INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt XXXX; **00**:1–19
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nem

Inter-function Anomaly Analysis for correct SDN/NFV deployment

Cataldo Basile, Daniele Canavese, Antonio Lioy, Christian Pitscheider and Fulvio Valenza

Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy
(e-mail: {cataldo.basile, daniele.canavese, antonio.lioy, christian.pitscheider, fulvio.valenza}@polito.it)

SUMMARY

Implementing the security of a network consists in individually configuring several network functions. Network functions are configured by means of a policy composed of a set of rules but their actual behaviour is influenced by the other policies implemented by all the other network functions around them. This paper proposes a formal model that can be used to detect inter-function anomalies, which are defined as the interferences between two or more functions deployed in the same network. We have proved with experiments that the proposed model is fast and scalable.

Received . . .

KEY WORDS: NFV/SDN, inter-function anomaly analysis, VNF policy

1. INTRODUCTION

The increasing complexity and heterogeneity of modern distributed systems is mining their own security. The impact of the 'human error factor' on continuously growing networks is certainly not negligible, as several studies have proved. Nearly 60% of the security breaches which have occurred in 2015 are attributable to errors made by system and network administrators [1].

This situation is exacerbated by the recent growth in virtualization based technologies which increase the complexity of networks, especially software-defined networking (SDN) and network functions virtualization (NFV). NFV decouples software implementation of middleboxes (e.g., router, firewall, NAT) from the compute, storage and networking resources through a virtualization layer. A middlebox is defined as any intermediary device that performs network or security functions as well as the normal IP router operations on the datagram path between a source host and destination host [2]. In this context, the ETSI standards organization is working on the definition of an architecture and the requirements for the deployment of any virtual network functions (VNFs).

Currently the majority of the network functions that enforce security policies are configured by making use of low level function-specific parameters, which are set in a manual fashion. Even though some approaches have been proposed for configuring middleboxes through a unified interface in a VNF architecture [3], these works need anyway the function-specific parameters from users and overlook the special requirements in configuring security functions. As it is already envisioned by the I2NSF [4] working group in ETSI, the configuration of single VNFs is not enough to enforce security in the whole network, because administrators have to consider also the interactions between different security controls in distributed environment for making the network secure.

However, unless a system is really simple, an administrator cannot actually evaluate the global effect of the enforced security policy, which is obtained by the configuration of all the functions deployed in the network. In other words, this important task is performed without a holistic view of the overall security requirements, and this increases the chance of misconfigurations. In addition, the security administrators must deal with the highly dynamic nature of these deployments, hence worsening the problem even further. Indeed, VNFs can run on a range of industry standard server hardware, and can be moved and instantiated at any locations in the network, without the need of new equipment installation [5].

The typical approach is trial-and-error. When one or more misconfigurations are reported, the administrators correct them by creating ad-hoc rules and repeat the process until (hopefully) no more errors are present. This methodology, although simple, is only a temporary palliative since it can produce serious maintenance problems in the future. Guaranteeing the absence of misconfigurations is however nearly impossible without an appropriate software tool. It is therefore highly desirable to have a practical solution to evaluate the policy actually enforced which is based on sound theoretical foundations.

In the last few years, several authors have tried to identify potential misconfigurations by detecting and resolving policy conflicts. These works have classified and detected conflicts in the same device (intra-policy) or conflicts between homogeneous devices, e.g., two firewalls or two cascading IPsec devices (inter-policy) [6, 7]. Nevertheless, the complexity of real systems is not self-contained, as each network function may affect the behaviour of other functions in the same network. For instance, a firewall may block some encrypted communication channels or a network address translation may alter the decision of several packet filters. For this reason, it is indispensable to help the administrators by supporting, in a general analysis framework, different types of functions (e.g., firewalls, content filters, channel protection devices, logging, monitoring, and so on) and their interactions.

In this paper we propose a novel approach that is able to analyse a SDN/NFV scenario when heterogeneous networking devices and technologies are used. Our approach also works if different types of policy-enabled VNFs are deployed. Our solution is both easy to extend to other function types and fast to compute, as our performance tests have proved. The main contribution of this paper is a formal model that allows the identification of inter-function anomalies, which are defined as the *interferences* that originate when two or more functions are deployed and configured in the same network. These anomalies need to be carefully analysed by administrators, as they usually hide misconfigurations. Inter-function anomaly types have been classified and described. These anomalies are characterized by the correlations among rules belonging to the policies of different network functions types. We have focused our attention on the modifications that one network function applies to the packets before they are processed by another network function. For instance, these modifications include changes in the packet header, encryption of the payload, and tunneling (i.e., new header addition). Indeed, the most difficult part of the algorithm has been to model the transformations without simulating the changes. To the best of our knowledge, this is the first model that allows the anomaly analysis of non-homogeneous functions and, in particular, of those functions that operate modifications on the packets.

The paper is structured as follows. Section 2 presents the related work. Section 3 starts to introduce our approach by looking at a simple reference example. Section 4 describes how our approach can be applied in the NFV architecture, while Sections 5, 6 and 7 present the formal definition of policy-enabled devices, our analysis model and the model for the transformation resolution, respectively. Finally, Section 8 contains the performance evaluation of our tool and Section 9 presents the conclusions.

2. RELATED WORK

NFV and SDN paradigms allow multiple applications and users to program the same physical network simultaneously, at the cost of potentially creating conflicting rules that alter the intended behaviour of one or more applications. In recent years, a significant amount of investigations have

DOI: 10.1002/nem

addressed conflict analysis and configuration modelling. Some previous research has focused on conflict analysis for specific middleboxes, while, on the other hand, a new branch of research has focused on OpenFlow misconfiguration and analysis of network invariants. However, there are not many works that analyse the problem of inter-function anomalies.

2.1. Intra-function conflict analysis

The most prominent work on firewall conflict analysis has been performed by Al-Shaer et al., Yuan et al., Liu et al., and Cuppens et al. The anomaly analysis of filtering policies in distributed scenarios was first introduced by Al-Shaer and Hamed [6]. The authors presented a classification scheme for packet filter rule relations, based on which they defined four types of intra-policy rule anomalies (shadowing, correlation, generalisation and redundancy) and five intra-policy anomalies (shadowing, spuriousness, redundancy, correlation and irrelevance). Liu et al. focused only on detecting and removing redundant rules [8]. The authors categorized the redundant rules into upward redundant rules, which are never matched, and downward redundant rules, which are matched but enforce the same action as some other rules with a lower priority. Cuppens at al. proposed a system of anomaly resolution [9]. Although they only support shadowing and redundancy anomalies of single policies, the resulting policy is anomaly free. FIREMAN, by Yuan et al., was one of the first approaches to help network administrators to correctly configure a network [10]. Their model makes use of binary decision diagrams (BDDs) to represent packet filtering policies and it also verifies that an end-to-end policy has been correctly deployed. Basile and Lioy proposed a model to perform anomaly analysis of application layer policies and applied it to check HTTP policies [11].

There is also a considerable amount of work on detecting anomalies in routing configurations. The first systematic study of BGP configuration errors that propagate across the backbone of the Internet was presented by Mahajan et al. [12]. Feamster and Balakrishnan described the design and the implementation of a router configuration checker, a tool that finds faults in BGP configurations using static analysis [13]. Finally, Alimi et. al. presented and evaluated the novel idea of shadow configurations [14]. A shadow configuration allows a configuration evaluation before its deployment and can thus reduce potential network disruptions.

Research on IPSec and VPN anomalies was proposed by Fu et al., Hamed et al. and Basile et al. Fu et al. [15] presented a first approach for IPsec policy anomalies detection. The analysis is performed on a set of policy implementations written in a high-level language and the anomalies are identified by verifying the implemented policies against the desired ones. Fu et al. defined an anomaly as ‘a situation where the policy implementation does not satisfy the requirements of the desired policy’. Hamed and Al-Shaer [7] formalized the classification scheme of Fu. Their model not only incorporated the encryption capabilities of IPsec, but also its packet filter capabilities (this feature can be seen as the extension of its packet filter classification). In particular, the authors identified two new IPsec errors (overlapping-session and multi-transform anomalies), both valid for inter and intra-policy analysis.

Finally, Basile et al. [16, 17] presented a novel classification of communication protection policy anomalies. The proposed model allows the detection of a number of anomalies arising from the interactions between various protection protocols that work at different layers of the ISO/OSI stack (IPSec, TLS, SSH and WS-Security), security properties and communication scenarios such as end-to-end channels, VPN and remote-access communications.

2.2. OpenFlow misconfigurations

OpenFlow was first introduced by McKeown et al. to standardize the communication between the switches and the software-based controller in an SDN architecture [18]. Although the goal of OpenFlow was to provide a platform that would allow researchers to run experiments in production networks, the industry has also embraced SDN and OpenFlow as a strategy to increase the functionality of the network while reducing costs and hardware complexity.

In the last years, several works have been proposed to find anomaly misconfigurations in OpenFlow, in order to allow multiple applications to run on the same physical network in a non-conflicting manner. Al-Shaer and Al-Haj described a tool, FlowChecker, to identify any

DOI: 10.1002/nem

intra-switch misconfiguration within a single FlowTable. They also described the inter-switch or inter-federated inconsistencies in a path of OpenFlow switches across the same or different OpenFlow infrastructures [19]. NICE, proposed by Canini et al., performs a symbolic execution of OpenFlow applications and applies a model checking evaluation to explore the state space of an entire OpenFlow network [20]. In addition, the authors devised a number of new, domain-specific techniques for mitigating the state space explosion that plagues approaches such as ours. Finally, Batista et al. presented an approach for the conflict detection using several first-order logic rules to define possible antagonisms and employ an inference engine to detect conflicting flows before the OpenFlow controller implement in the network elements [21].

2.3. Network invariant analysis

In few years the verification of the network invariant violations (i.e. presence of forwarding loops, black holes, etc.) has become a hot topic in the SDN and VNF context. Today the SDN controllers are capable of handling around 30K new flow installs per second while maintaining a sub-10 ms flow install time [22], then they check for updates and analyse the invariants before they hit the network, raise an alarm, if needed, even preventing bugs as they occur by blocking problematic changes.

VeriFlow makes use of graph search techniques to verify network-wide invariants and handles dynamic changes in real time. The VeriFlow tool finds faulty rules issued by SDN applications and optionally prevent them from reaching the network and causing an anomalous network behavior [23]. ConfigChecker, proposed by Al-shaer et al., converts the network rules (configuration and forwarding rules respectively) into boolean expressions that are checked for network invariants [24, 25]. Other works, instead, are oriented in verifying isolation properties in networks that include some 'dynamic data path' elements using a model checking approach [26, 27]: the notion of verification is extended to networks containing dynamic data paths, where a checking of invariants such as the connectivity or isolation is performed.

3. EXAMPLE

In this section we present an example that will be used to informally introduce the concepts of our model. We use as a reference the simplified network scenario depicted in Figure 1.

The figure shows a simple corporate scenario, where different users connect via remote access to the corporate data center. Some services are used by all the corporate users and they are located in the Global subnet, which includes the mail server and an anti-spam system. Other services are department-specific and they are located in the $Department_1$-*3* subnets. $Department_1$ offers an internal repository and a database, the $Department_2$ provides an application service, while $Department_3$ has three file services managed by a file access control.

In this example a user is authorized to access the corporate mail server and the services available for the department he belongs to. To implement this policy, user traffic has to pass through a set of security controls before accessing the different services. Indeed, at the border there is a VPN gateway, a traffic monitor, a firewall and a router. These VNFs process the ingress traffic exactly in this order, dropping or altering the traffic by forwarding (i.e, the firewall) and modifying the headers and/or payloads of the packets received (i.e. the VPN gateway). Moreover, they can also log traffic info (i.e. the monitor).

Moreover, there are other high-level requirements to implement: (*i*) drop all the encrypted traffic in output to prevent any information disclosure; (*ii*) permit access only to authenticated users; (*iii*) use a load balancer to access the $department_2$ applications; (*iv*) supervise the access to the $department_3$ file servers (through an IDS).

Although the proposed scenario is small and simple, we can find some inter-function anomalies. If we assume the following authorized activities:

- $User_1$ wants to connect via SSH to the $Department_1$ DB;
- $User_2$ wants to access the $Department_2$ application service;

DOI: 10.1002/nem

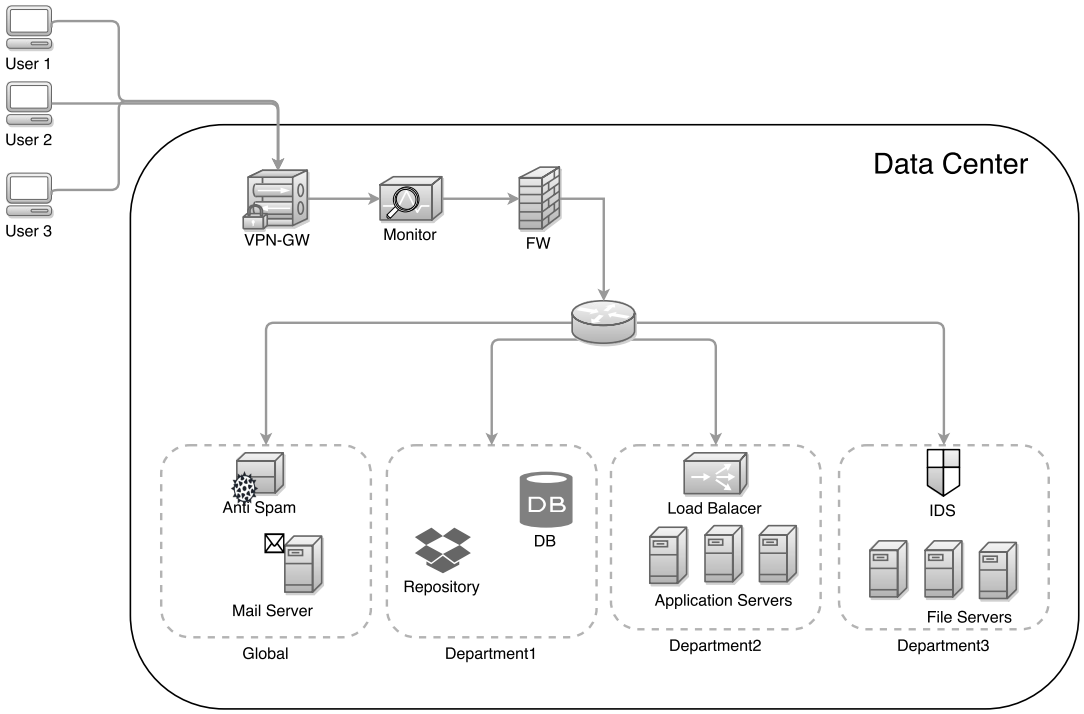


Figure 1. Example network.

- User_3 wants to download a file from the Department_3file server.

an administrator may notice four incorrect implementation of the security requirements:

1. the monitor cannot analyse the traffic towards the DB because it is encrypted;
2. User_1 does not receive responses from the DB because the firewall drops the traffic;
3. the monitor cannot distinguish the three different application servers because the load balancer *changes* the source IP addresses;
4. the IDS configuration depends directly on the firewall configuration.

These inconsistencies could have been detected by means of our model, which is able to identify the anomalies by analysing the policies of the VNFs in the network. In particular, we focus on the modification induced by the VNF into the packets.

The model we propose requires as input the network topology with the explicit indication of the VNF. For each VNF it is necessary to specify its capabilities (in terms of the actions it can apply to the packets) and the configuration rules., i.e., its policy. The model outputs the anomalies as a set of rule pairs that need to be inspected by an administrator. The anomalous rules may belong to different VNFs.

To use our model and discover anomalies, it is not necessary to specify the traffic patterns. The model simply identifies all the potential issues by analysing the conditions of the VNF configuration rules, which explicitly determine the packets and flows that will be matched by the different VNFs.

In the next sections we will show how our model enables the discovery of these anomalies. Informally, for anomalies (1) and (3), the monitor, a VNF that only reads information in the packet, has a rule to log packets that are received after another VNF has modified them. In the first case, the modification is performed by the VPN gateway, in the second one, by the load balancer. For anomaly (2), it is the rule to drop all the encrypted traffic that prevents replies from reaching User_2. Our model notices that there is a filtering rule dropping packets that are transformed (i.e., encrypted). Analogously, for anomaly (4), it discovers that the IDS has to look into traffic that is filtered by another VNF.

DOI: 10.1002/nem

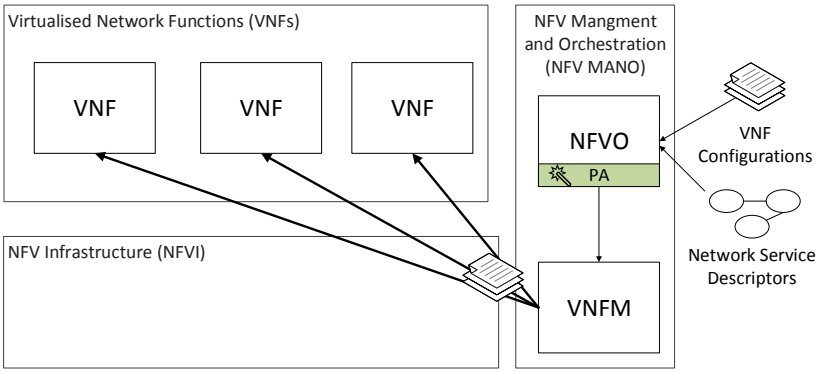


Figure 2. Our proposed extension of the ETSI NFV Architectural Framework.

4. APPROACH

The most important standard in NFV infrastructures is the *ETSI NFV Architectural Framework* [5] whose basic structure is sketched in Figure 2. Its main components are the VNFs themselves, the NFVI (NFV Infrastructure) and the NFV MANO (NFV Management and Orchestration).

In short, the NFVI provides the virtual resources required to support the execution of the VNFs, while the NFV MANO covers the virtualisation-specific management tasks necessary in the NFV framework. In particular, the NFV MANO is compose by:

- a set of VNFMs (VNF Managers), responsible for the VNFs' life-cycles management by making use of the operations such as instantiation, scaling (increase or reduce the capacity of the VNF), termination and so on;
- the NFVO (NFV Orchestrator), whose task is the management of the network service life-cycle. The NFVO performs its services by using the VNFM services and by orchestrating the NFVI that supports the interconnection between VNFs functionality.

In order to support the configuration and policy management of the VNFs, the NFVO collects a set of external inputs, it processes the acquired information and sends the configurations to each VNF via the NFV managers. The NFVO inputs include both the NSDs (Network Service Descriptors) and the current VNF configurations. The NSDs delineate the instantiation of a network service and they include several data such as the network service features, its deployment flavours and the service topology. The latter describes the relationships with the other VNFs, the virtual links, ... These dependencies are defined in terms of source and target VNFs (i.e. target VNF "depends on" source VNF). In other words, a source VNF must exist and connect to the service before the target VNF can be initiated/deployed.

In this context, the NFVO is a reactive but liberal component since it 'trusts' that the various VNFs can communicate and interacts with each other flawlessly. *Our approach aims to increase the controlling capabilities of the orchestrator* in order to make it aware of the anomalies that can arise between the VNFs.

In our approach, we added a new component to the NFVO, named the *Policy Analyser* (*PA*), whose ultimate goal is the VNFs anomaly analysis. Note that this component has no enforcement capabilities. Our goal is solely the anomaly detection. The results of the analysis can be used to produce a report for the administrators or to actively reconfigure the system on-the-fly by some other building block. Figure 2 sketches an architectural overview of an NFV with the Policy Analyser and its location inside the orchestrator.

In Section 3, we described a set of anomalies among several function that can hinder the offered services and/or decrease their performances, thus making the network operating far from the expected capacity (the VNFs may still work, but with seriously degraded performances). In order to

DOI: 10.1002/nem

identify such inter-function anomalies, the Policy Analyser must receive enough information from the NFVO. This data include the VNF configurations and the network service descriptors, so that it can extract:

- the single rules and their priority from each VNF configuration;
- the VNF forwarding graph and the dependencies among the functions from the network service descriptors.

From a mathematical point-of-view the Policy Analyser is built on a formal algebraic model that can detect the presence and absence of interferences (anomalies) between two or more functions deployed in the same network service. The advantage of our approach is that this technique can be used both *on-line*, by integrating the PA in an existing system, to perform an on-the-fly check of a network service, or *off-line*, by using the PA alone to perform a-priori checks before a real-world deployment can take place.

Although the Policy Analyser can be integrated in an existing NFV Architectural Framework, we consider describing this task out of scope for our work, hence we will focus our efforts on describing its internal model (Sections 5, 6 and 7) and its off-line performance tests (Section 8).

5. THE MODEL

This section presents a formal model, that extends the work proposed in [28] to allow the analysis of different types of network functions. A policy-enabled network function takes a packet in input, it applies some transformations to the received data and it outputs the transformed packet. Since a packet is a sequences of bits, we assume that the packets belong to a finite discrete set $\mathbb{P}$, the *packet space*, without loss of generality. The VNF is then defined as a function $V : \mathbb{P} \to \mathbb{P}$, such that $\mathbb{P} \ni x \mapsto V(x)$. Usually, a policy-enabled VNF is configured by means of a "set of rules " and a "a rule is an association of conditions to a set of actions" [29].

Rules are the method used for describing the transformations to apply to a packet. The policy-enabled functionality of these network functions can be modelled by means of two processes: (1) a process that selects the rules matching a packet, and (2) a process that decides the transformation to apply to the packet. The VNF must also trigger a process that actually applies the transformation to the packet, but it is belong of this work to describe. To fully model a policy-enabled VNF, it is necessary to formally define the packets, the rules (actions and conditions), the matching process and the resolution process.

5.1. Packets

We introduce a structure in the packet space to model the conditions. We assume an organization of the packet bitstream in a *set of fields*. Fields are, for instance, the source IP address in an IP packet, the port number in UDP, the ACK in TCP as well as the MIME type of an HTTP reply. We suppose that all the fields are known and organized in a set $\mathbb{F} = \{\mathbb{F}_1, \ldots, \mathbb{F}_m\}$. In the rest of the paper we will use the notation $\mathbb{F}_I = \{\mathbb{F}_i \mid i \in I \subseteq [1, m]\}$ to indicate a subset of $\mathbb{F}$ indexed by some sets $I \subseteq [1, m]$. As the fields are a disjoint portion of a packet it is simple to prove that:

$$\mathbb{F}_I = \prod_{i \in I} \mathbb{F}_i$$

Every field is characterised by its length, that is, the number of bits that form it. Every $\mathbb{F}_i$ is composed by l_i bits, thus $\mathbb{F}_i$ is isomorphic to $\left[0, 2^{l_i} - 1\right] \subset \mathbb{N}$. Under this hypothesis, a packet becomes a set of values, a value for a specific field, that is, $x = \{f_i \in \mathbb{F}_i,\ \mathbb{F}_i \in \mathbb{F}_I,\ I \subseteq [1, m]\}$, that is $x \in \mathbb{F}_I$, for some set I. The packet space $\mathbb{P}$ becomes the set of all the possible field assignments. It is worth noting that not every packet may contains a value for each field of $\mathbb{F}$ (as in real cases).

DOI: 10.1002/nem

5.2. Conditions

A condition $c \subseteq \mathcal{C}$ in the decision space $\mathcal{C}$ is a set of values s_j for distinct fields in a packet.

$$c = \prod_{j \in I} s_j \qquad s_j \subseteq \mathbb{F}_j \qquad I \subseteq [1, m]$$

The decision space $\mathcal{C}$ is a subset of packet fields used to define the condition.

$$\mathcal{C} = \prod_{j \in I} \mathbb{F}_j \qquad \mathbb{F}_j \subseteq \mathbb{F}_I \qquad I \subseteq [1, m]$$

A packet $x \in \mathbb{F}_I$ satisfies a condition c only if all values s_i are satisfied. A value s_i is satisfied in $\mathbb{F}_i$ if and only if (1) the packet contains a value f_i in the field $\mathbb{F}_i$ and (2) f_i is in s_i (i.e., $i \in I$ and $f_i \in s_i$). For example a condition c which is defined over one single field $\mathbb{F}_1$, that is the destination IP address of the packet, and $s_1 = 1.1.1.1, 1.1.1.2$. Than only packets where the value f_1 in the field $\mathbb{F}_1$ is equal to 1.1.1.1 or equal to 1.1.1.2, that is, packets with destination address 1.1.1.1 or 1.1.1.2, satisfy the condition c.

5.3. Actions

An action is defined as a function $a : \mathbb{P} \to \mathbb{P}$, which transforms a packet. The set of all the possible actions is $\mathcal{A} = \{a : \mathbb{P} \to \mathbb{P}\}$. We extend $\mathbb{P}$ with the additional value $\emptyset$, a packet composed by no bits, the "*null packet*", that serves us to to formally describe when a packet is dropped. Henceforth the null packet will be considered $\emptyset \in \mathbb{P}$. Examples of actions are: forward (i.e., firewalls, network monitor), changing the packet (i.e., in network address translation), adding a new header (e.g., IP over IP or IPsec AH in transport mode), and encrypting the payload (e.g., IPsec ESP in transport mode).

The *action clause* $\gamma \subseteq \mathcal{T}$ defines the transformation and it is a subset of the transformation space $\mathcal{T}$. The action clause is composed by different values σ for distinct fields in a packet.

$$\gamma = \prod_{j \in I_{\mathrm{T}}} \sigma_j \qquad \sigma_j \subseteq \mathbb{F}_j \qquad I_{\mathrm{T}} \subseteq [1, m]$$

The transformation space is a subset of packet fields which are transformed by the action, I_{T} is the set of indices of those fields:

$$\mathcal{T} = \prod_{j \in I_{\mathrm{T}}} \mathbb{F}_j \qquad \mathbb{F}_j \subseteq \mathbb{F} \qquad I_{\mathrm{T}} \subseteq [1, m]$$

Some actions include:

- *ALLOW:* the simplest case is the ALLOW action that maps on the *identity function* (i.e., the function that leaves the packet unchanged);
- *LOG/MONITOR:* network monitoring and logging rules are modelled the same way as the ALLOW action, and are all represented by the identity function, since the packets are never modified but only read;
- *DENY:* a blocked packet is transformed in to a *null packet* by the function DENY: $\mathbb{P} \to \mathbb{P}$ such that $x \mapsto \emptyset$;
- *REDIRECT:* routing rules are modelled as on all interfaces where traffic is routed and the *identity function* is applied, while on all interfaces where the traffic is not routed the *null packet* is applied;
- *MODIFY:* when the action modifies certain packet fields, the action clause defines the new values this fields must be assigned to. For example, a NAT action contains the action clause with the source address field set to the IP address of the NAT;

DOI: 10.1002/nem

- *NEW HEADER:* we will assume in this model that adding a new header is always performed by changing the original packet header instead of adding a new one, without loss of generality. Therefore, a new header is modelled so that the action clause contains all header fields and is treated overriding the original header. This assumption is valid since downstream VNFs are not aware of nested headers and only the first one is considered.
- *ENCRYPT:* this action can encrypt either specific fields of the packet, or the entire payload of a specific layer. It is modelled so that the action clause contains all encrypted fields or the layer at with the encryption is applied (layer 4 PL_4, layer 7 PL_7). For example, TLS/SSL encrypts the application layer (layer 7) payload (PL_7), while IPsec encrypts the layer 4 payload (PL_4). The encryption is marked with the $\{\cdot\}$ operator. For example, $\{PL_4\}$ means that the payload at layer4 is encrypted.

5.4. Policies

In our model, a VNF-policy is a function represented as a four-tuple $(R, \mathfrak{R}, E, a_d)$ [28], where:

- $R = \{r_i\}_i$, $i \in [1, n]$ is the rule set.
- $\mathfrak{R} : 2^R \rightarrow \mathcal{A}$ is the resolution strategy used.
- $E = \{E_1, E_2, \ldots\}$ is the set of external data associated to the rules.
- a_d is the default action, applied when a packet matches no rules.

In particular we define:

- *rule:* in this model, a rule r is an association $r = (c, a)$, $c \subseteq \mathcal{C}$ and $a \in \mathcal{A}$;
- *matching process:* Given a set of rules $R = \{r_i = (c_i, a_i)\}_i$, we introduce the function $\texttt{match}_R$, returning the subset M of rules to R matching a given packet. That is, all rules where the packet satisfies the rule condition;
- *resolution strategy:* When the subset M returned by the matching process contains more than one rule, the resolution strategy $\mathfrak{R}$ is used to decide the action to apply to the packet. An example of resolution strategy is the First Matching Rule strategy (FMR). For multiple matching rules, (FMR) selects the action from the rule at highest priority;
- *external data:* external data are not part of the rules, nevertheless they are used to make decisions. The association is made using a set of external data functions $\varepsilon_k : R \rightarrow E_k$. FMR uses priorities as external data. Henceforth, the function that associates rules to priorities will be denoted by π.

6. ANOMALY ANALYSIS AND CLASSIFICATION

This section explains the different types of anomalies that might occur, their definitions and how they can be detected.

There exist three types of chain anomalies: blocked, encrypted and modified traffic anomalies. The different anomaly types are revealed by confronting the rule conditions in VNF configuration policies. Given their position in the network and depending on the network traffic we will define downstream VNF and upstream VNF.

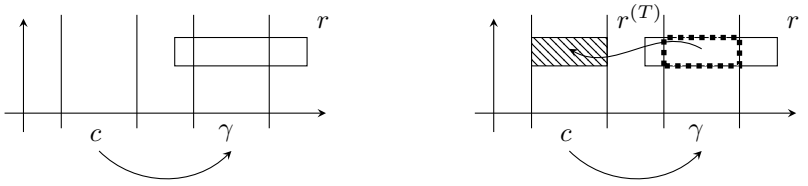
We say that two rules are correlated if they match packets belonging to the same traffic flow. Correlated rules may be found in the same VNF configuration, like the intra-policy anomalies presented by Al-Shaer for packet filters [30], or in the configuration of two distinct VNFs. A simplified case of correlated rules in distinct VNFs has been already analysed by Al-Shaer for packet filters, that named correlated rules inter-policy anomalies [6]. The inter-policy analysis for packet filters is a simple case, indeed these security controls only forward or block packets. Moreover, the approach that only looks for correlated rules is valid as long as the rules allow, deny, or monitor the traffic, but when the data is modified, the correlation between two rules can not be concluded by confronting rule conditions.

DOI: 10.1002/nem

In order to cope with the case of VNFs that also modify the traffic (with action MODIFY and NEW HEADER), we propose a process to change the configuration of upstream VNFs to report the traffic modification that will be applied by the downstream ones. The purpose is to perform changes so that all the anomalies can be detected by looking at correlated rules. That is, all the complexity is moved in a preliminary phase, which prepares policies to be analysed easily. This preliminary phase can be done once and reuse for several analyses. Note that these changes are only performed for analysis purposes. They need to be performed before the analysis, and does not need to be reported on the actual VNF configurations.

This process is named *transformation resolution* and formally described in Section 7. Essentially the transformation resolution inserts ad hoc rules for each traffic modification rule that will be encountered by packets in the downstream VNFs. The set of rules before the resolution is called *original rule set* R and the newly inserted rules are part of the *transformed rule set* $R^{(T)}$.

As an example, Figure 3 shows the policy p before (Figure 3a) and after (Figure 3b) the transformation resolution. Before the transformation resolution, the original policy only contains one rule r. Moreover, there is an upstream transformation policy with a rule that transforms packets in c (e.g., packets whose source address is in a private subnet) into packets in γ (e.g., packets with source address in a new small public subnet or single IP). For the transformation resolution to take place, r must intersect γ. After the transformation, a new rule $r^{(T)}$ is added to the policy, so that the new rule has a condition with the selectors overwritten by the condition c.



(a) A policy p with a rule intersecting some γ.

(b) The policy p after the transformation resolution.

Figure 3. Transformation resolution.

6.1. *Blocked traffic anomalies*

The blocked traffic anomaly occurs when a VNF includes a rule where the condition c matches some packets that have been blocked by another VNF or the packet is not routed to this VNF based on flow rules. This type of anomaly can be caused by a rule with action DENY or a flow rule misconfiguration.

The identification of this type of anomaly is possible as there is a linear correlation between the action of a upstream VNF and the condition of a downstream VNF rule.

The conditions of both rules intersect and a rule r_1 or a rule r_2 applies the $\emptyset$ transformation. In other words the rule r_2 is defined over the packets discarded by the rule r_1 or vice versa, that is:

$$c_1 \cap c_2 \neq \emptyset \wedge (a_1 = \emptyset \vee a_2 = \emptyset)$$

With reference to the anomaly 2 in the example in Section 3, the rule r_1 is configured on the DB and the rule r_2 is configured on the firewall FW. The rule r_1 creates a encrypted channel that is then dropped by the rule r_2, that is

$$r_1 = ip_{dest} = any \rightarrow ENCRYPT(PL_7)$$
$$r_2 = \{PL_4\} \rightarrow DENY$$

With reference to the anomaly 4 in the example in Section 3, the rule r_1 is configured on the firewall FW and the rule r_2 is configured on the intrusion detection system IDS. The firewall rule

DOI: 10.1002/nem

r_1 blocks all the traffic directed to the department$_3$, therefore the IDS can not monitor any traffic directed to 10.0.1.1, that is

$$r_1 = ip_{dest} = 10.0.1.0/24 \rightarrow DENY$$
$$r_2 = ip_{dest} = 10.0.1.1 \rightarrow MONITOR$$

6.2. Encrypted traffic anomalies

The encrypted traffic anomaly occurs when a VNF includes a rule where the condition c matches some packets that have been encrypted by an upstream VNF. This type of anomaly can be caused by a rule with the action ENCRYPT. The identification of an encrypted traffic anomaly is similar to the blocked traffic case, with the difference that it affects only the rules that contain conditions on fields that have been encrypted and therefore are no longer accessible.

This happens when a rule r_1 is applied before a rule r_2, the conditions of both rules intersect and r_1 encrypts a field used in the condition c_2. In other words, a rule r_2 is defined over the packets encrypted by rule r_1, that is

$$r_1 < r_2 \wedge c_1 \cap c_2 \neq \emptyset \wedge a_1 \cap c_2 = *$$

With reference to the anomaly 1 in the example in Section 3, the rule r_1 is configured on User1 and encrypts all the packets to 10.0.1.1 with an application layer encryption protocol, such as TLS/SSL and the rule r_2 is configured on the Monitor and must log all HTTP GET requests directed to the Global IP 10.0.1.0/24. In formulas the situation can be formalized as

$$r_1 = ip_{dest} = 10.0.1.1 \rightarrow ENCRYPT(PL_7)$$
$$r_2 = ip_{dest} = 10.0.1.0/24 \quad http_{method} = GET \rightarrow LOG$$

6.3. Modified traffic anomalies

The modified traffic anomaly occurs when a VNF includes a rule where the condition c matches the packets that have been modified by an upstream VNF. This type of anomaly can be caused by a rule with an action MODIFY or NEW HEADER.

To identify the modified traffic anomalies, one needs to verify that the condition of a upstream VNF rule that modifies the packet does not match a condition of a downstream VNF rule. Since a translation rule interferes not only with the rules by creating some anomalies, it also interferes with the anomaly detection since it resolves possible anomalies. Therefore, the model of the VNF networks must be updated accordingly so that the global behavior of the network is correct.

We have such anomalies when a rule r_1 is applied before another rule r_2, their conditions intersect, r_1 modifies a field used in the condition c_2 and both the rules must be in the original rule set. In other words the rule r_2 is defined over the packets modified by the rule r_1, that is

$$r_1 < r_2 \wedge c_1 \cap c_2 \neq \emptyset \wedge a_1 \cap c_1 = \emptyset \wedge r_1, r_2 \notin R^{(T)}$$

With reference to the anomaly 3 in the example in Section 3, the rule r_1 is configured on the Load Balancer and transforms all source addresses to 10.0.2.1 and the rule r_2 is configured on the Monitor and must log all traffic from 192.168.0.1. In formulas:

$$r_1 = ip_{src} = 192.168.0.0/24 \rightarrow NAT(ip_{src} = 10.0.2.1)$$
$$r_2 = ip_{src} = 192.168.0.1 \rightarrow LOG$$

DOI: 10.1002/nem

7. THE TRANSFORMATION RESOLUTION

When a transformation rule matches a packet, it transforms the packet as defined in the action. The resolution is only required when the transformation function is part of a rule with the action MODIFY or NEW HEADER.

We take all the rules $r = (c, a)$ in R whose condition $c = s_1 \times s_2 \times \cdots \times s_m$ intersects some action clause γ_i and create for each of them a new rule $r^{\mathrm{T}} = \left(c^{\mathrm{T}}, a\right)$ where $c^{\mathrm{T}} = s_1^{\mathrm{T}} \times s_2^{\mathrm{T}} \times \cdots \times s_m^{\mathrm{T}}$ and each $s_i^{\mathrm{T}} \subseteq \mathbb{F}_i$ is obtained as follows:

- if $\mathbb{F}_i$ is in $\mathcal{T}$ then $s_i^{\mathrm{T}} = c_i \subseteq \mathbb{F}_i$ from c_i;
- if $\mathbb{F}_i$ is not in $\mathcal{T}$, then $s_i^{\mathrm{T}} = s_i$.

Changes into the condition forced by the transformation controls are formally described by the *selector substitution function* that substitutes all the selectors in $\mathcal{T}$ of c_i with the corresponding conditions in c:

$$\Sigma(c_i, c) \mapsto c_i' = \prod_{j \in [1,m]} x_{ij}$$

where $x_{ij} = s_{ij}$ if $j \notin I_{\mathrm{T}}$ and $x_{ij} = c_{ij}$ if $j \in I_{\mathrm{T}}$.

The following functions, working on rules $r_i = (s_{i,1} \times \cdots \times s_{i,m}, a_i)$, help in making this treatment compact:

- $\mathrm{in}(\tau_i) = c$, named condition extraction;
- $\mathrm{out}(\tau_i) = \gamma_i$, named transformation-action clause extraction;
- $\Psi\left(r_i\right) = \prod_{j \in I_{\mathrm{T}}} s_{ij}$, named projection on transformation space;

where $\mathcal{C}(r_i) = c_i$.

The new rules added in the downstream policy are obtained by the *rule transformation function* Θ, defined as:

$$\Theta(r_i, \tau_j) : r_i \longmapsto r_i^{(j)}$$

where $r_i^{(j)}$ is defined as:

$$r_i^{(j)} = \begin{cases} (\Sigma(c_i, \mathrm{in}(\tau_i)), a_i) & \text{if } \Psi\left(r_i\right) \cap \mathrm{out}(\tau_j) \neq \emptyset \\ \emptyset & \text{if } \Psi\left(r_i\right) \cap \mathrm{out}(\tau_j) = \emptyset \end{cases}$$

Practically, a new rule $r_i^{(j)}$ is returned by the selector substitution function only if c_i intersects the transformation-action clause. If c_i does not intersects the transformation-action clause there is nothing to add to R, thus it returns the empty set. By abuse of notation, we will write:

$$r_i = \Theta^{-1}\left(r_i^{(j)}\right)$$

if $r_i^{(j)} = \Theta(r_i, c)$ for some τ_j.

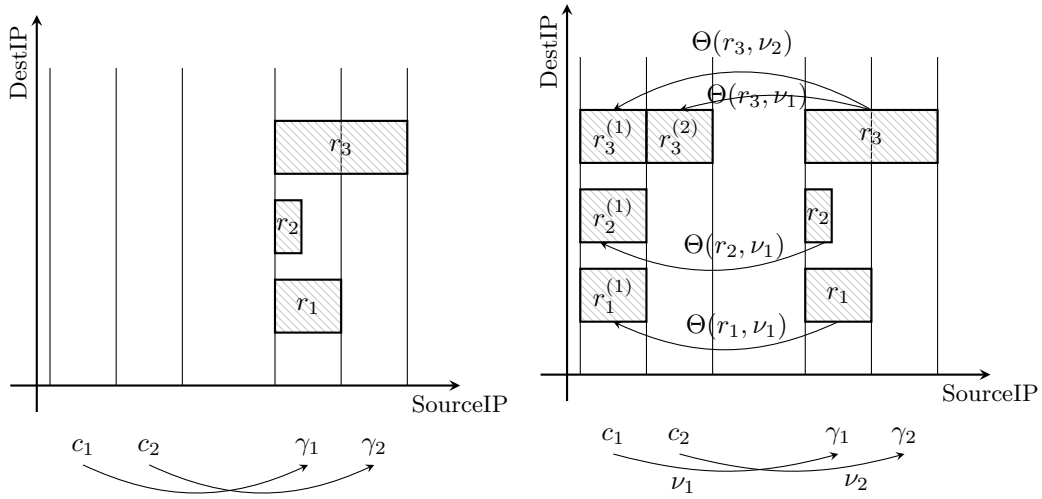
The last case to cover is when a packet is transformed but after the transformation it does not match any rule. In this case, we must ensure that the equivalent firewall enforces the default action. Therefore, for all the transformation rules whose output clause do not intersect any rule in R we create some ad-hoc rules (put at a higher priority) that enforce the rules matching the corresponding condition with the default action.

Formally, if $\forall r_i, \Psi\left(r_i\right) \cap \mathrm{out}(\tau_j) = \emptyset$ then we create the rules:

$$r^{(\tau_j)} = (\Sigma(\mathbb{F}_I, in(\tau_j)), a_d)$$

We name $D^{(T)}$ the set of all the $r^{(\tau_j)}$. It is worth noting that the cardinality of $D^{(T)}$ is always less than or equal to the cardinality of T.

DOI: 10.1002/nem



(a) A policy p with rules intersecting some γ.

(b) The policy p after the transformation.

Figure 4. Composition of transformation and filtering policies.

By applying $\Theta(r_i, \tau_j)$ for all the $r_i \in R$ and $\tau_j \in T$ and by adding the $D^{(T)}$, we obtain a set of rules:

$$R^{(T)} = \{\Theta(r_i, \tau_j)\}_{i,j} \cup D^{(T)}, \quad \text{with } i \leq m, j \leq t$$

These rules inherit the values of the external data from the rules they are derived from, i.e., the external data functions in $\{\Theta(r_i, \tau_j)\}_{i,j}$ are defined as:

$$\forall k, \varepsilon_k(r_i^{(j)}) = \varepsilon_k(r_i)$$

while for the rules in $D^{(T)}$ it is enough to suppose they have the highest priority.

Figure 4 presents an example of a policy before and after application of the rule transformation function. Figure 4a shows the original firewall configuration with three rules $\{r_1, r_2, r_3\}$, and, on the x-axis (the source IP axis), the NAT policy, composed of two rules, one mapping the x-interval c_1 into γ_1 and the second mapping c_2 into γ_2.

r_1 on the x-axis spans over the entire γ_1, r_2 partly covers γ_1, and r_3 spans over the union of γ_1 and γ_2. Figure 4b shows the resulting equivalent VNF. Both r_1 and r_2 have been "moved" and stretched to fit into c_1 to originate $r_1^{(1)}$ and $r_2^{(1)}$; r_3 has been first split into two rules, corresponding to subsets γ_1 and γ_2, then the single sub-rules have been moved and stretched to fit into c_1 and c_2, respectively, originating $r_3^{(1)}$ and $r_3^{(2)}$.

The *T-modified resolution strategy* $\mathfrak{T}$ is defined as:

$$\begin{array}{rcl} \mathfrak{T} : 2^{R \cup R^{(T)}} & \rightarrow & \mathcal{A} \\ M_{\text{eq}} \subseteq R \cup R^{(T)} & \longmapsto & \begin{cases} \mathfrak{R}(H) & \text{if } M^{(T)} \neq \emptyset \\ \mathfrak{R}(M) & \text{if } M^{(T)} = \emptyset \end{cases} \end{array}$$

where $M_{\text{eq}} = \text{match}_{R \cup R^{(T)}}(x) = M \cup M^{(T)}$, $M \subseteq R$, $M^{(T)} \subseteq R^{(T)}$, and $H = \{\Theta^{-1}\left(r_i^{(j)}\right) \mid r_i^{(j)} \in M^{(T)}\} \subseteq R$.

By composing the transformation policy defined by T with the policy $(R, \mathfrak{R}, E, d)$, we obtain the T-modified p policy, denoted as $p^{(T)}$, which can be represented as:

$$(R \cup R^{(T)}, \mathfrak{T}, E, d)$$

DOI: 10.1002/nem

	Source IP	PortS	Dest IP	PortD	Proto	Protocol States	Action
$r_5^{(1)}$	192.168.0.1	>1023	10.1.1.1	80	TCP	NEW	ALLOW
$r_6^{(2)}$	192.168.0.2	>1023	10.1.1.1	80	TCP	NEW	ALLOW
r_1	any	any	any	any	any	ESTABLISHED RELATED	ALLOW
r_2	10.1.2.7	any	10.1.1.9	any	any	any	DENY
r_3	10.1.2.7	any	10.1.1.19	any	any	any	DENY
r_4	10.1.2.0/24	>1023	10.1.1.0/24	80	TCP	NEW	ALLOW
r_5	10.2.2.1	>1023	10.1.1.1	80	TCP	NEW	ALLOW
r_6	10.2.2.2	>1023	10.1.1.1	80	TCP	NEW	ALLOW
...							
	∞	*	*	*	*	*	D

Table I. An excerpt of the filtering rules in a firewall after a NAT transformation.

It is worth noting that the T-modified resolution strategies are sound. That is, they give a result for any subset of $R \cup R^{(T)}$, because they always decide by using $\mathfrak{R}$ and, if available, external data are inherited from the original rules. Moreover, this works also for resolution strategies requiring unique values of the external data, such as FMR priorities. In fact, after the transformation, there are rules with duplicated external data values but they are never used in the same decision, as either original or transformed rules are taken into account.

Table I presents a policy after a NAT transformation, where the rules $r_5^{(1)}$ and $r_5^{(1)}$ are new and inserted with the highest priority. Rule $r_5^{(1)}$ is created from the intersection of γ_1 with r_5, and rule $r_6^{(2)}$ is created from the intersection of γ_2 with r_6. γ_1 is part of a NAT rule which transforms the source IP from 10.2.2.1 to 192.168.0.1. γ_2 is part of a NAT rule which transforms the source IP from 10.2.2.2 to 192.168.0.2. All the other rules remain untouched because their rule clause does not intersect with the source IPs 10.2.2.1 and 10.2.2.2.

7.1. Multiple transformations

If, before reaching a firewall, a packet encounters first the transformation control T_1 then the T_2, we generate the (T_1, T_2)-modified p policy $p^{(T_1,T_2)}$ recursively. That is, $p^{(T_1,T_2)}$ is the T_1-modified p^{T_2} policy:

$$p^{(T_1,T_2)} = \left(p^{(T_1)}\right)^{(T_2)}$$

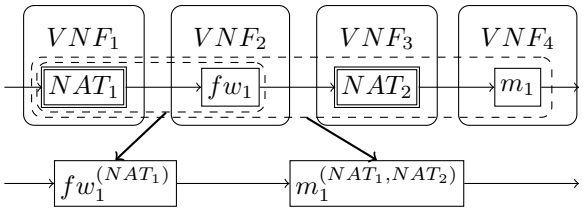


Figure 5. Composition of four cascaded VNFs with 2 NAT, 1 firewall and 1 monitor.

In these cases, the rule set is $R \cup R^{(T_1)} \cup \left(R \cup R^{(T)}\right)^{(T_2)} = R \cup R^{(T_1)} \cup R^{(T_2)} \cup R^{(T_2,T_1)}$. The resolution strategy obtained from these two transformations works in stages. First, it checks if some of the newly generated rules in $R^{(T_2,T_1)}$ apply, then it checks in $R^{(T_2)}$, then it checks if some rule from the previous transformation apply ($R^{(T_1)}$), and finally checks in R. Let's consider the scenario in Figure 5, with two serially connected filtering devices configured to perform pre- and post-NAT. The steps to obtain the equivalent firewall are:

- compute the NAT_1-modified fw_1 (filtering) policy $fw_1^{(NAT_1)}$ (i.e., the inner dashed rectangle in Figure 5);

DOI: 10.1002/nem

- compute the (NAT_1, NAT_2)-modified m_1 (monitoring) policy $m_1^{(NAT_1,NAT_2)}$ (i.e., the external dashed rectangle in Figure 5).

7.2. Inverse transformations

Another very important case to consider when working with tunnels is the inverse transformation. In fact, NAT/NAPT controls perform the inverse transformation for back traffic based on state information without the need of an explicit rule. For the tunnelling, this is not true and the inverse transformation must explicitly indicate the other end-point policy. Even if the inverse transformation can be considered as an independent one is evidently a sub-optimal case. For this reason, we pre-process the transformation controls in the path to identify and eliminate the inverse rules.

This scenario usually happens when a gateway with a policy p_1 contains a rule $r_1 = (c_1, a_1)$ and another one with a policy p_2 contains a rule $r_2 = (c_2, a_2)$. However, the rule annihilation must be considered even in a subset of the condition and action clauses. That is, given $c_1 \mapsto \gamma_1$ and $c_2 \mapsto \gamma_2$, the annihilation happens if $c_2 \subseteq \gamma_1$. In this case, instead of eliminating both the rules, we simply leave in one of the transformation controls the "biggest" rule, that is, either $c_1 \setminus c_2 \mapsto \gamma_1$ or $c_2 \setminus c_1 \mapsto \gamma_2$.

8. PERFORMANCE TESTS

The performance tests focus on the execution time of the Policy Analyser itself and not its interactions with other NFV components. In order to eliminate all the possible network related delays and differences in NFV implementations, the tests were executed isolated from a deployed network. As previously stated the Policy Analyser receives as input the same information required by the NFV orchestrator, being a sub-component of the latter. The first performance evaluation we performed is based on the representation of a campus network. Furthermore, since it is infeasible to perform a great number of tests on real networks of a reasonable size, we also performed extensive performance and scalability tests on several synthetic networks. Our model was implemented in Java 1.6 and run inside a virtual machine using two cores of an Intel Core i7-3630QM (2.4 GHz) CPU, with 16 GB RAM under Debian Linux 8 operating system.

8.1. Campus network

The campus network of the first test is represented in Figure 6. The network is composed of 15 firewalls, 15 NAT devices, about 10,000 hosts, 36 filtering zones with private IP addresses and 30 additional zones with public IP addresses. All filtering zones with private IP addresses are connected through a NAT devices with the rest of the network. The network has a star topology and in detail is composed by the following parts:

- The core router is connected to the Internet through the border firewall $\mathrm{fw_I}$.
- The boarder firewall contains about 150 rules and acts as a VPN concentrator for external remote connections.
- There are 11 department networks, composed of 2 zones each: a private zone, connected to the core router by means of a NAT/NAPT and a firewall ($\mathrm{fw_{DIP}}$), and a public zone connected just via the firewall ($\mathrm{fw_{DIP}}$) to the core router. The private zone can reach the campus network and the Internet but is not visible from outside. The public zone is reachable from Internet.
- There are three laboratory networks composed by 5 zones each and connected to the core router via an individual NAT/NAPT device (NAT). These laboratories can access the campus network and the Internet, but are not visible from the outside.
- The administration network is composed by 5 zones for its servers and an additional 5 zones for the employees' workstations. A firewall ($\mathrm{fw_A}$) containing approximately 250 rules connects all these zones to the core router. The services in the 5 server zones have higher security requirements than the ones allocated in the data centre and are accessed mainly by the users from the 5 administration user zones.

DOI: 10.1002/nem

- The data center is composed by 10 zones and contains various servers (web, e-mail, file servers, ...) some of which are also accessible from Internet. A single firewall ($\mathrm{fw_{DC}}$) with about 50 rules connects the date center to the core router.
- A device ($\mathrm{fw_{WLAN}}$) with about 20 rules (acting as NAT/NAPT and firewall) connects the 10 wireless networks to the core router. Wireless clients can access the Internet and the public parts of the campus network.

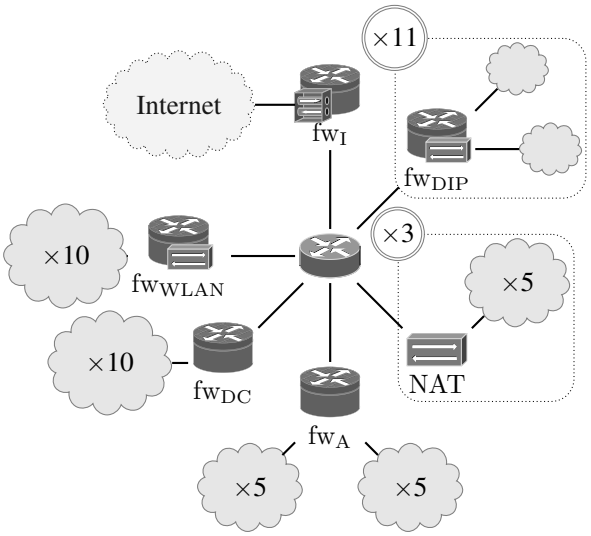


Figure 6. The campus network.

Although the campus network is not a VNF deployment it is still a valid performance evaluation because the network topology and involved configurations are similar. We transformed the network topology and the involved configurations into the required input format for the Policy Analyser. Each pair of zones in the network topology is interconnected through different paths and therefore through different transformations. The Policy Analyser applied the transformation resolution for all 4624 possible paths and analysed the resulting policies. The complete execution took about 3.7 minutes and the identified anomalies helped to improve the network configurations.

8.2. Synthetic network

For the scalability tests on synthetic networks we used a random policy generator. However, generating meaningful policies with a great number of rules is a complex task. Indeed, rules in cascaded VNFs must be related to each other and allow end-to-end communication paths. We ensured reachability on each path by inserting "allow rules" and generated rules according to the statistical data from Taylor [31] with an average number of anomalies according to Al-Shaer [30]. These works estimate that at most 5 rules intersect simultaneously and that no more than 25% of rules intersect each other. Therefore, random policies that have such statistical properties should represent very well (worst case) VNF policies.

The synthetic network path contains a variable number of cascaded VNFs and each VNF is configured with a variable number of rules. Additionally the network also contains transformation policy either before or after a VNF. The translation policies include one of the common transformation rules: NAT 1-to-1, NAT many-to-1, and tunnelling. We used three independent parameters in our performance evaluation:

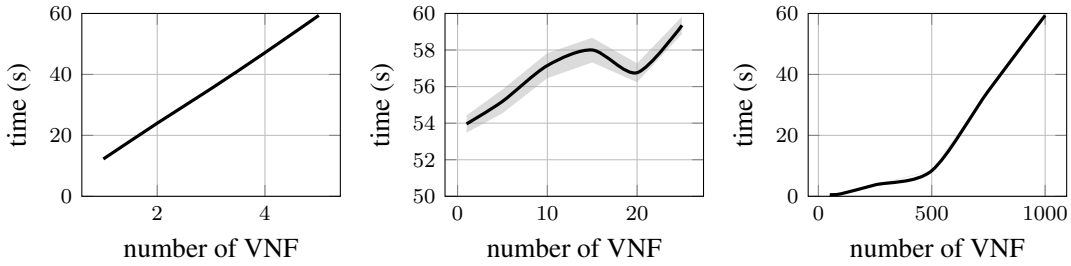
- f, the number of VNFs in each path, variable from 1 to 5;
- n, the number of rules per VNF, which varies from 50 to 1000;
- t, the number of translation policies for each path, which ranges from 1 to 25.

DOI: 10.1002/nem

The most complex case tested considered paths with 25 translation policies and 5 VNFs, with 1000 rules each. Basted on this numbers we are confident that the test represent worst case scenarios, and that the Policy Analyser performance is better on real VNF deployments.

We also performed test to evaluate the impact of the rule statistics on the performance. The results show that the statistics have almost no impact on the execution time. The reason for this is because the transformation resolution creates new rule thus alters the statistics significantly. However all policies are generated with the statistics from Taylor and Al-Shaer, to ensure the most accurate base case.

We performed each test case 100 times and the result shown in the graphs is the average value. For all the tests we calculated a 95% confidence interval, shown in the figures as a grey area around the graph curves. The confidence interval is very small, as shown in Figure 7b where the range of the y-coordinate is reduced. This small confidence interval proves that the Policy Analyser will have similar worst case performance on real networks with similar statistical properties.



(a) Time depending on f
$n = 1000, t = 25$

(b) Time depending on t
$f = 5, n = 1000$

(c) Time depending on n
$f = 5, t = 25$

Figure 7. Time to analyse a VNF path.

Figure 7 shows the average time needed to analyse a VNF path. The first plot (Figure 7a) shows the time to analyse a VNF path when the number of VNFs varies from 1 to 5, if the number of rules per VNF is fixed at 1000 and the number of transformations per path is fixed at 25. The second plot (Figure 7b) shows the analysis time when the number of transformations per path varies from 1 to 25, if there are 1000 rules per VNF and exactly 5 VNF per path. Finally, the third plot (Figure 7c) shows the time required when rules vary from 100 to 1000, if the number of VNFs per path is 5 and there are 25 transformations per path.

It is possible to see that the computation time depends linearly on the number of VNFs in the path (Figure 7a) and the number of translation rules (Figure 7b). The number of rules is the factor that affects the execution time the most. As shown in Figure 7c, the computation time quickly increases with the number of rules and, in the worst case scenario, reaches about 60 s.

These results prove that the analysis time is reasonably low and, furthermore, the majority of the VNFs contain less then 200 rules [32]. Our test proves that in such cases it takes less then 3 s to analyse a VNF path and therefore, for the majority of cases our implementation performs more than satisfactorily. This numbers are also confirmed by the results from the campus network, which is much smaller as the cases in the synthetic tests. Each path in the campus example has three devices with on average less than 100 rules and four transformations. On average the analysis of one path in the campus network took about 0,05 s.

9. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a formal model able to detect inter-function anomalies in a SDN/NFV deployment scenario. Our approach is able to cope with a wide array of different network functions, such as firewalls, NAT/NAPT devices, traffic monitors and encryption devices.

DOI: 10.1002/nem

The presented model can detect several kinds of errors and anomalies that originate from correlations between configuration rules of different network functions. In particular, we have presented three types of anomalies: blocked traffic, modified traffic and encrypted traffic.

We have provided a detailed description of the underlying mathematical model and developed a prototype. The prototype can be integrated in a VNF deployment within the orchestrator. It revives all required informations about the network and its configurations from the orchestrator. The results can be interpreted directly by the orchestrator or presented as a report to the administrator.

The prototype has been implemented in Java and we evaluated its correctness and performance. We tested the implementation on a campus network and on several synthetically generated networks. The results prove to be efficient and scalable, as it is able to check thousands of rules distributed on several functions in a few seconds.

In the future we are aiming to add support for a smart and optimized resolution of the conflicts. The model will then be able to remove redundancies and propose an ad-hoc allocation of the rules to the administrators. This will increase the performance of the network without altering its semantic.

We also aim to integrate the policy analysis module in an open source NFV implementation. This would provide the NFV orchestrator with additional information, increasing both the security and the efficiency of the deployment.

ACKNOWLEDGMENT

The research described in this paper is part of the SECURED project, co-funded by the European Commission (FP7 grant agreement no. 611458).

REFERENCES

1. Verizon, “2015 Data Breach Investigations Report” 2015.
2. B.Carpenter, S.Brim, “Middleboxes: Taxonomy and Issues”, RFC-3234, February 2002.
3. S.Spinoso, M.Leogrande, F.Risso, R.Sisto, S.Signgh, “Automatic Configuration of Opaque Network Functions in CMS”, NVSDN 2014: 1st International Workshop on Network Virtualization and Software-Defined Networks for Cloud Data Centres, London, United Kingdom, December 2014, pp. 750–755.
4. Internet Engineering Task Force, “Interface to Network Security Functions (I2NSF)” 2015.
5. European Telecommunications Standards Institute, “Network function virtualization - White Paper 2”, October 2013.
6. E.Al-Shaer, H.Hamed, “Discovery of policy anomalies in distributed firewalls”, INFOCOM2004 : 23rd IEEE Int. Conference on Computer Communications, Hong Kong, Cina, March 7–11 2004, pp. 2605–2616, doi:10.1109/INFCOM.2004.1354680.
7. E.Al-Shaer, H.Hamed, W.Marrero, “Modeling and Verification of IPSec and VPN Security Policies”, ICNP2015 : 13th IEEE Int. Conference on Network Protocols, Boston, MA, November 6–9 2005, pp. 259–278, doi:10.1109/ICNP.2005.25.
8. A. X.Liu, M. G.Gouda, “Complete Redundancy Detection in Firewalls”, DBSec2005 : 19th IFIP WG11.3 Conference on Data and Applications Security, Storrs, CT, August 7-10 2005, pp. 193–206, doi:10.1007/11535706_15.
9. J. G.-A.F. Cuppens, N. Boulahia, “Detection and Removal of Firewall Misconfiguration”, CNIS2005 : Communication, Network, and Information Security, Phoenix, AZ, November 14–16 2005.
10. L.Yuan, H.Chen, J.Mai, C.-n.Chuah, “FIREMAN: A Toolkit for FIREwall Modeling and ANalysis”, SP2006 : IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, May 21–24 2006, pp. 199–213, doi:10.1109/SP.2006.16.
11. C.Basile, A.Lioy, “Analysis of Application-layer Filtering Policies with Application to HTTP”, IEEE/ACM Trans. Netw., Vol. 23, No. 1, Feb. 2015, pp. 28–41, doi:10.1109/TNET.2013.2293625.
12. R.Mahajan, D.Wetherall, T.Anderson, “Understanding BGP Misconfiguration”, ACM SIGCOMM Computer Communication Review, Vol. 32, No. 4, August 2002, pp. 3–16, doi:10.1145/964725.633027.
13. N.Feamster, H.Balakrishnan, “Detecting BGP Configuration Faults with Static Analysis”, NSDI05 : 2nd Conference on Symposium on Networked Systems Design & Implementation, Boston, MA, May 2–4 2005, pp. 43–56.
14. R.Alimi, Y.Wang, Y. R.Yang, “Shadow Configuration As a Network Management Primitive”, ACM SIGCOMM Computer Communication Review, Vol. 38, No. 4, August 2008, pp. 111–122, doi:10.1145/1402946.1402972.
15. Z.Fu, S. F.Wu, H.Huang, K.Loh, F.Gong, I.Baldine, C.Xu, “IPSec/VPN Security Policy: Correctness, Conflict Detection, and Resolution”, POLICY2001 : 2nd IEEE Int. Workshop on Policies for Distributed Systems and Networks, Bristol, UK, January 29–31 2001, pp. 39–56, doi:10.1007/3-540-44569-2_3.

16. C.Basile, D.Canavese, A.Lioy, F.Valenza, “Inter-technology Conflict Analysis for Communication Protection Policies”, CRISIS2014 : 9th Int. Conference on Risks and Security of Internet and Systems, Trento, Italy, August 27-29 2014, pp. 148–163, doi:10.1007/978-3-319-17127-2_10.
17. F.Valenza, S.Spinoso, R.Basile, Cataldo Sisto, , A.Lioy, “A Formal Model of Network Policy Analysis”, 4th Cyber Security and Privacy Innovation Forum on, Torino, Italy, September 2015.
18. N.McKeown, T.Anderson, H.Balakrishnan, G.Parulkar, L.Peterson, J.Rexford, S.Shenker, J.Turner, “OpenFlow: Enabling Innovation in Campus Networks”, ACM SIGCOMM Computer Communication Review, Vol. 38, No. 2, March 2008, pp. 69–74, doi:10.1145/1355734.1355746.
19. E.Al-Shaer, S.Al-Haj, “FlowChecker: configuration analysis and verification of federated openflow infrastructures”, SafeConfig10 : 3rd ACM workshop on Assurable and usable security configuration, Chicago, USA, October 4-8 2010, pp. 37–44, doi:10.1145/1866898.1866905.
20. M.Canini, D.Venzano, P.Perešíni, D.Kostić, J.Rexford, “A NICE way to test openflow applications”, NSDI12 : 9th USENIX conference on Networked Systems Design and Implementation, San Jose, CA, April 25-27 2012, pp. 10–10.
21. B.Lopes Alcantara Batista, G. A.Lima de Campos, M. P.Fernandez, “Flow-based conflict detection in OpenFlow networks using first-order logic”, ISCC2014 : IEEE Symposium on Computers and Communication, Funchal, Portugal, June 23-26 2014, pp. 1–6, doi:10.1109/ISCC.2014.6912577.
22. A.Tavakoli, M.Casado, T.Koponen, S.Shenker, “Applying NOX to the Datacenter”, HOTNETS09 : 8ht ACM Workshop on Hot Topics in Networks, New York City, NY, October 22-23 2009.
23. A.Khurshid, W.Zhou, M.Caesar, P.Godfrey, “Veriflow: verifying network-wide invariants in real time”, ACM SIGCOMM Computer Communication Review, Vol. 42, No. 4 2012, pp. 467–472, doi:10.1145/964725.633027.
24. E.Al-Shaer, W.Marrero, A.El-Atawy, K.Elbadawi, “Network configuration in a box: towards end-to-end verification of network reachability and security”, ICNP2009 : 17th IEEE Int. Conference on Network Protocols, Princeton, NJ, October 13–16 2009, pp. 123–132, doi:10.1109/ICNP.2009.5339690.
25. E.Al-Shaer, W.Marrero, A.El-Atawy, K.Elbadawi, “Network configuration in a box: towards end-to-end verification of network reachability and security”, ICNP2009 : 17th IEEE Int. Conference on Network Protocols, Princeton, NJ, October 13-16 2009, pp. 123–132, doi:10.1109/ICNP.2009.5339690.
26. A.Panda, O.Lahav, K. J.Argyraki, M.Sagiv, S.Shenker, “Verifying Isolation Properties in the Presence of Middleboxes”, http://arxiv.org/abs/1409.7687, September 2014.
27. S.Spinoso, M.Virgilio, W.John, A.Manzalini, G.Marchetto, R.Sisto, “Formal Verification of Virtual Network Function Graphs in an SP-DevOps Context”, ESOCC2015 : 4th European Conference Service Oriented and Cloud Computing, Taormina, Italy, September 15-17, 2015. Proceedings 2015, pp. 253–262, doi:10.1007/978-3-319-24072-5_18.
28. C.Basile, A.Cappadonia, A.Lioy, “Geometric Interpretation of Policy Specification”, POLICY2008 : IEEE Workshop on Policies for Distributed Systems and Networks, Palisades, New York, June 2–4 2008, pp. 78–81, doi:10.1109/POLICY.2008.36.
29. A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser, “Terminology for Policy-Based Management”, RFC-3198, November 2001.
30. E.Al-Shaer, H.Hamed, “Modeling and Management of Firewall Policies”, IEEE Trans. Netw. Service Manag., Vol. 1, No. 1, April 2004, pp. 2–10, doi:10.1109/TNSM.2004.4623689.
31. D.Taylor, “Survey and taxonomy of packet classification techniques”, ACM Comput. Surv., Vol. 37, No. 3, September 2005, pp. 238–275, doi:10.1145/1108956.1108958.
32. Algosec Survey Insights, “Examining the Dangers of Complexity in Network Security Environments” 2012.