoid, in this way, to waste computation to process wrong or redundant policies.

Automatic service composition. After the specification of the network security policies, the first automatic computation step targets the creation of the virtual service. The logical topology representing the interconnection of the security functionalities, e.g., firewalls, deep packet inspectors, and so on, is called in literature Service Function Graph, or more simply **Service Graph (SG)**, and it represents the generalization of the **Service Function Chain (SFC)** concept [37]. The main difference is that in the latter the functions are chained, so that the traffic flow passes through a specific ordered list of functions, while in the former the ramified structure allows the definition of a richer full service, yet making the design of the service more complex.

Concerning this aspect, if the SFC definition [38] already deals with a consistent number of issues, e.g., topological dependencies, consistent ordering, elastic service delivery or limited end-to-end service visibility [39], then the definition of a SG involves a number of challenges that is even higher. A first reason is that the end users can have multiple access points to the service that can change over time. Therefore, traffic flows from a certain user to a certain destination might follow different paths, and all these paths must be considered when protecting those flows. For example, if a policy requires that the traffic to a given destination crosses a specific list of NSFs, then the SG must be designed so that each path that such traffic may follow crosses the required list of NSFs. This is one of the reasons why a correct design of the service taking into account all the security requirements is hard to achieve manually, and an automated solution is needed.

The result of this step is the SG, enriched with the NSFs that are needed to enforce the requested security policies, but the functions still miss the configuration that will have to be enforced on the corresponding VNFs, when deployed on the physical infrastructure. It is also worth mentioning that this step is a generalization of the VNF Chain Composition, which is the first component of the NFV-RA problem [8]. The former, in fact, takes also security requirements into account, while the latter exclusively focuses on networking constraints.

Automatic function configuration. The automatic computation of function configurations follows the design of the virtual service, but it can also be joined with service design into a single step of the workflow. The goal is to determine the configuration rules of each NSF, according to their position in the topology, so as to satisfy security requirements. Even though the final outputs must be the low-level configurations, the policy refinement activity that is intrinsically involved can be organized into a number of steps.

This activity is the most critical one in the whole workflow, because, as stated in Section 2, most of the breaches are due to erroneous configuration of the NSFs. It is, altogether, the most difficult operation: If, on the one hand, composition requires us to design a service that offers all the requested functionalities, then, on the other hand, configuration is the operation in which these functionalities are put to work. Optimization plays a crucial role in this phase to obtain an efficient service. For example, the minimization of the number of configuration rules for a packet filtering firewall is known to optimize its efficiency, since each received packet must be compared to less rules. A similar reasoning can be applied to a VPN gateway: Establishing the minimum number of algorithms that must be applied for a channel communication protection reduces the computation complexity and, consequently, the overhead needed to enforce security.

Deployment, enforcement, and scheduling. The result of the previous two phases is a virtual service, including functions and corresponding configurations. However, this topology is designed at logical level, and a mapping to a physical infrastructure is still necessary. In fact, the substrate network is typically made of general-purpose servers on which the VNFs composing the logical service must be placed in the best way. This problem is known in literature as Virtual Network Embedding or, alternatively, as VNF-Forwarding Graph Embedding. It represents the second step of the NFV-RA problem. It has been, altogether, one of the most researched themes in the context of network softwarization over the past few years. Finding the optimal solution to this problem is not trivial. However, this problem usually has to do only with network-related requirements, such as resource consumption or latency constraints. All the required security properties, instead, should already be enforced thanks to the previous workflow stages. For this reason, a further dissection of this problem and its related literature are out of the scope of this survey. The interested reader can find a full presentation of the topic in Reference [40].

Additionally, other two tasks must be performed at this stage. The first one is the enforcement of the configurations, automatically computed at the previous stage, onto the target NSFs. This operation may involve only a change of format and language of the configurations to adapt them to the vendor-specific characteristics of the selected NSFs implementations. The second one, instead, is known in the literature as the third stage of NFV-RA and traditionally named VNF Scheduling. At this stage, the best execution order of the VNFs is identified, respecting all precedences and dependencies, with the goal of minimizing the total execution time of the network service, so improving the overall performance. However, as for the embedding phase, also the VNF Scheduling problem targets network optimization rather than security. It is therefore out of our specific scope.

Detection. After the embedding, enforcement, and scheduling operations are completed, the network security service is finally active, and it can provide both network functionalities to end users and protection from cyber attacks. However, this protection is never full: Attacks, such as the exploitation of vulnerabilities, are still possible at any time. Consequently, it is essential to install IDS to find out such attacks.

Mitigation and reconfiguration. When an alert is raised by an IDS, the detected attack must be mitigated (e.g., blocked, or isolated). Consequently, automation must be exploited again, because the previous configuration is lacking or not consistent with the new security goals that arise from the mitigation strategy (e.g., attack isolation). The results of all the previous phases must be questioned and possibly repeated. In this case, however, the input of the automatic service generation is not represented exclusively by user-specified policies but also by the information about the attack collected during the detection phase. In Figure 2, this is represented by the loopback connection. In attack mitigation, reconfiguration should minimize the number of changes, so that the operations are faster. For example, instead of designing a completely different service architecture, the current structure could be kept by adding a new security function; then, when the configuration must be computed, instead of reorganizing the rules of each middlebox, the minimum set of rules that need modification should be identified, so saving time and at the same time minimizing the number of interactions between an automated orchestrator and the single devices. However, in some circumstances, changing a configuration to satisfy new security requirements while trying to minimize the changes is a task more difficult than regenerating the service from scratch. For this reason, not all the approaches available in literature propose a smart reconfiguration mechanism, but some simply assume that all configurations must be recomputed in the mitigation step.

The just described workflow matches the requirements of several typical use cases in modern computer networks. Here we provide three concrete examples: (1) University campus networks have been migrating their authentication and access control mechanisms toward SDN [41]. Manually controlling a big network topology, while guaranteeing the access privileges, correctly and promptly reacting to attacks, is not easy. Automating their full configuration would be compliant with the dynamism required by campus networks and it would reduce the human workload. (2) A broad range of IoT-based applications and cyber-physical systems (e.g., autonomous cars) have strict requirements in terms of secure communications. Therefore, the multi-access edge computing paradigm is gaining high momentum, as edge environments represent a strategic position to enforce security features in a network [42]. However, the number of network nodes enforcing security increases, and automation is becoming necessary to overview all of them at the same time. (3) Virtualization has recently contributed to the management of home networks, enabling personalization of smart devices [43]. Automation can compensate the lack of technical and security knowledge of the smart devices users, by assisting them in securely configuring their home network.

In summary, in the typical workflow of automated virtual network configuration, three main processes take part in automation: security configuration, deployment-enforcement-scheduling, and detection. This survey focuses on security configuration, composed of two main operations: automatic service composition and automatic function configuration. The other processes have already been dissected in other papers: deployment in Reference [8], scheduling in Reference [44], detection in References [45, 46].

5 AUTOMATIC NETWORK SECURITY SERVICE COMPOSITION

This section surveys the most relevant work about automatic composition of network security services. Table 2 provides a complete overview of all the papers that we selected and that fall in this area. The meaning of the columns of Table 3 is explained in Table 1. These studies are divided into two groups, according to the taxonomy illustrated in Figure 1: (1) papers focusing on SDN-based networks (Subsection 5.1) and (2) papers focusing on the synthesis of NFV-based security services or on the enrichment of existing virtual networks with security functions (Subsection 5.2).

5.1 Automatic Service Composition in SDN-based Networks

In an SDN network, the main goal related to automatic service composition is to design the architecture of a network made of SDN switches. The security requirements are commonly expressed in terms of traffic steering policies, e.g., for a traffic flow the path that it should follow is specified, or the requirement that such flow does not reach some destinations or that it does not cross non-permitted switches. However, switch configurations, including the forwarding and filtering rules by means of which the policies are enforced, are not managed by the approaches presented in this subsection, whose purpose is the design of the service architecture, but by those presented in Subsection 6.2.

A milestone for automatic security service composition in SDN networks was the solution proposed in Reference [47], named FRESCO. It is a framework based on the well-known OpenFlow protocol, one of the communication protocols most commonly used by SDN controllers to access to the forwarding plane of network switches. FRESCO introduces the possibility of designing composable security architectures made of detection and mitigation modules. This framework uses code snippets, called modules, which can be combined to create security functions. These modules can inter-operate and exchange helpful information to make more grounded decisions, which represents a novelty not appearing in any other work related to the automatic creation of SDN-based services. Moreover, each module can be triggered according to an action-reaction paradigm,

Table 2. Comparison among Solutions for Automatic Network Security Service Composition

Reference	Target	Fixing	Scratch	Correctness	Optimality	Knowledge base	Technology	Scalability
[47]	Virtual	X	✓	X	X	U	SDN	4,000 rules
[49]	Virtual	X	✓	X	✓(ILP)	U	SDN	~250 functions
[50]	Virtual	X	✓	X	X	U	SDN	250 switches
[51]	Virtual	X	✓	✓	X	U	SDN	No information
[52]	Virtual	X	✓	X	X	U	SDN	No information
[53]	Virtual	✓	X	X	X	U, S	SDN	No information
[54]	Virtual	✓	X	✓	X	S	SDN	~70,000 rules
[55]	Virtual	X	✓	X	X	U	NFV	5 functions
[35]	Virtual	X	✓	X	X	U	NFV	16 functions
[56]	Virtual	X	✓	X	X	U	NFV	15 functions
[57]	Virtual	X	✓	X	X	U	NFV	79 nodes
[58]	Virtual	X	✓	X	✓(heuristic)	U	NFV	10 functions
[59]	Virtual	X	✓	X	✓(ILP)	U	NFV	8 functions
[60]	Virtual	X	✓	X	✓(ILP)	U	NFV	No information
[61]	Virtual	X	✓	X	✓(ILP)	U	NFV	7 functions
[62]	Virtual	X	✓	X	✓(ILP)	U	NFV	No information
[29]	Virtual	X	✓	X	✓(ILP)	U	NFV	~20 functions
[63]	Virtual	X	✓	X	✓(ILP)	U	NFV	~10 functions
[64, 65]	Virtual	X	✓	✓	✓(ILP)	U	NFV	~1000 functions
[66]	Both	✓	✓	X	X	U, S	NFV	100 functions
[67]	Virtual	✓	X	X	X	U, S	NFV	No information
[68]	Both	✓	✓	X	✓(heuristic)	U, S	NFV	60 firewalls
[36]	Both	X	✓	✓	✓(iterative SMT)	U	NFV	20 firewalls
[69, 70]	Both	X	✓	✓	✓(MaxSMT)	U	NFV	~100 firewalls
[71]	Both	X	✓	✓	✓(MaxSMT)	U	NFV	~25 firewalls
[72]	Virtual	X	✓	✓	✓(MaxSMT)	U	NFV	~100 functions

U = User-specified policies, S = Security chain

thus avoiding further human interventions after the initial design of the service. Even though neither formal verification nor optimization enrich this framework, FRESCO represents the peak of several works dealing with OpenFlow-based declarative query languages (e.g., Frenetic [48]) and the basis for other related automated approaches.

Subsequent papers related to this area include References [49–52]. The first two (i.e., References [49, 50]) formulate the traffic steering problem as an **Integer Linear Programming (ILP)** problem, targeting optimality criteria to be fulfilled in the design of a security service. Reference [49] proposes a framework called Software-defined Middlebox Policy Enforcement, which exploits SDN to automatically enforce policies when planning the placement of middleboxes in a network. The purpose is to optimally balance the traffic load across the middleboxes that are laid to compose the service. Reference [50], instead, proposes a solution based on a special data structure called MultiPoint-To-Point Tree, which was originally created for Multi Protocol Label Switching networks for managing traffic steering. In both approaches the requirements that a user can specify are security related (e.g., it could be required that a specific traffic flow crosses the sequence of firewall, IDS and proxy). However, the same does not apply to the optimization goals, which are only related to networking parameters, such as the minimization of the load across the middleboxes composing the service. Reference [51] proposes a rule-based system for automatic composition of security chains including formal verification of their compliance with the requirements to be fulfilled. This approach uses logic programming for establishing the functional specification of the security chains after evaluating the different kinds of traffic in the network and classifying them. The automatically synthesized chains are then created by using the Pyretic language [73], which is part of the Frenetic framework [48], for programming the SDN controller. However, differently from the previous ones, this approach is mainly meant to protect Android applications, even though the proposed formal models for achieving the synthesis of a security service could also be theoretically applied in different environments. Instead, Reference [52]

Table 3. Comparison among Solutions for Automatic Network Security Function Configuration

Reference	Target	Fixing	Scratch	Correctness	Optimality	Knowledge base	Supported NSFs	Scalability
[33, 74]	Traditional	X	✓	X	X	U	Firewall	Single firewall
[75]	Traditional	X	✓	X	X	U	Access control devices	~10 devices
[76]	Traditional	X	✓	X	X	U	Firewall	Distributed firewall
[77]	Traditional	X	✓	X	X	U	Firewall	No information
[78]	Traditional	X	✓	✓	X	U	Firewall	No information
[79]	Traditional	X	✓	✓	X	U	Access control devices	~1,000 nodes
[80]	Traditional	X	✓	✓	X	U	Access control devices	No information
[81]	Traditional	X	✓	✓	X	U	Firewall	Single firewall
[82]	Virtual	X	✓	✓	X	U	Firewall	Single firewall
[83]	Both	X	✓	✓	X	U	Firewall	~5 firewalls
[84]	Both	X	✓	X	X	U	Access control devices	~50 devices
[85]	Both	X	✓	X	X	U	Access control devices	~200 devices
[86]	Both	✓	✓	✓	X	U	Access control devices	~10 devices
[87]	Virtual	X	✓	X	X	U	Firewall	~1,700 firewalls
[88]	Virtual	X	✓	✓	X	U	Firewall	~15 policies
[89]	Both	X	✓	X	X	U	Access control devices	~1,000 policies
[90]	Both	X	✓	X	X	U	Access control devices	~60 policies
[91]	Both	X	✓	✓	X	U	Firewall	3 firewalls
[92]	Both	X	✓	✓	X	U	Access control devices	texttt~100 devices
[69]	Both	X	✓	✓	✓(MaxSMT)	U	Firewall	~50 firewalls
[70]	Both	X	✓	✓	✓(MaxSMT)	U	Firewall	~100 firewalls
[71]	Both	X	✓	✓	✓(MaxSMT)	U	Firewall	~25 firewalls
[93]	Traditional	✓	X	X	X	N	Firewall	No information
[94]	Traditional	✓	X	X	X	A	Access control devices	No information
[95]	Traditional	✓	X	X	X	F	Firewall	No information
[96]	Traditional	✓	X	X	X	F	Firewall	60 rules
[97]	Traditional	✓	X	X	X	F	Firewall	No information
[98]	Traditional	✓	X	X	X	F	Firewall	No information
[99]	Traditional	✓	X	✓	X	F	Firewall	Single firewall
[100]	Traditional	✓	X	✓	X	F	Firewall	Single firewall
[101]	Traditional	✓	X	✓	✓(MaxSMT)	A	Access control devices	~400 devices
[102]	Traditional	✓	X	✓	✓(calculus)	A	Access control devices	~5 devices
[103, 104]	Virtual	X	✓	X	X	U	SDN switch	~10 switches
[105]	Virtual	✓	✓	X	X	U	SDN switch	No information
[47]	Virtual	X	✓	X	X	U	SDN switch	Single switch
[106]	Virtual	X	✓	X	X	U	SDN switch	~100 switches
[107]	Virtual	X	✓	X	X	U	SDN switch	~15 switches
[108]	Virtual	✓	X	X	✓(ILP)	U	SDN switch	~35 switches
[109]	Virtual	X	✓	X	X	T	SDN switch	~15 switches
[110]	Virtual	✓	X	X	X	U	SDN switch	~10 switches
[111, 112]	Traditional	X	✓	X	X	U	VPN gateway	~35 gateways
[113]	Traditional	X	✓	X	X	U	VPN gateway	~85 requirements
[114]	Traditional	X	✓	X	X	U	VPN gateway	~50 gateways
[115]	Traditional	X	✓	X	X	U	VPN gateway	60 requirements
[116]	Traditional	✓	X	X	X	V	VPN gateway	~1,000 gateways
[117]	Traditional	✓	X	X	X	V	VPN gateway	500 rules
[118]	Virtual	X	✓	X	X	U	VPN gateway	No information
[119]	Both	X	✓	✓	✓(MaxSMT)	U	VPN gateway	No information
[120–124]	Traditional	X	✓	X	X	U	Embedded systems	No information
[125, 126]	Traditional	X	✓	X	X	U	Embedded systems	~10 devices
[127]	Both	X	✓	✓	X	U	Embedded systems	~100,000 devices
[128]	Traditional	X	✓	✓	X	U	Embedded systems	~20,000 devices
[129]	Both	X	✓	✓	✓(MaxSMT)	U	Embedded systems	~ 100 devices
[130, 131]	Both	X	✓	✓	X	U	Embedded systems	~ 30 devices
[42, 132]	Virtual	X	✓	X	X	U	Embedded systems	No information
[133]	Virtual	X	✓	✓	X	U	Embedded systems	No information
[134]	Virtual	X	✓	X	X	U	Embedded systems	~50 devices
[43]	Both	X	✓	X	X	U	Embedded systems	No information
[135]	Both	X	✓	X	X	U	Embedded systems	~100 devices
[136]	Traditional	✓	X	X	X	F, V	Firewall and VPN gateway	No information
[137]	Traditional	X	✓	✓	✓(logic programming)	F, I	Firewall, IDS	No information
[34]	Traditional	X	✓	X	✓(ILP)	U	Firewall, IDS, VPN	No information
[62]	Virtual	X	✓	X	✓(ILP)	U	All NSFs	No information
[29]	Virtual	X	✓	X	✓(ILP)	U	All NSFs	~20 functions
[138]	Virtual	X	✓	X	X	U	All NSFs	No information
[139]	Virtual	X	✓	X	X	U	All NSFs	No information
[51]	Virtual	X	✓	✓	X	U	All NSFs	No information
[72]	Virtual	X	✓	✓	✓(MaxSMT)	U	All NSFs	~100 firewalls

U = User-specified policies, A = Access control configuration, F = Firewall configuration, V = VPN configuration, I = IDS configuration, N = Network addresses, T = Network traffic.

describes an architecture that performs automatic intent-based provisioning of a security service in a multilayer network. This operation is performed by using an SDN orchestrator developed on top of the Open Network Operating System controller. The intents discussed in this article are only related to encryption requirements, and they might not be sufficient to specify more complex requirements (e.g., based on mutual authentication mechanisms or key exchange protocols).

Finally, References [53, 54] focus on methodologies for refactoring an existing security service to fulfill the input requirements. Reference [53] exploits Nile, a high-level comprehensive intent definition language, for the specification of the security intents that must be achieved in the service. After they have been formulated in a human-readable representation, a refinement process establishes which NSFs should be added to the service and in which position (i.e., between which pair of already present functions) so as to fulfill the intent. Instead, Reference [54] defines a technique for designing security chains based on Markov models, i.e., learning finite automata. In particular, with the aim to minimize the total number of security functions in the service, two algorithms are presented: The first one identifies multiple functions in the same chain that could be replaced by a single one, whereas the second one searches for different chains that could be refactored into a single one. Both References [53, 54] do not apply when the full service should be created from scratch.

5.2 Automatic Service Composition in NFV-based Networks

Given the importance that NFV achieved in the networking field since the early 2010s, most of the approaches for automatic service composition—including some based on SDN—exploit its virtualization principles.

As discussed in Section 2, the introduction of NFV into the techniques for automatic network security configuration has enabled the introduction of several optimality criteria. Despite this statement, some authors [35, 55, 56] do not formulate the problem taking optimization objectives into account, but they focus on other features. More precisely, Reference [55] proposes an automated mechanism that, starting from requirements expressed in controlled natural language by business-level operators, automatically generates security service graphs based on them. The engine requires a repository of VNFs from which it chooses the needed ones by matching their capabilities with the fields of the requirements themselves. The approach has been further extended by the authors in Reference [35]. In this case, when the needed VNFs must be selected, the k-means clustering algorithm is invoked, so that groups of VNFs are created according to the level of security they can provide. For example, if a medium level of security in the detection of attacks is required, then an IDS from the cluster labeled with "medium security" will be selected. This action can improve performance, because the choice for the refinement of each intent is thus restricted to a smaller set of functions. Another approach [56] proposes a function composition algorithm based on a Trie tree, which finds a security service composition that meets user's requirements. This approach shows high flexibility, because it can manage the IP addresses of the Virtual Machines automatically, so addressing the problem that, in cloud environments, IP addresses often change.

Let us now analyze, instead, the consistent number of other approaches [29, 57–62] that aim at designing network security services by fulfilling, at the same time, some optimization criteria.

In References [57–60], some heuristic strategies have been explored to minimize the total hardware and power resource usage. In virtualized environments and cloud scenarios, this purpose is evidently reached by minimizing the number of VNF instances installed in the network, since each one requires some resources. In particular, Reference [57] proposes the APPLE framework, in which each security policy is a sequence of security functions that each kind of traffic flow must traverse. APPLE achieves the aimed objective by creating a sequential ordering of functions that satisfies all the specified policies: In this way, the same instance of function could be present in more flow paths. The heuristics described in Reference [58], instead, after receiving multiple requests of generating security chains, refine these requirements by establishing a single combination of functions that consumes as few network node resources as possible. This algorithm is based on a greedy iterative approach: At each iteration it considers a possible function combination and it gives priority to security services where the composing functions have the maximum total throughput. In Reference [59], a novel heuristic based on a breadth first search is proposed to reach near-optimal solutions in polynomial time. This algorithm consists of two steps: First, the functions needed to enforce the policies are identified; then, they are composed by constructing the breadth first search tree and by minimizing the objective function, which considers parameters such as CPU, storage, bandwidth, and latency. Instead, in Reference [60] the heuristic algorithm is based on the partitioning concept: Instead of computing the full topology at the same time, the network is divided into partitions, and the problem is solved for each partition independently. This approach provides great scalability, while guaranteeing that the ordering constraints between the NSFs are still respected. The last two studies [59, 60] also introduce an ILP formulation of the problem but only to have a reference to assess the performance of the proposed heuristic.

Other approaches [29, 61–65], instead, propose to use ILP formulations rather than heuristics. Reference [61] describes a formulation based on the creation of an augmented graph: Considering all the security requirements related to the service design, all the possible instances are introduced in this virtual topology, the augmented graph, with all the possible interconnections. Only a subset of instances and connections will be actually present in the final service, in accordance with the objective function, which aims to minimize the number of VNF instances. Reference [62] describes an optimization engine, called Policy Manager, which exploits some policy refinement techniques to identify the NSFs that can be used to enforce the policies. The selection is typically based on a tradeoff among different criteria, such as cost, performance, reliability, or reputation of the different functions. However, additional usage profiles are provided to the users by means of which some parameters can be assigned greater importance than others when selecting the functions. The constraints about the profiles and the objective function are represented with a **Mixed Integer Linear Program (MILP)** problem, whose variables can be discrete (i.e., they can take a limited number of possible values). A further improvement of this methodology has been later presented in Reference [29]. In this case, the key role is covered by a Security Awareness Manager, which is in charge of the policy refinement process, but it provides also additional features with respect to the Policy Manager. For example, it supports dynamic adaptation to network changes, so that the security policies are kept enforced even when a security function fails. Considering the larger number of optimization parameters taken into account (e.g., user rating, experts trustworthiness expectations and security evaluation), the problem is formulated with a multi-objective approach. Reference [63] reaches an optimal construction of service function chains proposing an abstraction of multiple categories of security demands, so as to summarize them with a numerical value named security level of the chain. The ILP problem that is formulated aims to maximize not only parameters related to physical resources, such as CPU capacity and utilization time, but also the security level itself. Reference [64] proposes a novel abstraction of virtual network security functions, called functionality, which extracts the essential configuration parameters of each function in a vendor-independent representation. As better discussed in the extended version of this approach in Reference [65], this abstraction allows disjoining service composition from its deployment in the physical infrastructure, because functionalities can be used to compose the virtual service before establishing which VNFs are needed to enforce the security requirements and how they should be configured.

Other studies address the automatic fixing of an already-deployed security service in the NFV context, too. Reference [66] defines four operations that can be applied in the refactoring process:

separating a security service into multiple ones, chaining VNFs into a single structure, merging the unnecessary VNFs to optimize system resources, reordering the VNF organization. Reference [67], instead, proposes a dynamic defense provisioning mechanism, where a single IDS can be broken down into separate light network functions, each one in charge of detecting some attacks at different levels of the ISO/OSI stack (e.g., from packet header inspectors to deep packet inspectors).

All the works analyzed so far concern either the automatic composition of the full network service from scratch, including the security functions, or its refinement, by changing a previously generated service. There are other studies, such as References [36, 68, 69, 71], that address the different scenario of a network administrator who wants to enforce security requirements on an already defined network service, which does not yet include security functions. For example, the service could be exclusively made of network functions such as NATs, web caches, and load balancers but not NSFs. In this specific situation, the administrator may want that the service topology is kept and not changed, i.e., all the service functions should be crossed by the traffic flows as forced by the original design. In this case, the security policies are enforced by just allocating some NSFs in the existing topology.

This problem is more complex than the previous one, mainly because the presence of the preexisting middle-boxes must be considered when deciding the allocation scheme of the NSFs: Each network function can, in fact, have a behavior that could impact on the enforcement of the security policies themselves. A trivial solution such as allocating NSFs between any pair of network functions, even though potentially correct, would consume too many resources, would be inefficient, and would increase the configuration work. For this reason, solutions that try to optimize the way the network service is enriched with NSFs have been proposed.

Reference [68] solves the so-called firewall placement problem by identifying how firewalls should be placed in a network topology so that the maximum firewall rule set, which would be needed to satisfy the input security policies, can be minimized. Since this problem is NP-complete, a heuristic algorithm based on a variant of the shortest path algorithm is exploited to approximately achieve this optimization goal. Another approach [36] automates the generation of the allocation scheme for access control devices with an optimized and formal approach based on the definition of an iterative **Satisfiability Modulo Theories (SMT)** problem.[3] The basic idea is that, at each step of the algorithm, the access control architecture is tuned until all the security policies are properly enforced by the achieved result. With respect to Reference [68], the optimization criterion is, in this case, the minimization of the access control elements allocated in the network. In both these papers the presented methodologies are general enough to be applied to both traditional and virtual networks. A more recent approach [69–71] proposes a definition of the firewall allocation problem as a **Maximum Satisfiability Modulo Theories (MaxSMT)** problem.[4] In this case, formal correctness of the computed solution is achieved with a correctness-by-construction approach, thus avoiding an *a posteriori* formal verification and speeding up the overall process. The optimality criterion that is considered is the minimization of the number of allocated firewalls. Moreover, with respect to the other two works, in this case not only the allocation scheme but also the firewall configuration rules are computed. Hence, the algorithm can represent a complete proof-of-concept of an automatic network security service generation, albeit limited to firewalls only. Some ideas about how to extend this approach for the composition of services with multiple types of security functions have been discussed in Reference [72], where the MaxSMT formulation is again presented to design a service with the minimum size.

[3]An SMT problem is the generalization of the traditional Boolean satisfiability problem. The main difference is that additional theories, such as integer or string theories, can be used in the problem formulation.

[4]With respect to an SMT problem, MaxSMT is an optimization-enhanced version where some constraints, which do not require to be always satisfied to achieve a correct solution, represent the optimization goals.

6 AUTOMATIC NETWORK SECURITY FUNCTION CONFIGURATION

If automatic network security service composition is a novel research path scarcely investigated in the past, then automated methodologies for configuring NSFs have been proposed for years, even though recently this theme has made a strong comeback. The main reason is that the former task is intrinsically inherent to the recently emerged network softwarization paradigms, whereas the latter had been already investigated for traditional networks with security-related middleboxes. Consequently, a larger number of approaches has to be analyzed in this section. A complete overview of them is provided in Table 3. The meaning of its columns is again explained in Table 1. According to the taxonomy depicted in Figure 1, the automated methodologies that have been proposed in these papers deal with different kinds of NSFs. This consideration is reflected by the structure of this section, where each subsection focuses on a specific NSF type.

6.1 Automatic Firewall and Access Control Configuration

A first class of network security functions is represented by the functions that can decide if a traffic flow with certain characteristics is allowed to continue to its destination or if it must be blocked. In this category, the function that has been most investigated is the packet filtering firewall (i.e., a firewall that can analyze fields of the IP 5-tuple), because this kind of function is sufficient to protect an end-to-end service in most situations, and, at the same time, it requires a much simpler configuration than what is needed by other security functions. In this class, nevertheless, we will consider also traditional functions, such as filtering routers, that can be configured as access control devices. The reasons of this classification choice are that most of the methodologies share the same concepts, since firewalls themselves are exploited for access control, and these functions themselves take decisions that are mostly based on the same information.

The oldest work that we found in this category is dedicated to filtering routers [75]. It discusses the possibility to compute a set of filters for the individual routers of a distributed networked system, so as to enforce a global network access control policy. However, being the first proposal in this area, this methodology is lacking from several points of view. First, the abstraction level at which it works is really high, so that the actual configuration of access control devices would require an additional refinement. Second, the filtering rules that are produced are not guaranteed to be optimal. Finally, an algorithm to check the consistency of the computed configurations is presented, but it is not based on formal verification techniques; consequently, it does not provide a full formal correctness assurance. Despite all these limitations, this article opened the path to other similar research works in the immediate next years in a period that is largely antecedent to the advent of network virtualization.

A framework mainly targeted to traditional small-sized networks was proposed few years later, and it became a milestone for access control automation: Firmato [33] is a firewall management toolkit that can automatically compute a firewall configuration so as to enforce a set of global security policies into a network. The presence of a model compiler allows the framework to provide an abstract representation of the output configuration with respect to the specific setup of each firewall. Besides, with the aid of an additional module called Fang [140], a human being can easily interact with the automated framework through a query-and-answer session, by means of which she can find out if a global policy is correctly satisfied by the enforced firewall rules. This work has become so important in this field, because it has been the first proposal of an automatic firewall configuration, based on an abstract and vendor-independent representation. Although designed for distributed firewall architectures, Firmato was validated by considering a single border firewall.

After this first work, other proposals tried to overcome its limitations, at the same time providing additional features. The next papers that appeared after Reference [33] are References [74, 76, 77]. In Reference [74], concrete firewall configuration rules are derived from high-level network security policies by exploiting an *Organization Based Access Control* model. The operation that is performed is actually a translation, more than a refinement, just providing abstraction from firewall implementations. The methodology proposed in Reference [76] exploits a framework for policy-based management, called STRONGMAN [141], to automatically establish and enforce local access control rules that are compliant with high-level global security policies: The compliance checker module can, in fact, compose the rules into a coherent enforceable set for each device. Finally, FACE [77] is a framework that can automatically analyze and generate the rules for a distributed firewall, so as to satisfy a filtering policy expressed with a threat model where the attacks for which a protection must be enforced are defined. These works apply policy-refinement to a distributed access control architecture, even though only Reference [76] really validates the approach in a distributed system, while Reference [74] validates it with a single firewall and in Reference [77] no effective validation is showed. At the same time, none of these works uses formal verification techniques.

Formal assurance of configuration correctness was introduced in References [78–81]. Specifically, Reference [78] exploits formal logic programming methods to model real-world situations, where firewall rules are automatically generated by a reasoning engine so that they not only enforce security policies, but they are not affected by anomalies such as redundancies or contradictions. This approach, however, is limited to configurations that are compliant exclusively with the syntax of IPChains and Cisco's PIX. The methodology illustrated in Reference [79], instead, automatically refines a conflict-free set of access control policies into distributed rules; the consistency and correctness of the access control list implementation are then formally verified with respect to the original policy model by exploiting a Boolean satisfiability problem formulation. B-Method, an approach based on formal methods applied to abstract machine notation, is instead used in Reference [80] for the enforcement of security policies through a simplified refinement where the information composing the abstract notation is derived from the external network environment. Finally, the technique described in Reference [81] is based on formal argumentation and preference reasoning, exploited to both analyze and generate firewall rules from high-level policies. Even if the rule ordering is not specified in the high-level policies, the correct ordering of the firewall rules is automatically computed by means of abductive reasoning.

When network virtualization became dominant in research, a number of papers [82–88] started investigating methodologies that can be applied to virtualized networks. Indeed, most of these methodologies can be applied to traditional networks, too, the only exceptions being References [82, 87, 88], that propose, respectively, an automatic approach for computing Netfilter configurations through the user specification of unordered policies [82], a method to automatically define the rules for iptables, to ensure access control protection in a large-scale synthetic power system [87], and a method that applies to three UNIX firewall systems (iptables, ipfw, pf) [88].

Coming to the approaches that can be applied to both traditional and virtualized networks, Reference [83] addresses the lack of a formal semantic to distribute multiple security policies in the access control middleboxes of a network by designing an automated refinement process, based on algebraic requirements. This approach also supports a formal verification step. Another framework, proposed in Reference [84], is SyNET, which is based on a correctness-by-construction approach. The synthesized rules for access control devices are computed as the output of a Datalog program, which contains all the information about both the network topology and the security policies. An alternative paradigm, proposed by the same authors in Reference [85], is the NetComplete framework, which, like SyNet, is based on an SMT problem formulation, but it avoids to model policies as constraints of the SMT problem if they can be solved and verified in an alternative way, and it also exploits domain-specific heuristics, such as partial evaluation, to reduce the solution space, so improving the overall efficiency. Another comprehensive approach for access control policy refinement and formal verification is illustrated in Reference [86]. The designed methodology can derive the proper access control configuration from security policies, but it can also fix an existing one if it does not correctly enforce the user-specified requirements.

Firewall and access control configuration proves to be a thriving research area at the beginning of this new decade, too, with a new range of studies [69–71, 89–92]. References [89] and [90] aim to improve the scalability of automatic access control configuration. The former applies machine learning on the operator-provided feedback, while the latter transforms the user-specified policies into a representation named *priority-based domain type enforcement*, which considerably reduces the complexity of policy specification and therefore its impact in terms of performance. Instead, References [69, 70, 91, 92] apply formal methods to ensure configuration correctness. Reference [91] achieves this goal by using preference-based argumentation reasoning, so as to identify all the possible conflicts and anomalies among the user requirements before refining them into the firewall rules. Reference [92] shows that modeling the user-specified policies with the metagraph algebra leads to reduce the problem complexity, as policies are decoupled from implementation-specific intricacies of low-level middleboxes. Finally, Reference [69], already presented among the approaches for automatic composition of network security services in Section 5, proposes a framework that can also automatically compute the configuration of a distributed firewall architecture as the solution of a MaxSMT problem, which also provides formal correctness assurance and optimization, such as minimization of the cardinality of the firewalls' rule sets. This approach has been recently extended in Reference [70], where an algorithm for the computation of the traffic flows to be later modeled in the MaxSMT problem allows achieving better performance. An alternative way to model packet classes in a MaxSMT problem has been preliminarily investigated in Reference [71]. In that study, each packet class considered for traffic flow computation is the minimal one and is disjoint from the others. Then, each class can be associated with an integer number and fewer variables are included in the problem formulation, thus increasing the overall scalability.

All the papers discussed so far exploit automation to compute firewall configurations from scratch, assuming all the access control boxes are just placed in the service but are not yet characterized by any filtering rule. However, a consistent group of papers [94–102] focused on automated reconfiguration, with the purpose of fixing an existing configuration that is not compliant anymore with a new set of network security policies or which is affected by some policy anomalies. Even though at first sight this capability could seem more limited than the computation of a full configuration, it represents a crucial component of the full workflow of an automated network configuration, described in Section 4. Whenever a new security requirement is identified by the user (e.g., a server has been recently victim of an attack and the access to this host must be prohibited, thus avoiding any interaction that could make the attack propagate in the network), the reconfiguration of the security architecture should be as fast as possible. If the rules of each function must be computed from scratch each time the user adds or modifies policies, then the required time could soon become unacceptable, although shorter than that required by a manual intervention.

Looking more closely at the methodologies belonging to this class, Reference [94] designed a policy engine that can perform an automated reasoning on vendor-independent network function models so as to compute new configuration parameters every time the need to change them is brought over by the violation of a security policy. The followed approach has an iterative nature, because the policy engine generates a compliance test for each policy, executes all of them, and derives new configurations for a network partition; if compliance is not yet reached, then the process is then repeated for another partition. The technique presented in Reference [95], instead, targets the firewall reconfiguration problem by applying a reduction algorithm to the result represented with a *Firewall Decision Diagram*, while maintaining its consistency and completeness at the same time. Another approach [96] derives anomaly-free rule sets by analyzing the relationships between the filtering rules of each firewall, searching for coincidence, disjunction, and inclusion phenomena in the condition attributes of the rules. Following an approach similar to References [95, 96], Reference [100] illustrates five algorithms that can reconfigure a faulty firewall configuration after incurring in one of the five corresponding issues (wrong rule order, missing rules, wrong condition predicates, wrong decision actions, wrong extra rules). However, Reference [97] proposes a change management process, based on changeability analysis on a Service Graph, which recomputes the access control configuration to make it consistent with end-to-end connectivity requirements. This work was limited to a specific type of security requirements, while Reference [98] considers additional classes of constraints, such as reliability and performance requirements. Finally, two other papers exploit formal verification to provide the administrator correctness assurance of the achieved results. Reference [101] describes the design of the Control Plane Repair algorithm, based on a MaxSMT formulation, which also minimizes the number of configuration changes by using an abstract representation of the control plane semantics. Reference [102] uses a dedicated calculus to formally verify if the access control configuration is compliant with the security policies and, if not, to automatically generate the optimal configuration repair.

6.2 Automatic SDN Switch Configuration

An SDN switch is an access control device that works in symbiosis with an SDN controller. The filtering rules can be lowered on a switch in a proactive way, previously established by the administrator. However, the optimal working mode is the reactive one, where an automated controller computes configurations by itself and changes the rules when needed, e.g., when a new security policy must be enforced or an attack has overcome all the barriers. The main differences with respect to traditional access control devices are that (i) a full framework for SDN switches management is not needed, because an application integrated within the controller is sufficient, and (ii) reaction time is typically lower, because most of the operations are performed by software.

A first problem addressed in literature about automation of SDN switch configuration is the need of high-level languages to specify policies in such a way that security requirements can be independent from vendor-specific switch implementations. This challenge was faced in the early years of SDN, before the advent of the first automated configuration techniques, which inherited the most useful principles of these original languages. The milestone languages are Ethane [142], Frenetic [48], and PolicyCop [143]. Ethane [142] allows administrators to write high-level access control policies with a flow-based language, while Frenetic [48] is a programming language specifically developed for OpenFlow networks, is able to express both user-specified policies and policies for automatic reaction to events. Instead, PolicyCop [143] is a language for the specification of service level agreements within Openflow. A more extended dissection of specification languages, which would be out of scope for this article, is provided in Reference [144].

After this preliminary research phase, OpenFlow has been used to develop automated methodologies for configuration (and reconfiguration) of SDN-based networks [47, 103, 105, 106]. The common ground of these techniques is the automatic enforcement of user-specified policies, expressed with a user-friendly language, in a distributed SDN switch architecture, providing abstraction between policies and filtering rules. Each of them has some significant exclusive peculiarities described below, and there is no dominant work in this group overshadowing all the others.

CloudWatcher [103] can define the optimal (i.e., minimum) route for each traffic flow such that it crosses some nodes dedicated to intrusion detection. In fact, the main goal is that such traffic flow is inspected, thus enabling automatic reaction. However, the reaction can be performed only by means of external modules, developed by the cloud administrator, since internal reactive algorithms are not provided. Procera [105] is a framework based on functional reactive programming, inspired by the Frenetic language. Its main peculiarity is the simplification of event reaction, by the definition of reactive policies that capture all the information needed to enforce security constraints after a specific event. However, a limitation is the complexity of the language used for policy specification, since it is similar to a programming language and less user-friendly than the others. Fresco [47], already presented among the approaches for automatic composition of virtual network security services in Section 5, also includes a number of modules that can be exploited and combined to create and then enforce network security policies. A great advantage of this approach is that each policy is not independent of the others, but they can share the enforcing modules. As for Procera, user friendliness is not very good, because each policy is similar to a code script. OpenSec [106] is characterized by an internal mechanism to offer event reaction and a more user-friendly policy language. Nevertheless, with respect to CloudWatcher, optimization is missing, whereas, with respect to Fresco, each policy is independent and isolated from the others.

The four papers just discussed [47, 103, 105, 106] represent the core of automation for SDN switch configuration. However, another group [104, 107–110] is worth being discussed, since they present innovations that go beyond the principles on which CloudWatcher, Procera, FRESCO, and OpenSec are based. Reference [107] describes the architecture of a framework to enforce security policies on SDN communications that introduces the innovation of a Policy Manager that can be both intra- and inter-domain, thus showing the feasibility of this approach for big multi-domain scenarios. An extension of this work [109] introduces the presence of Switch Security Components, i.e., enforcement mechanisms directly residing in the switches rather than in the SDN controller, where only the orchestrating software, called Security Management Application, runs; this novelty is particularly suitable for a proactive prevention of the attacks from malicious end points connected to the network. Instead, Reference [108] proposes a methodology, based on a MILP formulation, to optimally update the configuration of an SDN-based network whenever the waypoint traversal of traffic flows must be enforced. The optimality criterion is the minimization of the number of rounds through which the traffic can reach the specific waypoint. Reference [104] defines the MTDSynth framework, which exploits moving target defense techniques to automatically enforce agility specifications, i.e., the definition of which security actions must be applied in reaction to specific mutation events. Finally, Reference [110] formalizes security policies with multi-attributed graphs to easily remove the conflicting flow rule from the switches and automatically install the new ones requested by the user.

6.3 Automatic VPN Gateway Configuration

Another category of network security functions is represented by communication protection devices that can be used to enforce policies about how traffic should be protected when crossing the network. The two main security properties that are dealt with are data integrity (i.e., the guarantee that the correctly received data have not been modified by the middle-boxes with respect to the ones originally sent by the source) and source authentication (i.e., the guarantee for the destination that the source of communication is the expected one). The most frequently used technology is IPsec, often adopted for creating VPNs, i.e., networks that, although crossing public or non-trusted nodes, are protected and made private by security mechanisms. When using this mechanism, VPN gateways are the security functions used to enforce the required communication protection by creating secure IP tunnels.

As for firewalls, also for VPN gateways a milestone paper can be identified in the research area of automatic configuration: Reference [111] was the first paper that proposed a clear classification of the user-specified requirements for secure communication. These requirements are divided into four categories: (1) access control requirements for restricting access only to trusted traffic;

(2) security coverage requirements to define which security algorithms should be applied to a specific type of traffic; (3) content access requirements to establish which network nodes can have visibility of the decrypted traffic; and (4) security association requirements, related to the set of Security Associations, i.e., sets of shared security attributes between two network nodes. Given these four requirement types, the authors propose three different algorithms for computing the configuration of VPN gateways. These algorithms have been the base of all the other ones developed later. In a nutshell, the first strategy, called direct approach, simply creates a VPN tunnel for each security coverage requirement. It is very efficient, because it is a mapping between requirements and function rules. However, it can lead to incomplete solutions, because it may fail in enforcing a security requirement if a one-to-one relationship with a VPN tunnel is not enough. A second possibility is the bundle approach, which consists of grouping traffic flows into disjoint sets, for which the VPN rules are created: The solution that is reached is always complete and correct, but the strategy is less efficient and scalable. Finally, a combined approach is proposed as a tradeoff to overcome the weak efficiency of the bundle approach and the lack of completeness of the direct approach.

One issue with this proposal is that, with any algorithm proposed in Reference [111], the number of automatically generated VPN rules is often greater than necessary. With the aim to overcome this limitation, the author proposed a fourth algorithm in Reference [112], called ordered-split approach. With respect to the other three ones, this strategy is optimized, because it builds the minimum number of VPN tunnels to satisfy the user-specified requirements. This objective is achieved with a solution based on the traditional “task scheduling” algorithm.

All these concepts were then exploited in Reference [113] for the description of a distributed and automated architecture, called BANDS. The main purpose is to apply the algorithms proposed in Reference [111] or Reference [112] to an inter-domain environment, where VPN tunnels cross multiple Autonomous Systems. The increased complexity is managed by discovering which path of Autonomous Systems each traffic flow must cross in the tunnel and by the activation of a negotiation protocol, through which each Autonomous System can negotiate the VPN policies with the others. This negotiation is performed automatically, for each Autonomous Systems, by a server specifically in charge of this task.

In the subsequent years, other authors studied the automatic configuration of VPN gateways by proposing alternative methodologies for the computation of rules or improving the features of the existing ones. Reference[114] proposes an algorithm to automatically generate conflict-free VPN rules, with the aim of solving all the possible conflicts that could arise from an incorrect specification of the requirements. Their strategy is based on an iterative approach: After each requirement is mapped to a specific VPN rule, at each step the policies are ordered by decreasing length and possible overlapping or redundancy anomalies are managed starting from longest tunnels. Instead, the algorithm described in Reference [115] aims at automatically generating the policies by exploiting recursive binary trees. These data structures allow previously created rules to be reused to satisfy new requirements by only adjusting their selectors, thus avoiding the creation of new rules and improving efficiency. Reference [116] describes the design of a framework, based on a fully automated strategy to perform a distributed configuration of IPSec domains, including scenarios such as nested networks or mobile environments. With respect to the previous approaches, this algorithm reaches good robustness against potential failures, high scalability in terms of VPN gateways, and the agility needed by mobile IPsec devices. Instead, Reference [117] proposes an efficient approach to solve conflicts in VPN policies by creating new rules that are free from any identified issue. The main limitation of this work is that, with respect to the previous ones, it cannot create the configuration of a VPN gateway from scratch, but it works on previously established rules.

All the approaches discussed so far for VPN configuration can be applied only to physical networks. Two studies that went beyond this limitation are Reference [118, 119]. The former designs a hybrid SDN architecture that allows network devices from different providers to be connected through VPN tunnels configured automatically. The latter uses a MaxSMT formulation to ensure that the automatically computed VPN configuration is compliant with the user demands. Independently of the technical methods, both these approaches show that automating VPN configuration is a relevant state-of-the-art problem in modern networks, too, also because of the large variety of available VPN technologies, which may make manual decisions slower and more error-prone.

6.4 Automatic Configuration of Embedded Devices

New security issues arise in managing the configuration of embedded devices, such as those in cyber-physical systems. The heterogeneity, pervasiveness, and distributed nature of embedded devices make their manual configuration much more error prone than what it is for traditional networked systems. This peculiarity can be observed especially with IoT devices, due to new security challenges that arise in protecting communications between constrained devices [145]. For example, privacy and data protection requirements add to access control constraints, and they should be treated with a high degree of flexibility in compliance with the adaptation and self-healing feature of IoT infrastructures. Therefore, expressing and refining security policies to govern distributed embedded systems (such as IoT) is a difficult task, because, in distributed architectures, complex processes are carried out by several interacting entities. This complexity is shown in the proposal of a policy language for distributed systems, called *Hierarchical Policy Language for Distributed Systems* [146]. This language enables expressing the relationship between an abstract policy, related to the whole network, and its distributed implementations at different locations through reference monitors that control the flow of information among distributed devices.

Following the proposal of this policy language, early attempts have been made for automatically refining security policies related to embedded devices, especially in IoT environments [120–123]. The enforcement solution proposed in Reference [120] is a security toolkit, named SecKit, which cooperates with distributed systems via the MQTT protocol, a widely adopted technology to enable lightweight communications among constrained devices. The MQTT broker is extended with the capability of enforcing policies, concerning authorizations and obligations, in compliance with internal meta-models. As this was the first proposal for IoT devices, it has some drawbacks: All the enforcement operations are executed at the broker level, so this may hinder the safety and efficiency of the whole system. Instead, the idea behind the work illustrated in Reference [121] consists in the integration of an existing flexible and distributed IoT middleware, named **NetwOrked Smart objects (NOS)** [147], with a security-aware policy enforcement framework. This extension of NOS is in charge of handling access control and service provisioning under security and quality requirements, e.g., to protect data resources and user sensitive information. A similar approach was pursued in Reference [122], where the security policy refinement mechanism thought for generic IoT environments is cast into networked systems for smart health. There, security policies concern the provision of authentication and authorization (e.g., nurses cannot access to sensitive data related to the doctors), or the anonymity of personal information. Instead, Reference [123] designs an automated framework, named ARCADIAN-IoT, for the holistic management of security policies related to multiple relevant properties for IoT-based networks, such as trust and transparent, user-controllable, and decentralized privacy.

After these initial studies, other ones [127–131] have introduced formal verification to improve the automation of embedded devices configuration. In greater detail, Reference [127] aims to reach a conflict-free enforcement in multi-administrative IoT environments, throughout a vendor-independent graph-based policy specification mechanism. Reference [128] challenges the auto-configuration problem for a peculiar class of embedded devices, smart meters, which forms an Advanced Metering Infrastructure, interconnected along with heterogeneous cyber-physical components transmitting usage reports or control commands between meters and the utility. An SMT problem is formulated to synthesize resiliency configurations for the smart meters, so as to enforce, in a provable way, security requirements, operational integrity invariants, and robustness constraints. Reference [129] uses a MaxSMT formulation for modeling the auto-configuration problem in IoT-aware networks. The proposed mechanism is tailored to solve specific problems of IoT environments, such as the simultaneous presence of numerous interconnected devices and strict latency requirements. Reference [130] defines a tree search-based algorithm that looks for potential alternatives of embedded devices configuration through a deterministic search process, and it proposes a verification mechanism to validate that all the threats described in the security requirements are successfully mitigated in the generated configuration. As discussed in Reference [131], this approach has been later improved with the inclusion of the full MITRE ATT&CK taxonomy [148], so that the proposed algorithm can also use its extensive knowledge to formally verify system design security characteristics.

Some other approaches propose to adapt a particular class of policies, named sticky policies, to the needs of IoT systems [124–126]. According to the original definition in Reference [149], the main idea behind the concept of sticky policies is to attach security and privacy policies to owners' data to drive access control decisions and policy enforcement. In the extension of NOS proposed in Reference [124], each user can establish their policies on data, respecting the constraints defined by a trust authority, and attach them to the data themselves. After receiving the data, NOS can regulate access control by means of the attacked sticky policies. Similarly, in Reference [125], IoT users are enabled to personally enforce their privacy preferences. Thus, the policy enforcement mechanism relies on privacy metadata (e.g., privacy preferences, data categories, and data history), generated by smart objects owned by the users themselves. This idea is further improved in Reference [126], where a permissioned blockchain mechanism in introduced in NOS, with the aim to protect the sticky policies, which regulate the access to IoT resources, against tampering and violation. In all three studies, embedding the enforcement mechanism into the devices leads to some overhead, but it is compensated by the absence of a single point of failure.

Automating IoT security configuration has been also a central research topic in a recent EU H2020 research project, ANASTACIA [132]. The project created a framework, composed of distributed security components and enablers, that can dynamically refine user security preferences into the configuration of cyber-physical systems and IoT architectures. To this end, it also encompasses online monitoring to allow more automated adaptation of the system, with the aim to mitigate unexpected security vulnerabilities. In the frame of this project, multiple works have been carried out [42, 133, 134]. In Reference [42], different levels of abstractions for the IOT security policies are investigated, so as to enable a technology-agnostic refinement process. Security policies are refined in two steps: First sentences expressed in natural language are translated into technology-independent representations, and then those representations are refined into the effective configuration of each IoT device. This study has been enriched in Reference [133] with a logic formalism based on rule reasoning and the Semantic Web technology. The former envisions the usage of reasoners to infer new knowledge, not explicitly defined by human users, whereas the latter enables formal verification of the entire automated configuration process. Thus, their combined usage allows detecting conflicts that may appear in the IoT security policies or in the refined configurations. Finally, these ideas are applied to the automatic configuration of highly interactive honeypots in Reference [134]. Pro-active security policies are defined by the users to configure monitoring agents, which feeds the reaction part of the framework, and the monitored data are processed to generate and provide a set of countermeasures as new security policies, so that the configuration process for the IoT network can be repeated.

Finally, a pair of studies [43, 135] investigate how the problem of automatic security configuration can be addressed for IoT systems located in smart homes. Both pursue user-centered design, so as to include usability goals and user characteristics in the design of their solutions. On the one hand, Reference [43] discusses a policy harmonization technique employed in the policy refinement process, so as to reconcile policies specified by different people living in the same home and solve inconsistencies transparently. On the other hand, Reference [135] focuses on the refinement of attribute-based access control policies, allowing automatic decision-making processes based on environmental attributes like time of day, and on the location from which the attempt to access an IoT device originates.

6.5 Automatic Configuration of Heterogeneous Security Services

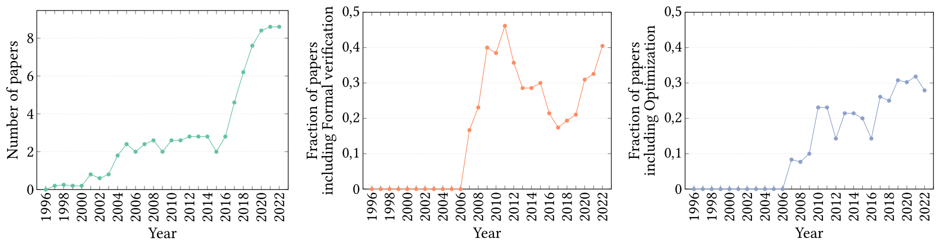
All the works surveyed so far in this section deal with a single kind of NSF (packet filter, SDN switch, VPN gateway, embedded device). Consequently, they can be applied under the assumption that either the security service is composed of that single function type or all the other security functions have already been configured. Few researchers addressed the case when this assumption is false, i.e., an automatic configuration of heterogeneous services is necessary, including different function types configured all together or iteratively, according to the adopted strategy.

The first work in this area is Reference [136], which defines formal models of a large range of security functions and exploits them to check whether a set of security goals is achieved in a given network description. Indeed, this approach cannot automatically compute the configuration of the security devices from scratch, but it can identify possible violations of the security policies and recommend which devices should be reconfigured to eliminate the identified issues. Hence, this strategy requires that the functions are already configured before applying it. Despite this limitation, it can be considered as a first step toward fully automated configuration of an heterogeneous service.

After this initial verification-oriented attempt, Reference [137] provides the first joint automatic generation of configuration rules for two different types of security functions: firewalls and IDSs. Correctness assurance of the computed results was also achieved, by defining the problem through a constraint logic programming language. Additionally, the computed configurations are optimal with respect to a specific cost function, according to which the rule configured for enforcing a specific policy should be placed so as to minimize bandwidth and packet drop rate. Even though this work shows that a heterogeneous security service can be configured from scratch, firewalls and IDSs are configured at different stages of the algorithm, and only locally optimal solutions are reached.

MIRAGE [34] overcomes the limitations of the previous work by reaching a simultaneous top-down refinement of global security policies into configurations of three different kinds of functions: firewalls, IDSs, and VPN gateways. It can also perform a bottom-up analysis of already deployed network security configurations to guarantee correctness and consistency: This analysis works both at intra-function and at inter-function level. However, if any anomaly is detected, then no recovery mechanism is provided. Nonetheless, the optimality of this approach is achieved through the formulation of a linear programming problem, whose objective function is to minimize the number of used functionalities for the deployment of the rules.

Other approaches that can configure full network security services are References [29, 51, 62, 72, 139]. In Reference [62], the developed framework, called Policy Manager, automatically generates configurations for the virtual functions that have been selected in the previous step. This configuration process is made of two refinement phases: First, the function rules are expressed with an



(a) All papers (b) Formal verification (c) Optimization

Fig. 3. Time distribution of the studies (last 5 years moving averages).

abstract and vendor-independent language; then, a final translator adapts the rules to the syntax required by the deployed VNF. The functions that can be managed are packet filters, stateful firewalls, L7 filters, and basic content inspectors, whereas VPN gateways, proxies, and IDSs are not dealt with. This initial work has been further extended by the authors in Reference [29]. An important innovation is the proposed capability model, according to which each network security function is characterized by functionalities that are shared by other functions; for example, the packet filtering capability can be performed both by a traditional packet filter or by a web-application firewall. Consequently, the automatic configuration is targeted to the capabilities instead of the functions by introducing a further abstraction level. All the concepts presented in Reference [29] have been validated with the implementation of Security Awareness Manager, the first framework that can automatically manage both composition and configuration of a security service. An extension of this meta-model was later presented in Reference [138] introducing the possibility to configure a larger number of capability features. Another study [139] proposes an automatic data model mapper to automate the refinement of high-level user-specified policies. Instead, the methodology illustrated in Reference [51] pursues an approach similar to the one described in Reference [29], as it refines user-specified policies into NSF configurations after synthesizing the chain in which they must be combined. Finally, Reference [72] proposes an alternative methodology, where the configuration problem is stated as a MaxSMT problem, where a correctness-by-construction approach is exploited to provide formal correctness assurance without requiring an *a posteriori* verification. Furthermore, the MaxSMT approach is also exploited to achieve optimality criteria, such as minimization of the number of configured rules. However, with this approach, a reconfiguration would not be possible, since the methodology can only work on user-specified requirements from scratch.

7 DISCUSSION ABOUT RESEARCH QUESTIONS

As a final step of our survey, we answer the research questions that have been presented in Section 3, in light of the studies illustrated above. These answers are meant to provide guidance to the readers in following the trends emerging from the reviewed state of the art, so as to introduce further improvements in the network security configuration automation field.

RQ1 (Time distribution): Figure 3(a) shows the time distribution of the 95 reviewed papers from 1996 to 2022 by plotting the moving average number of papers per year, with averages computed on the last 5 years. According to this chart, a first peak of interest was reached in 2004, which is the publication year of Firmato, the first milestone paper on firewall configuration automation. This work inspired a consistent number of following works. Another consideration is that the average number of papers per year remained sustained in the years after 2004, with a more pronounced growth over the past few years. One factor that stimulated this growth was the advent of virtualization technologies, which shook the traditional vision of networking and became an enabler and a stimulus for the research on network security configuration automation. This analysis confirms the current interest by the scientific community for this topic.

RQ2 (Enhancing features): From the analysis of the collected papers, it resulted that two relevant features that are recurrently paired with automation to improve network security configuration are formal verification and optimization. These features are rarely achieved manually, as they would require excessive expertise and time, but they have been shown to be feasible within automatic methodologies. These features enhance the results that can be achieved substantially. For example, if an automated framework is seen as a "black box" that automatically computes an untrusted result, then the pairing of a formal verification method can increase our trust in the result correctness. At the same time, optimization can minimize costs and maximize performance. Figures 3(b) and Figure 3(c) plot the moving average of the fraction of papers enhanced, respectively, by formal verification and by optimization, versus time. This is computed as the ratio between the average number of papers enhanced by each feature in a certain year and the previous four years and the average total number of papers published in the same years, as reported in Figure 3(a). These charts show that, close to the beginning of the 2010s, there was peak interest in formal analysis and optimization techniques, and this interest is still high in more recent years. It is also interesting to notice that, among the 23 papers that incorporate optimization, the following strategies have been adopted: ILP was adopted by 11 papers, heuristics by 2 papers, MaxSMT by 7 papers, Constraint Logic Programming by 1 paper, Calculus by 1 paper, and Iterative SMT by 1 paper.

In response to the research question, this analysis highlights that the current trend in this research area is to enrich automation with as many features as necessary to generate high-quality security configurations, including formal analysis and optimization. However, it can be noted that, even if interest has been shown for both these features, they have been almost always included separately from each other. For this reason, their combination should be further investigated. Moreover, for each of the two features, more advanced solutions can be researched. For what concerns formal verification, *a priori* verification methods are an interesting path to be pursued, as already shown by some state-of-the-art studies, because they avoid time-consuming *a posteriori* verification. For what concerns optimization, nowadays globally optimal solutions can be reached in reasonable time thanks to advanced solvers of mathematical constraint programming problems. Therefore, in parallel to heuristics, these optimal approaches should be investigated, so that they can be applied particularly when the requirement of optimality is the most important one.

RQ3 (Limitations): The limitations that have been identified when reviewing the state of the art can be grouped into three main categories: (1) heterogeneity of the security services, (2) performance of automated techniques, and (3) full autonomy in network security.

Heterogeneity of the security services: Most of the network security services are heterogeneous, i.e., different functions are exploited to provide defense against cyber attacks. For example, an architecture exclusively based on firewalls would not offer adequate protection, since it is possible that some attacks are not blocked by the rules configured on the firewalls and reach their targets. Instead, if an intrusion detection system is installed behind a firewall following the *security in depth* paradigm, then this second line of defense would increase the possibility of detecting the attack, thus allowing a consequent reaction. However, this central characteristic of the network security services is not fully matched by the state-of-the-art approaches that aim to automate their configuration. As it has been illustrated in this survey, most of the automated methodologies for configuring a network security service work on a single function type, with packet filtering firewall being the most dominant one. Therefore, their applicability is limited to specific domains. Also the limited number of works aiming to automatically configure multiple function types are either still in progress or have limitations (e.g., they can be applied only to chains, instead of ramified graphs, or they lack optimization).

Overcoming this limitation represents an important research path that should be pursued in the future. In investigating how automation can be leveraged for the automatic configuration of a heterogeneous security service, several additional challenges must be addressed. First, given some network security policies describing the requirements to be fulfilled in a service, the correct and minimum set of security function types should be identified. This operation is easier than the automatic generation of the configuration itself, but it is a fundamental intermediate step. A match should be identified between the capabilities offered by the available security function types and the policy characteristics. If such a match is not feasible, then it means the available function types are not enough to enforce the requested security level. Second, after automatically computing the configurations, a correctness check must guarantee the absence of inter-policy anomalies. When a single function type is used, this check can be performed more easily, since all the function instances work at the same levels of the network protocol stack. With multiple types that can be instantiated in different points of the service, the complexity of this problem increases. Third, a more sophisticated interface for the deployment of the automatically computed configuration into the NSFs is needed, because each function type requires different parameters for its internal setup.

Performance of automated techniques: Computing the configuration of network security functions is a non-trivial time-consuming operation. This statement holds if the task is either performed manually by a human being or automated with techniques such as policy refinement. In the former case, a correct completion of the task may require days if we include the time taken to check correctness. For this reason, if automated methodologies that also guarantee the correctness of their output can perform such a task in lower magnitude orders of time, then that is already a significant positive outcome. From this point of view, many proposed solutions available in the literature can reach their goals in some minutes or hours in the worst cases (e.g., for big networks). Nevertheless, the relevance of dynamism in the management of modern computer networks is constantly increasing. In the context of network security, its impact is even bigger. For example, if the automatic methodology should compute a new security configuration after an attack has been detected, then hours of computation would not be acceptable. Besides, as reported in Tables 1 and 2, most of the state-of-the-art approaches can scale to computer networks with a limited size (e.g., up to 100 nodes). For this reason, improving the performance and scalability of automatic security configuration processes is another important future challenge. A good balance between heuristics and optimization models should be investigated, and the mitigation mechanisms should be reinvigorated with novel, fast techniques for automating the reconfiguration of security functions.

Achieving full autonomy in network security: Another limitation is that at the moment an approach that can fully automate the whole security workflow analyzed in Section 4 does not exist. Currently, each approach reviewed in this survey still needs interaction with a human being, e.g., to receive the security policies that must be fulfilled by the computed configurations. Research should overcome this limitation by investigating autonomic processes that would extract the information needed for policy refinement from the network itself, thus closing an action–reaction loop that would no longer involve external interventions. This would require the definition of intrusion prevention methodologies that would be able to perform the so-called policy discovery, i.e., extraction of policies from network monitoring. New algorithms based on machine learning and artificial intelligence could be defined to perform the autonomic reconfiguration of the security service whenever the statistics computed from the extracted information would characterize an ongoing cyber attack. Alongside with this achievement, a fully autonomic platform should be also capable of keeping safe all the service functionalities even in short periods where some security defenses have been temporarily compromised, until a full reconfiguration confines the danger.

8 CONCLUSIONS

The introduction of automation into the automatic configuration of network security services can provide several benefits against cyber attacks, thus balancing the increasing complexity and size of modern computer networks, such as in industrial or IoT environments. Paradigms, such as network softwarization and policy-based management, can offer dynamism and agility that are required by automated techniques to work successfully.

In light of these considerations, in this article we have surveyed the state of the art of the techniques for automating the configuration of network security services. After identifying how the orchestration of a full service should be performed in a virtualized environment, we focused on two different aspects, that are the design of the service architecture and the actual configuration of the composing functions. For each category, we analyzed the existing works by considering different features, such as the fulfillment of optimality criteria or the exploitation of formal verification. On the basis of this analysis, we also identified some main research trends for the future. In particular, the limited support for heterogeneous security services, the limited performance and scalability of the methodologies, and the limited autonomy emerged as the main drawbacks of the current approaches, which might be targets of possible future work with the aim of further improving the current state of the art.

REFERENCES

[1] Manuel Cheminod, Luca Durante, and Adriano Valenzano. 2013. Review of security issues in industrial networks. *IEEE Trans. Ind. Inf.* 9, 1 (2013), 277–293.

[2] S. Balaji, Karan Nathani, and R. Santhakumar. 2019. IoT technology, applications and challenges: A contemporary survey. *Wireless Pers. Commun.* 108, 1 (2019), 363–388.

[3] M. Uma and G. Padmavathi. 2013. A survey on various cyber attacks and their classification. *Int J. Netw. Secur.* 15 (5) (2013), 390–396.

[4] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Serena Spinoso, Fulvio Valenza, and Jalolliddin Yusupov. 2021. Improving the formal verification of reachability policies in virtualized networks. *IEEE Trans. Netw. Serv. Manage.* 18, 1 (2021), 713–728.

[5] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. 2016. Network function virtualization: State-of-the-art and research challenges. *IEEE Commun. Surv. Tuts.* 18, 1 (2016), 236–262.

[6] Sandra Scott-Hayward, Gemma O'Callaghan, and Sakir Sezer. 2013. SDN security: A survey. In *IEEE SDN for Future Networks and Services (SDN4FNS'13).* IEEE, Los Alamitos, CA.

[7] Raouf Boutaba and Issam Aib. 2007. Policy-based management: A historical perspective. *J. Netw. Syst. Manage.* 15, 4 (2007), 447–480.

[8] Juliver Gil-Herrera and Juan Felipe Botero. 2016. Resource allocation in NFV: A comprehensive survey. *IEEE Trans. Netw. Serv. Manage.* 13, 3 (2016), 518–532.

[9] Ana Carolina Riekstin, Guilherme Carvalho Januario, Bruno Bastos Rodrigues, Viviane Tavares Nascimento, Tereza Cristina Melo de Brito Carvalho, and Catalin Meirosu. 2016. A survey of policy refinement methods as a support for sustainable networks. *IEEE Commun. Surv. Tutor.* 18, 1 (2016), 222–235.

[10] Amani Abu Jabal, Maryam Davari, Elisa Bertino, Christian Makaya, Seraphin Calo, Dinesh Verma, Alessandra Russo, and Christopher Williams. 2019. Methods and tools for policy analysis. *ACM Comput. Surv.* 51, 6 (2019), 121:1–121:35.

[11] Ehab Al-Shaer, Hazem H. Hamed, Raouf Boutaba, and M. Hasan. 2005. Conflict classification and analysis of distributed firewall policies. *IEEE J. Select. Areas Commun.* 23, 10 (2005), 2069–2084.

[12] F. Valenza, C. Basile, D. Canavese, and A. Lioy. 2017. Classification and analysis of communication protection policy anomalies. *IEEE/ACM Trans. Netw.* 25, 5 (2017), 2601–2614

[13] Kresimir Popovic and Zeljko Hocenski. 2010. Cloud computing security issues and challenges. In *Proceedings of the 33rd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO'10).*

[14] Anny Martínez, Marcelo Yannuzzi, Víctor López, Diego R. López, Wilson Ramírez, René Serral-Gracià, Xavier Masip-Bruin, Maciej Maciejewski, and Jörn Altmann. 2014. Network management challenges and trends in multi-layer and multi-vendor settings for carrier-grade networks. *IEEE Commun. Surv. Tutor.* 16, 4 (2014), 2207–2230.

[15] Jacob Z. Haislip and Kalin S. Kolev. 2019. The economic cost of cybersecurity breaches: A broad-based analysis. In *Proceedings of the Workshop on the Economics of Information Security.*

[16] Liyuan Liu, Meng Han, Yan Wang, and Yiyun Zhou. 2018. Understanding data breach: A visualization aspect. In *Proceedings of the 13th International Conference on Wireless Algorithms, Systems, and Applications*, Vol. 10874.

[17] Rafal Leszczyna and Adrian Litwin. 2020. Estimating the cost of cybersecurity activities with CAsPeA: A case study and comparative analysis. In *Proceedings of the 16th International Conference on Information Systems Security*.

[18] Ken Goldberg. 2012. What is automation? *IEEE Trans. Autom. Sci. Eng.* 9, 1 (2012), 1–2.

[19] Evangelos Haleplidis, Kostas Pentikousis, Spyros G. Denazis, Jamal Hadi Salim, David Meyer, and Odysseas G. Koufopavlou. 2015. Software-defined networking (SDN): Layers and architecture terminology. RFC 7426.

[20] Nick McKeown, Thomas E. Anderson, Hari Balakrishnan, Guru M. Parulkar, Larry L. Peterson, Jennifer Rexford, Scott Shenker, and Jonathan S. Turner. 2008. OpenFlow: Enabling innovation in campus networks. *Comput. Commun. Rev.* 38, 2 (2008), 69–74.

[21] Rajendra Patil and Chirag Modi. 2019. An exhaustive survey on security concerns and solutions at different components of virtualization. *ACM Comput. Surv.* 52, 1 (2019), 12:1–12:38.

[22] René Peinl, Florian Holzschuher, and Florian Pfitzer. 2016. Docker cluster management for the cloud - survey results and own solution. *J. Grid Comput.* 14, 2 (2016), 265–282.

[23] Bob Moore, Ed Ellesson, John Strassner, and Andrea Westerinen. 2001. Policy core information model—Version 1 specification. RFC 3060.

[24] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. 2001. The ponder policy specification language. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks (POLICY'01)*.

[25] Andrzej Uszok, Jeffrey M. Bradshaw, Matt Johnson, Renia Jeffers, Austin Tate, Jeff Dalton, and J. Stuart Aitken. 2004. KAoS policy management for semantic web services. *IEEE Intell. Syst.* 19, 4 (2004), 32–41.

[26] Lalana Kagal. 2002. Rei: A Policy Language for the Me-Centric Project. HP Labs.

[27] Tan Phan, Jun Han, Jean-Guy Schneider, Tim Ebringer, and Tony Rogers. 2008. A survey of policy-based management approaches for service oriented systems. In *Proceedings of the 19th Australian Software Engineering Conference (ASWEC'08)*.

[28] Steven Davy, Brendan Jennings, and John Strassner. 2008. The policy continuum-policy authoring and conflict analysis. *Comput. Commun.* 31, 13 (2008), 2981–2995.

[29] Cataldo Basile, Fulvio Valenza, Antonio Lioy, Diego R. Lopez, and Antonio Pastor Perales. 2019. Adding support for automatic enforcement of security policies in NFV networks. *IEEE/ACM Trans. Netw.* 27, 2 (2019), 707–720.

[30] Ana Hermosilla, Alejandro Molina Zarca, Jorge Bernal Bernabé, Jordi Ortiz Murillo, and Antonio F. Skarmeta. 2020. Security orchestration and enforcement in NFV/SDN-Aware UAV deployments. *IEEE Access* 8 (2020), 131779–131795.

[31] Donald Norman. 1990. The'problem' with automation: Inappropriate feedback and interaction, not 'over-automation'. *Philos. Trans. Roy. Soc. Lond. Ser. B Biol. Sci.* 327, 1241 (1990), 585–593.

[32] Barbara Kitchenham. 2004. Procedures for performing systematic reviews. *Keele, UK, Keele Univ.* 33 (2004).

[33] Yair Bartal, Alain J. Mayer, Kobbi Nissim, and Avishai Wool. 2004. *Firmato*: A novel firewall management toolkit. *ACM Trans. Comput. Syst.* 22, 4 (2004), 381–420.

[34] Joaquín García-Alfaro, Frédéric Cuppens, Nora Cuppens-Boulahia, and Stere Preda. 2010. MIRAGE: A management tool for the analysis and deployment of network security policies. In *Proceedings of the 5th International Workshop, Data Privacy Management and Autonomous Spontaneous Security (DPM'10)*.

[35] Eder J. Scheid, Cristian Cleder Machado, Muriel Figueredo Franco, Ricardo Luis dos Santos, Ricardo J. Pfitscher, Alberto E. Schaeffer Filho, and Lisandro Zambenedetti Granville. 2017. INSpIRE: Integrated NFV-based intent refinement environment. In *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM'17)*.

[36] Mohammad Ashiqur Rahman and Ehab Al-Shaer. 2017. Automated synthesis of distributed network access controls: A formal framework with refinement. *IEEE Trans. Parallel Distrib. Syst.* 28, 2 (2017), 416–430.

[37] Joel M. Halpern and Carlos Pignataro. 2015. Service function chaining (SFC) architecture. RFC 7665.

[38] Paul Quinn and Thomas D. Nadeau. 2015. Problem statement for service function chaining. RFC 7498.

[39] Fulvio Valenza, Serena Spinoso, and Riccardo Sisto. 2019. Formally specifying and checking policies and anomalies in service function chaining. *J. Netw. Comput. Appl.* 146 (2019), 1–14.

[40] Enrique Dávalos and Benjamín Barán. 2018. A survey on algorithmic aspects of virtual optical network embedding for cloud networks. *IEEE Access* 6 (2018), 20893–20906.

[41] Feliksas Kuliesius and Vainius Dangovas. 2016. SDN enhanced campus network authentication and access control system. In *Proceedings of the IEEE International Conference on Ubiquitous and Future Networks*.

[42] Alejandro Molina Zarca, Jorge Bernal Bernabé, Ivan Farris, Yacine Khettab, Tarik Taleb, and Antonio F. Skarmeta. 2018. Enhancing IoT security through network softwarization and virtual security appliances. *Int. J. Netw. Manage.* 28, 5 (2018), 1–18.

[43] Daniele Bringhenti, Fulvio Valenza, and Cataldo Basile. 2022. Toward cybersecurity personalization in smart homes. *IEEE Secur. Priv.* 20, 1 (2022), 45–53.

[44] Zhi-Hui Zhan, Xiao Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. 2015. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.* 47, 4 (2015), 63:1–63:33.

[45] Chirag Modi, Dhiren R. Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. 2013. A survey of intrusion detection techniques in Cloud. *J. Netw. Comput. Appl.* 36, 1 (2013), 42–57.

[46] Preeti Mishra, Emmanuel S. Pilli, Vijay Varadharajan, and Udaya Kiran Tupakula. 2017. Intrusion detection techniques in cloud environment: A survey. *J. Netw. Comput. Appl.* 77 (2017), 18–47.

[47] Seungwon Shin, Phillip A. Porras, Vinod Yegneswaran, Martin W. Fong, Guofei Gu, and Mabry Tyson. 2013. FRESCO: Modular composable security services for software-defined networks. In *Proceedings of the 20th Network and Distributed System Security Symposium*

[48] Nate Foster, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. 2011. Frenetic: A network programming language. In *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming.*

[49] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. 2013. SIMPLE-fying middlebox policy enforcement using SDN. In *Proceedings of the ACM SIGCOMM Conference*

[50] Andrey Gushchin, Anwar Walid, and Ao Tang. 2015. Scalable routing in SDN-enabled networks with consolidated middleboxes. In *Proceedings of the ACM Workshop on Hot Topics in Middleboxes and Network Function Virtualization.*

[51] Nicolas Schnepf, Remi Badonnel, Abdelkader Lahmadi, and Stephan Merz. 2018. Rule-based synthesis of chains of security functions for software-defined networks. *Electr. Commun. EASST* 76 (2018).

[52] Thomas Szyrkowiec, Michele Santuari, Mohit Chamania, Domenico Siracusa, Achim Autenrieth, Victor Lopez, Joo Cho, and Wolfgang Kellerer. 2018. Automatic intent-based secure service creation through a multilayer sdn network orchestration. *J. Opt. Commun. Netw.* 10, 4 (2018), 289–297.

[53] Arthur Selle Jacobs, Ricardo José Pfitscher, Ronaldo Alves Ferreira, and Lisandro Zambenedetti Granville. 2018. Refining network intents for self-driving networks. In *Proceedings of the Workshop on Self-Driving Networks (SelfDN'18).*

[54] Nicolas Schnepf, Remi Badonnel, Abdelkader Lahmadi, and Stephan Merz. 2019. Automated factorization of security chains in software-defined networks. In *Proceedings of the IFIP/IEEE Int. Symposium on Integrated Network Management (INM'19).*

[55] Eder J. Scheid, Cristian Cleder Machado, Ricardo Luis dos Santos, Alberto E. Schaeffer Filho, and Lisandro Zambenedetti Granville. 2016. Policy-based dynamic service chaining in network functions virtualization. In *IEEE Symposium on Computers and Communication (ISCC'16).*

[56] Zheng Hao, Zhaowen Lin, and Ran Li. 2018. A SDN/NFV security protection architecture with a function composition algorithm based on trie. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering (CSAE'18).*

[57] Xin Li and Chen Qian. 2016. An NFV orchestration framework for interference-free policy enforcement. In *Proceedings of the 36th IEEE International Conference on Distributed Computing Systems (ICDCS'16).*

[58] Yi Liu, Hongqi Zhang, Jiang Liu, and Yingjie Yang. 2017. A new approach for delivering customized security everywhere: Security service chain. *Secur. Commun. Netw.* 2017 (2017), 9534754:1–9534754:17.

[59] Yicen Liu, Yu Lu, Wenxin Qiao, and Xingkai Chen. 2018. A dynamic composition mechanism of security service chaining oriented to SDN/NFV-enabled networks. *IEEE Access* 6 (2018), 53918–53929.

[60] Alireza Shameli Sendi, Yosr Jarraya, Makan Pourzandi, and Mohamed Cheriet. 2019. Efficient provisioning of security service function chaining using network security defense patterns. *IEEE Trans. Serv. Comput.* 12, 4 (2019), 534–549.

[61] Andrés F. Ocampo, Juliver Gil-Herrera, Pedro Heleno Isolani, Miguel C. Neves, Juan Felipe Botero, Steven Latré, Lisandro Zambenedetti Granville, Marinho P. Barcellos, and Luciano Paschoal Gaspary. 2017. Optimal service function chain composition in network functions virtualization. In *Proceedings of the International Conference on Autonomous Infrastructure, Management, and Security (AIMS'17).*

[62] Cataldo Basile, Antonio Lioy, Christian Pitscheider, Fulvio Valenza, and Marco Vallini. 2015. A novel approach for integrating security policy enforcement with dynamic network virtualization. In *Proceedings of the 1st IEEE Conference on Network Softwarization.*

[63] Dhanu Dwiardhika and Takuji Tachibana. 2019. Optimal construction of service function chains based on security level for improving network security. *IEEE Access* 7 (2019), 145807–145815.

[64] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, and Fulvio Valenza. 2021. A novel approach for security function graph configuration and deployment. In *Proceedings of the 7th IEEE International Conferenceon Network Softwarization.*

[65] Daniele Bringhenti, Riccardo Sisto, and Fulvio Valenza. 2023. A novel abstraction for security configuration in virtual networks. *Comput. Netw.* 228 (2023), 1–13.

[66] Woosik Lee and Namgi Kim. 2017. Security policy scheme for an efficient security architecture in software-defined networking. *Information* 8, 2 (2017), 1–16.

[67] Younghee Park, Pritesh Chandaliya, Akshaya Muralidharan, Nikash Kumar, and Hongxin Hu. 2017. Dynamic defense provision via network functions virtualization. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (SDN-NFVSec'17).*

[68] MyungKeun Yoon, Shigang Chen, and Zhan Zhang. 2010. Minimizing the maximum firewall rule set in a network with multiple firewalls. *IEEE Trans. Comput.* 59, 2 (2010), 218–230.

[69] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Fulvio Valenza, and Jalolliddin Yusupov. 2020. Automated optimal firewall orchestration and configuration in virtualized networks. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*

[70] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Fulvio Valenza, and Jalolliddin Yusupov. 2023. Automated firewall configuration in virtual networks. *IEEE Trans. Depend. Secur. Comput.* 20, 2 (2023), 1559–1576.

[71] Simone Bussa, Riccardo Sisto, and Fulvio Valenza. 2022. Security automation using traffic flow modeling. In *Proceedings of 8th IEEE International Conferenceon Network Softwarization.*

[72] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, Fulvio Valenza, and Jalolliddin Yusupov. 2019. Towards a fully automated and optimized network security functions orchestration. In *Proceedings of the International Conference on Computing, Communications and Security.*

[73] Joshua Reich, Christopher Monsanto, Nate Foster, Jennifer Rexford, and David Walker. 2013. Modular SDN programming with pyretic. *;login: USENIX Mag.* 38, 5 (2013).

[74] Frédéric Cuppens, Nora Cuppens-Boulahia, Thierry Sans, and Alexandre Miège. 2004. A formal approach to specify and deploy a network security policy. In *Proceedings of the 2nd IFIP TC1 WG1.7 Workshop on Formal Aspects in Security and Trust (FAST'04).*

[75] Joshua D. Guttman. 1997. Filtering postures: Local enforcement for global policies. In *Proceedings of the IEEE Symposium on Security and Privacy.*

[76] Angelos Keromytis, Kostas Anagnostakis, Sotiris Ioannidis, Michael Greenwald, and Jonathan Smith. 2003. Managing access control in large scale heterogeneous networks. In *Proceedings of the NATO Consultation, Command and Control Interoperable Networks for Secure Communication Symposium*

[77] Pavan Verma and Atul Prakash. 2005. FACE: A firewall analysis and configuration engine. In *Proceedings of the IEEE/IPSJ International Symposium on Applications and the Internet (SAINT'05).*

[78] John Govaerts, Arosha K. Bandara, and Kevin Curran. 2008. A formal logic approach to firewall packet filtering analysis and generation. *Artif. Intell. Rev.* 29, 3–4 (2008), 223–248.

[79] Padmalochan Bera, Soumya Kanti Ghosh, and Pallab Dasgupta. 2010. Policy based security analysis in enterprise networks: A formal approach. *IEEE Trans. Netw. Service Manage.* 7, 4 (2010), 231–243.

[80] Nicolas Stouls and Marie-Laure Potet. 2007. Security policy enforcement through refinement process. In *Proceedings of the 7th International Conference of B Users.*

[81] Arosha K. Bandara, Antonis C. Kakas, Emil C. Lupu, and Alessandra Russo. 2009. Using argumentation logic for firewall configuration management. In *Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management.*

[82] Pedro Adão, Claudio Bozzato, G. Dei Rossi, Riccardo Focardi, and Flaminia L. Luccio. 2014. Mignis: A semantic based tool for firewall configuration. In *Proceedings of the IEEE 27th Computer Security Foundations Symposium*

[83] Dinesha Ranathunga, Matthew Roughan, Phil Kernick, and Nick Falkner. 2016. The mathematical foundations for mapping policies to network devices. In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications.*

[84] Ahmed El-Hassany, Petar Tsankov, Laurent Vanbever, and Martin T. Vechev. 2017. Network-wide configuration synthesis. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV'17).*

[85] Ahmed El-Hassany, Petar Tsankov, Laurent Vanbever, and Martin T. Vechev. 2018. NetComplete: Practical network-wide configuration synthesis with autocompletion. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI'18).*

[86] Manuel Cheminod, Luca Durante, Lucia Seno, Fulvio Valenza, and Adriano Valenzano. 2019. A comprehensive approach to the automatic refinement and verification of access control policies. *Comput. Secur.* 80 (2019), 186–199.

[87] Abhijeet Sahu, Patrick Wlazlo, Nastassja Gaudet, Ana Goulart, Edmond Rogers, and Katherine Davis. 2022. Generation of firewall configurations for a large scale synthetic power system. In *Proceedings of the IEEE Texas Power and Energy Conference.*

[88] Lorenzo Ceragioli, Pierpaolo Degano, and Letterio Galletta. 2022. Can my firewall system enforce this policy? *Comput. Secur.* 117 (2022), 1–20.

[89] Arthur Selle Jacobs, Ricardo J. Pfitscher, Rafael Hengen Ribeiro, Ronaldo A. Ferreira, Lisandro Zambenedetti Granville, Walter Willinger, and Sanjay G. Rao. 2021. Hey, Lumi! using natural language for intent-based network management. In *Proceedings of the USENIX Annual Technical Conference.*

[90] Manel Smine, David Espes, Nora Cuppens-Boulahia, Frédéric Cuppens, and Marc-Oliver Pahl. 2021. A priority-based domain type enforcement for exception management. In *Proceedings of the International Symposium on Foundations and Practice of Security.*

[91] Erisa Karafili, Fulvio Valenza, Yichen Chen, and Emil C. Lupu. 2020. Towards a framework for automatic firewalls configuration via argumentation reasoning. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium.*

[92] Dinesha Ranathunga, Matthew Roughan, and Hung X. Nguyen. 2022. Verifiable policy-defined networking using metagraphs. *IEEE Trans. Depend. Secur. Comput.* 19, 1 (2022), 482–494.

[93] Mark J. McArdle, Brent A. Johnston, Philip D. R. Nathan, and James Dool. Automatically configuring a computer firewall based on a network connections. U.S. Patent 7 284 267, Mar. 2001.

[94] J. Burns, A. Cheng, P. Gurung, S. Rajagopalan, P. Rao, D. Rosenbluth, A. V. Surendran, and D. M. Martin. 2001. Automatic management of network security policy. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, Vol. 2.

[95] Mohamed G. Gouda and Alex X. Liu. 2004. Firewall design: Consistency, completeness, and compactness. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04).*

[96] Frédéric Cuppens, Nora Cuppens-Boulahia, and Joaquin Garcia-Alfaro. 2006. Detection of network security component misconfiguration by rewriting and correlation. *Conference on Security in Network Architectures and Security of Information Systems.*

[97] Sruthi Bandhakavi, Sandeep N. Bhatt, Cat Okita, and Prasad Rao. 2009. Analyzing end-to-end network reachability. In *Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management.*

[98] Sanjai Narain, Rajesh Talpade, and Gary Levin. 2010. *Network Configuration Validation.* Guide to Reliable Internet Services and Applications, part of the Computer Communications and Networks book series, Springer, 2010.

[99] Nihel Ben Youssef and Adel Bouhoula. 2011. A fully automatic approach for fixing firewall misconfigurations. In *Proceedings of the 11th IEEE International Conference on Computer and Information Technology (CIT'11).*

[100] Fei Chen, Alex X. Liu, JeeHyun Hwang, and Tao Xie. 2012. First step towards automatic correction of firewall policy faults. *ACM Trans. Auton. Adapt. Syst.* 7, 2 (2012), 27:1–27:24.

[101] Aaron Gember-Jacobson, Aditya Akella, Ratul Mahajan, and Hongqiang Harry Liu. 2017. Automatically repairing network control planes using an abstract representation. In *Proceedings of the 26th Symposium on Operating Systems Principles.*

[102] Kamel Adi, Lamia Hamza, and Liviu Pene. 2018. Automatic security policy enforcement in computer systems. *Comput. Secur.* 73 (2018), 156–171.

[103] Seungwon Shin and Guofei Gu. 2012. CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In *Proceedings of the IEEE International Conference on Network Protocols.*

[104] Md. Mazharul Islam, Qi Duan, and Ehab Al-Shaer. 2019. Specification-driven Moving Target Defense Synthesis. In *Proceedings of the 6th ACM Workshop on Moving Target Defense (MTD@CCS'19).*

[105] Andreas Voellmy, Hyojoon Kim, and Nick Feamster. 2012. Procera: A language for high-level reactive network control. In *Proceedings of the 1st workshop on Hot topics in software defined networks (HotSDN'12).*

[106] Adrian Lara and Byrav Ramamurthy. 2016. OpenSec: Policy-based security using software-defined networking. *IEEE Trans. Netw. Serv. Manage.* 13, 1 (2016), 30–42.

[107] Vijay Varadharajan, Kallol Krishna Karmakar, and Udaya Kiran Tupakula. 2017. Securing communication in multiple Autonomous System domains with Software Defined Networking. In *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management.*

[108] Arne Ludwig, Szymon Dudycz, Matthias Rost, and Stefan Schmid. 2018. Transiently policy-compliant network updates. *IEEE/ACM Trans. Netw.* 26, 6 (2018), 2569–2582.

[109] V. Varadharajan and U. Tupakula. 2019. Counteracting attacks from malicious end hosts in software defined networks. *IEEE Trans. Netw. Service Manage.* (2019), 160–174.

[110] Mudassar Hussain, Nadir Shah, and Ali TahirÂă. 2019. Graph-based policy change detection and implementation in sdn. *Electronics* 8, 10 (2019), 1–21.

[111] Zhi Fu and Shyhtsun Felix Wu. 2001. Automatic generation of IPSec/VPN security policies in an intra-domain environment. In *Proceedings of the 12th International Workshop on Distributed Systems (DSOM'01).*

[112] Yanyan Yang, Charles U. Martel, and Shyhtsun Felix Wu. 2004. On building the minimum number of tunnels: An ordered-split approach to manage IPSec/VPN policies. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*

[113] Yanyan Yang, Zhi (Judy) Fu, and Shyhtsun Felix Wu. 2003. BANDS: An inter-domain internet security policy management system for IPSec/VPN. In *Proceedings of the IFIP/IEEE 8th International Symposium on Integrated Network Management (IM'03).*

[114] Chi-Lan Chang, Yun-Peng Chiu, and Chin-Laung Lei. 2005. Automatic generation of conflict-free IPsec policies. In *Proceedings of the 25th IFIP WG 6.1 International Conference Formal Techniques for Networked and Distributed Systems (FORTE'05).*

[115] Mohammad Mehdi Gilanian Sadeghi, Borhanuddin Mohd Ali, Hossein Pedram, Mehdi Dehghan, and Masoud Sabaei. 2008. A new method for creating efficient security policies in virtual private network. In *Proceedings of the 4th International Conference Collaborative Computing: Networking, Applications and Worksharing, (CollaborateCom'08).*

[116] Michael Rossberg, Guenter Schaefer, and Thorsten Strufe. 2010. Distributed automatic configuration of complex ipsec-infrastructures. *J. Netw. Syst. Manage.* 18, 3 (2010), 300–326.

[117] Salman Niksefat and Masoud Sabaei. 2010. Efficient algorithms for dynamic detection and resolution of IPSec/VPN security policy conflicts. In *Proceedings of the 24th IEEE Conference on Advanced Information Networking and Applications.*

[118] Lotfi Firdaouss, Ayoub Bahnasse, Belkadi Manal, and Yazidi Ikrame. 2021. Automated VPN configuration using DevOps. In *Proceedings of the International Conference on Emerging Ubiquitous Systems and Pervasive Networks.*

[119] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, and Fulvio Valenza. 2020. Short paper: Automatic configuration for an optimal channel protection in virtualized networks. In *Proceedings of the Workshop on Cyber-Security Arms Race.*

[120] Ricardo Neisse, Gary Steri, and Gianmarco Baldini. 2014. Enforcement of security policy rules for the Internet of Things. In *Proceedings of the IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications.*

[121] Sabrina Sicari, Alessandra Rizzardi, Daniele Miorandi, Cinzia Cappiello, and Alberto Coen-Porisini. 2016. Security policy enforcement for networked smart objects. *Comput. Net.* 108 (2016), 133–147.

[122] S. Sicari, A. Rizzardi, L.A. Grieco, G. Piro, and A. Coen-Porisini. 2017. A policy enforcement framework for Internet of Things applications in the smart health. *Smart Health* 3-4 (2017), 39–74.

[123] Sérgio Figueiredo, Paulo Silva, Alfonso Iacovazzi, Vitalina Holubenko, João Casal, Jose M. Alcaraz Calero, Qi Wang, Pedro Colarejo, Ross Little Armitt, Giacomo Inches, and Shahid Raza. 2022. ARCADIAN-IoT - Enabling autonomous trust, security and privacy management for IoT. In *Proceedings of the 5th Global IoT Summit. Lecture Notes in Computer Science*, Vol. 13533. Springer.

[124] Sabrina Sicari, Alessandra Rizzardi, Daniele Miorandi, and Alberto Coen-Porisini. 2017. Security towards the edge: Sticky policy enforcement for networked smart objects. *Inf. Syst.* 71 (2017), 78–89.

[125] Gokhan Sagirlar, Barbara Carminati, and Elena Ferrari. 2018. Decentralizing privacy enforcement for Internet of Things smart objects. *Comput. Net.* 143 (2018), 112–125.

[126] Alessandra Rizzardi, Sabrina Sicari, Daniele Miorandi, and Alberto Coen-Porisini. 2022. Securing the access control policies to the Internet of Things resources through permissioned blockchain. *Concurr. Comput. Pract. Exp.* 34 (2022), 1–19.

[127] Vasudevan Nagendra, Arani Bhattacharya, Vinod Yegneswaran, Amir Rahmati, and Samir Ranjan Das. 2020. An intent-based automation framework for securing dynamic consumer IoT infrastructures. In *Proceedings of the Web Conference.*

[128] Mohammad Ashiqur Rahman, Amarjit Datta, and Ehab Al-Shaer. 2021. Automated configuration synthesis for resilient smart metering infrastructure. *EAI Endors. Trans. Secur. Saf.* 8, 28 (2021), 1–14.

[129] Daniele Bringhenti, Jalolliddin Yusupov, Alejandro Molina Zarca, Fulvio Valenza, Riccardo Sisto, Jorge Bernal Bernabé, and Antonio F. Skarmeta. 2022. Automatic, verifiable and optimized policy-based security enforcement for SDN-aware IoT networks. *Comput. Net.* 213 (2022), 1–12.

[130] Sian En Ooi, Razvan Beuran, Yasuo Tan, Takayuki Kuroda, Takuya Kuwahara, and Norihito Fujita. 2022. SecureWeaver: Intent-driven secure system designer. In *Proceedings of the ACM Workshop on Secure and Trustworthy Cyber-Physical Systems.*

[131] Ooi Sian En, Razvan Beuran, Takayuki Kuroda, Takuya Kuwahara, Ryosuke Hotchi, Norihito Fujita, and Yasuo Tan. 2023. Intent-driven secure system design: Methodology and implementation. *Comput. Secur.* 124 (2023), 1–20.

[132] Sébastien Ziegler, Antonio F. Skarmeta, Jorge Bernal Bernabé, Eunsook Eunah Kim, and Stefano Bianchi. 2017. ANASTACIA: Advanced networked agents for security and trust assessment in CPS IoT architectures. In *Proceedings of the IEEE Global Internet of Things Summit.*

[133] Alejandro Molina Zarca, Jorge Bernal Bernabé, Antonio F. Skarmeta, and Jose M. Alcaraz Calero. 2020. Virtual IoT HoneyNets to mitigate cyberattacks in SDN/NFV-enabled IoT networks. *IEEE J. Select. Areas Commun.* 38, 6 (2020), 1262–1277.

[134] Alejandro Molina Zarca, Miloud Bagaa, Jorge Bernal Bernabé, Tarik Taleb, and Antonio F. Skarmeta. 2020. Semantic-aware security orchestration in SDN/NFV-enabled IoT systems. *Sensors* 20, 13 (2020), 1–24.

[135] Gaurav Goyal, Peng Liu, and Shamik Sural. 2022. Securing smart home IoT systems with attribute-based access control. In *Proceedings of the ACM Workshop on Secure and Trustworthy Cyber-Physical Systems.*

[136] Joshua D. Guttman and Amy L. Herzog. 2005. Rigorous automated network security management. *Int. J. Inf. Secur.* 4 (1) (2005), 29–48.

[137] Tomás E. Uribe and Steven Cheung. 2007. Automatic analysis of firewall and network intrusion detection system configurations. *J. Comp. Sec.* 15, 6 (2007), 691–715.

[138] Cataldo Basile, Daniele Canavese, Leonardo Regano, Ignazio Pedone, and Antonio Lioy. 2022. A model of capabilities of Network Security Functions. In *Proceedings of 8th IEEE International Conference on Network Softwarization.*

[139] Patrick Lingga, Jeonghyeon Kim, Jorge David Iranzo Bartolomé, and Jaehoon Jeong. 2021. Automatic data model mapper for security policy translation in interface to network security functions framework. In *Proceedings of the IEEE International Conference on Information and Communication Technology Convergence.*

[140] Alain J. Mayer, Avishai Wool, and Elisha Ziskind. 2000. Fang: A firewall analysis engine. In *Proceedings of the Symposium on Security and Privacy.*

[141] Angelos D. Keromytis, Sotiris Ioannidis, Michael B. Greenwald, and Jonathan M. Smith. 2003. The STRONGMAN architecture. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX-III'03).*

[142] Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. 2007. Ethane: Taking control of the enterprise. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.*

[143] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba. 2013. PolicyCop: An autonomic qos policy enforcement framework for software defined networks. In *Proceedings of the IEEE SDN for Future Networks and Services (SDN4FNS'13).*

[144] Celio Trois, Marcos Didonet Del Fabro, Luis Carlos Erpen De Bona, and Magnos Martinello. 2016. A survey on SDN programming languages: Toward a taxonomy. *IEEE Commun. Surv. Tutor.* 18, 4 (2016), 2687–2712.

[145] Hamed HaddadPajouh, Ali Dehghantanha, Reza M. Parizi, Mohammed Aledhari, and Hadis Karimipour. 2021. A survey on internet of things security: Requirements, challenges, and solutions. *Internet Things* 14 (2021), 1–19.

[146] Matteo Dell'Amico, Gabriel Serme, Muhammad Sabir Idrees, Anderson Santana de Oliveira, and Yves Roudier. 2013. HiPoLDS: A hierarchical security policy language for distributed systems. *Inf. Secur. Tech. Rep.* 17, 3 (2013), 81–92.

[147] Alessandra Rizzardi, Daniele Miorandi, Sabrina Sicari, Cinzia Cappiello, and Alberto Coen-Porisini. 2015. Networked smart objects: Moving data processing closer to the source. In *Proceedings of the International Summit on IoT Infrastructures.*

[148] Wenjun Xiong, Emeline Legrand, Oscar Åberg, and Robert Lagerström. 2022. Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix. *Softw. Syst. Model.* 21 (2022), 157–177.

[149] Günter Karjoth, Matthias Schunter, and Michael Waidner. 2002. Privacy-enabled services for enterprises. In *Proceedings of the 13th IEEE International Work. on Database and Expert Systems Applications.*

Received 14 January 2021; revised 23 June 2023; accepted 6 August 2023