A Formal Model of Network Policy Analysis

(Article begins on next page)

18 April 2024

A Formal Model of Network Policy Analysis

Fulvio Valenza, Serena Spinoso, Cataldo Basile, Riccardo Sisto and Antonio Lioy

Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy

Email: {fulvio.valenza, serena.spinoso, cataldo.basile, riccardo.sisto, antonio.lioy}@polito.it

Abstract—**The complexity of network topology together with heterogeneity of network services make the network configuration a hard task, even for skilled and experienced administrators. In order to reduce the complexity of the network configuration, administrators have leveraged network policies, introducing hence new possibility of error. Indeed, erroneous and unexpected network behaviour (e.g., security flaws) can derive from the wrong network policy definition, but also from the possible anomalies among policies of different domains.**

This paper presents a formal model for detecting inter- and intra-domain policy anomalies. Policy anomalies allow administrators to identify all the network behaviours they consider erroneous or to be monitored. To validate the generality of the proposed solution, the model has been applied to three policy domains (packet filtering, communication protection and service function chaining) and the impact of an anomaly detection analysis was tested in different sized networks.

I. INTRODUCTION

Nowadays networks are very complex systems to manage for administrators, due to the number and type of services available: for example, monitoring on network nodes, traffic steering and also network security and traffic filtering assurances. Due to this complexity, administrators can introduce errors and redundancy in the configuration of the available services, causing erroneous and unexpected network behaviours [1].

In order to simplify the administrator work, a different network configuration approach based on the use of network policies was suggested in literature in [2]. Even though this policy-based system could appear as the solution, the problem of erroneous network configuration was just moved to the possible wrong network policy definition. Indeed there have been different works on the analysis of network policy conflicts and anomalies. Those works have been carried on in different policy application domains: communication protection [3], filtering [4], service function chaining [5] and others. Unfortunately, such works have not been applied into the real activities of a network administrator, as well as the major part of them generally is limited to a specific policy application domain (*intra-domain*), overlooking anomalies between different policy domains (*inter-domain*).

Hence it could be useful a detection system that gives an overview of the network errors and conflicts deriving from the wrong policy definition. Moreover, having a dashboard containing any irregular network conditions and events that an administrator wants to monitor and to be alerted (i.e., no errors and conflicts) could make more flexible and efficient the network configuration task. Such network errors, conflicts, irregular events can be identified through the definition of

policy anomalies, either *intra-domain* or *inter-domain*. In our view, an anomaly arises when the effects of one network policy is influenced or altered by other policies (one or more).

To better understand such policy anomalies, let us consider an enterprise network, where communication protection policies are defined to encrypt all the traffic sent to the Internet. In this example, an administrator would like to be advertised if there are gateways that decrypt and encrypt the received packets to forward them through the secure path, in order to check the trustworthiness of that nodes. This is due to fact that the secure path between the source and destination nodes can be composed of multiple channels (e.g., remote access VPN), of which the end-points could be untrusted. Also an example of inter-domain policy anomalies is when an administrator configures a filtering policy to drop all encrypted traffic sent to the Internet for monitoring activities. Hence, this filtering policy collides with the communication protection policy of the previous example, where all traffic routed to the Internet is encrypted.

Thanks to its usefulness, a detection system of policy anomalies can be used by several actors (e.g., administrators, network operators...) and applied to different scenarios. An example is the Software Defined Networking (SDN) environment, where the current trend is to make networks more programmable [6]. Recent works propose the configuration of forwarding policies [7] to specify which traffic flows should be processed by an ordered and user-defined set of network functions (e.g., DPI, NAT, load balancer, etc...), implementing the Service Function Chain concept [8].

Hence, a possible extension is to enable the SDN controller to perform the detection task before starting the policy refinement process and, of course, before changing the network configuration. In this way, an administrator can be sure that unexpected and erroneous network behaviour will not occur after a network configuration update.

In line with the SFC paradigm, recently, the innovation of the Network Function Virtualization (NFV) [9] has enabled more powerful and flexible services to the end-users. Many SDN/NFV-related solutions, in fact, allow users to define their own service graph, where users can specify all the desired network services for processing their traffics, such us in the UNIFY project [10]. Here the use of network policies belonging to different application domains (e.g, security, forwarding, verification graph policy...) could enrich the provided service, as it is already partially envisioned within the UNIFY project [11][12]. In this kind of scenario, in fact, an inter-domain anomaly detection is particularly meaningful in order

to guarantee a correct service provisioning to the end-users.

The contribution of this paper is to present a formal model for detecting network conflicts and irregularities by defining *inter-* and *intra-domain* policy anomalies, in order to avoid erroneous and unexpected network behaviours. The proposed model has been applied also to three case studies of different policy domains. In order to simplify the model understanding, one of such case studies (Filtering Policy) is analysed in line with the model description (Section II). In Section III, the other two case studies are described in detail, while Section IV gives an overview of a possible implementation of the proposed model. Finally, we conclude the paper by presenting the different works that have provided our background on the policy conflict analysis (Section V) and some possible future works (Section VI).

II. THE APPROACH

In this section, we propose a formal model, which is able to both represent network policies of different domains and to detect the intra- and inter-domain anomalies among such policies. In order to perform those targets, the model is composed of four sets, that are:

- *network fields*, atomic elements that identify information a network policy should keep trace. Examples of such network fields (but are not limited to) could be the packet headers. In fact, other information (e.g., network node ID, traffic label, cipher algorithm etc...) could be needed to designate the events and conditions an administrator wants to manage;
- *policy actions*, a set of atomic elements that represent either the real action performed by a network node (e.g., a firewall is configured to deny or allow a packet under certain conditions), or the parameters and information that characterized that action (e.g., algorithm, technologies, protocols to use);
- *Policy Implementations* (*PIs*), data-structures to pinpoint in a formal and abstract way the policy rules enforced by a network node for a certain domain. This means that the *PI* data-structure must be defined so that the *PI*: *(i)* identifies the condition and events that administrators want to manage through conditions expressed on the network fields; *(ii)* knows the policy actions that describe the way those events are managed; *(iii)* is designed to be applied to a specific policy domain;
- *detection rules* are a set of conditions that distinguish the possible anomalies among *PIs*. For a particular policy domain, it is possible to exploit the existing works on the policy analysis for that domain. Otherwise an administrator could define his set of *PI* anomalies, either intra-domain and inter-domain.

In order to simplify the understanding of this model and its main elements, we can consider the real use case of the Filtering Policies (FP) as example. FP are usually used in a single- or multi-firewall environment to defend networks by filtering unwanted or unauthorized traffic from or to the secured network. Actually, many works exist in this domain, which focus on the FP conflict definition like the work by Al-Shaer *et al.* [4]. In particular we have extrapolated the main network fields and actions needed to distinguish the filtering conflicts presented by the authors in order to build the model specific for the FP domain. Those fields and actions are:

- an incremental firewall identifier (f) to reflect the order in which the received traffic is processed by a sequence of firewalls;
- an incremental firewall rule identifier (r), valid within the firewall identified by f;
- the source and destination IP addresses of the traffic (ip_src and ip_dst);
- the protocol type (t) that can assume TCP, UDP or * (don't care) as value;
- the ranges of the source and destination port numbers (p_src and p_dst);
- the policy action (a) that the firewall must carry out if the received traffic matches with the current policy and that can assume either accept or deny as value.

Hence the PI that defines a filtering policy is an ordered set of network fields and policy actions and can have the following structure:

$$pi_{fp} = (f, r, ip\_src, ip\_dst, t, p\_src, p\_dst, a)$$

Now we start to analyse in depth the *PI* structure and then the *PI* anomalies.

A. The PI structure

A *PI* has a structure composed of a sequential set of network fields (*n*) and a set of policy actions (*a*):

$$pi_i = (n_{i1}, n_{i2}, ..., n_{in}, a_{i1}, a_{i2}, ..., a_{in})$$

Actually, a *PI* has a different set of network fields and policy actions, based on the domain where the *PI* is defined. In addition, among the network fields and policy actions of the *PIs*, a set of relations $\mathfrak{R}$ must be defined in order to establish the *PI* anomalies. In detail, the proposed model supports four relations $\mathfrak{R}$ between network fields (the same relations can be defined for the policy actions):

- **equivalence**: two network fields are equivalent (or equal) if they have exactly the same value;

$$n_{i1} = n_{j1}$$

- **dominance**: a network field dominates another one if it is a generalization of the latter. For example, n_{i1} is the IP addresses 1.1.*.* and n_{j1} is the IP address 1.1.1.*, in this case n_{i1} dominates n_{j1};

$$n_{i1} \succ n_{j1}$$

- **correlation**: two network fields are correlated if they share some common values, but none of them includes (or dominates) the other one. For example, if n_{i1} and n_{j1} are port number and n_{i1} ranges from 1 to 75, while

n_{j1} from 50 to 100, then they are correlated because the range [50, 75] is shared by both fields;

$$n_{i1} \sim n_{j1}$$

- **disjointness**: two network fields are disjoint if they do not share any value. On the other hand, if a network field is equivalent, correlated or dominates another, those fields are *not-disjointed* ($n_{i1} \not\perp n_{j1}$).

$$n_{i1} \perp n_{j1}$$

A certain relation can be applied to a network field (or a policy action) depending on the type of the network field value (e.g., integer, IP address, boolean, enumeration and more). For example, a port number can be equal to another one, but it cannot dominate a range of port numbers. Instead a range of port numbers can dominate a single value of port number.

The proposed set of relations allows the model to achieve high-level flexibility and generality in order to: *(i)* do not be limited to a single policy domain; *(ii)* detect policy anomalies among different domains; *(iii)* enrich the set of policy anomalies (inter- and intra-domain) by allowing administrators to define their own set of anomalies. In fact this set of relations $\Re$ gives the means to impose conditions on *PI* elements, whose value types are not known a-priori.

To clarify how those relations can be applied, let us consider again the example of a filtering policy. Having identified the *PI* structure to detect the filtering policy anomalies that Al-Shaer *et al.* have defined [4], the set of the relations $\Re$ can be applied to the FP network fields and policy action so that:

- f_i and r_i can be equal to or can dominate the relative field of pi_j (e.g., $f_i \succ f_j$, if f_i is equal to 6, while f_j is equal to 3);
- ip_src, ip_dst, p_src and p_dst can be equal, disjointed or can dominate the relative fields of pi_j;
- finally, a_i and t_i can be equal or disjointed to t_j and a_j.

B. The PI anomaly detection rules

The *PI* anomalies (*I*) are defined in form of *detection rules*. The detection rules are, in turn, defined according to the existing relations $\Re$ among the network fields and policy actions of a *PI*.

In the proposed model, the detection rules are based on the First Order Logic (FOL) and are expressed using the Horn clauses. Horn clauses are frequently encountered in model theory because they exhibit a simple and natural rule-like form. Also, these clauses can be easily translated in many different logic programming languages, such as Prolog, or generic programming language such as C or Java. In particular, the Horn clauses can be simply used to represent all the axioms used in the proposed model, expressed in the form of disjunction of literals (clauses) with at most one positive literal:

$$\neg C_1 \vee \neg C_2 \vee \cdots \vee \neg C_n \vee I$$

Alternatively, they can be expressed in a more natural way as a set of positive conditions implying an assertion and this is the form chosen in our model:

$$C_1 \wedge C_2 \wedge \cdots \wedge C_n \Rightarrow I$$

In our model, also, every clause is the relation between network fields (e.g., n_k) or policy actions (e.g., a_k) of one or more *PIs* (e.g., pi_i and pi_j), even belonging to different domain, like this example:

$$C_1 := n_{ik} \,\Re\, n_{ih}, \; C_2 := n_{ik} \,\Re\, n_{jh}$$
$$C_3 := a_{ik} \,\Re\, a_{ih}, \; C_4 := a_{ik} \,\Re\, a_{jh}$$

where n_{ik} and n_{jh} (or a_{ik} and a_{jh}) identify two generic network fields (or policy actions) in the ordered sequence of network fields (or actions) in the *PI* structure.

Hence possible detection rules can take a form like the following, but they are not limited to this structure:

$$n_{iq} \,\Re\, n_{jq} \,\wedge\, ... \,\wedge\, n_{ik} \,\Re\, n_{jh} \,\wedge\, ... \,\wedge\, a_{ik} \,\Re\, a_{jh} \,\Rightarrow I$$

As example of detection rules, we can consider again the anomaly classification of filtering policy proposed in [4]. In this case, we show the detection rules of two examples of anomalies. Those detection rules follow the aforementioned definition of *PI* for FP and are:

- *Intra-Firewall Shadowing anomaly* occurs when two *PIs* (pi_i and pi_j) match the same traffic, but they enforce different actions

$$f_i = f_j \wedge r_j \succ r_i \wedge ip\_src_i \succeq ip\_src_j \wedge t_i \succeq t_j \wedge$$
$$ip\_dst_i \succeq ip\_dst_j \wedge p\_src_i \succeq p\_src_j \wedge p\_dst_i \succeq p\_dst_j$$
$$\wedge \, a_i \neq a_j \Rightarrow \text{Shadowing}(pi_i, pi_j)$$

- *Inter-Firewall Redundancy anomaly* occurs when *PIs*, belonging to different firewalls, have the same pattern matching and the same action of blocking the traffic

$$f_j \succ f_i \wedge ip\_src_i \succeq ip\_src_j \wedge ip\_dst_i \succeq ip\_dst_j \wedge$$
$$p\_src_i \succeq p\_src_j \wedge p\_dst_i \succeq p\_dst_j \wedge t_i \succeq t_j \wedge$$
$$a_i = a_j = \text{deny} \Rightarrow \text{Redundacy}(pi_i, pi_j)$$

It is worth noting that detection rules identify the anomaly scenarios in the network, but the administrator is the only who decides which anomaly is a real unwanted situation.

Hence he can decide when and what should be changed in the *PI* set. Certainly there could be automatic resolution processes that figure out how to solve the *PI* anomalies (inter- and intra-domain) by following some decision strategies chosen by the administrator. However resolving the anomalies automatically or not is out of scope of this work, even if it could be an interesting topic to be investigated as future work.

III. CASE STUDIES

In this section, we analyse two more policy domains: the communication protection and service function chain policies. For each of them, we describe an example of possible *PI* structure and a definition of detection rules that represent the

PI anomalies (both intra- and inter-domain). The first case study exploits an existing work in its domain presented in literature. As concerning the latter, a new definition of *PI* structure and detection rules is proposed to validate that the proposed model is able to support administrator-defined policy anomalies for a given domain.

A. Communication Protection Policy

Communication Protection Policies (CPPs) allow the definition of how to make the network communications secure in critical contexts, like financial or corporate. Several proposals have been presented in literature for detecting CPP conflicts, but most of them focus only on a single security technology. This is the reason why, we exploit the model proposed in [13], since the authors consider more than one security protocols.

Briefly, in this model a CPP pi has the following structure:

$$pi = (s, d, t, G, c^h, c^p, c^c)$$

where:

- s and d symbolize the source and destination nodes and their possible value could be either the network node id (Node) or an its IP address (IP), port number (Port) or URI (URI);
- t specifies the adopted technology, that could be IPsec, TLS, SSH, WS-Security or NULL;
- $G = (g_1, \ldots, g_n)$ is an ordered list of crossed gateway nodes, in the case of site-to-site or remote access communications;
- c^h, c^p and c^c characterize the policy actions and are three Boolean values that denote if the header integrity, payload integrity and confidentiality are required.

In addition, the main relations $\Re$ among network field and policy actions of CPPs can be summarized as follows:

- the source and destination values are designed in a hierarchical way, so that for a certain network node the following condition holds:

$$\text{Node} \succ \text{IP} \succ \text{Port} \succ \text{URI}$$

- a source (s) must assume different values from any other nodes in the network, independently from the type of value that s assumes (i.e., Node, IP, etc..). The same assumption must be done for the destinations (d) and it can be expressed as:

$$\{\text{Node, IP, Port, URI}\}_i \perp \{\text{Node, IP, Port, URI}\}_j$$

- the relations on the technology (t) values satisfy the following formulas:

$$\{\text{IPsec, SSH, TLS, WS-Security}\} \perp \text{NULL}$$
$$\text{IPsec} \succ \text{SSH} \succ \text{WS-Security}$$
$$\text{IPsec} \succ \text{TLS} \succ \text{WS-Security}$$
$$\text{SSH} \sim \text{TLS}$$

- header integrity, payload integrity and confidentiality flags (c^h, c^p and c^c) dominate the relative flags of another *PI*, it they assume True as value:

$$\text{True} \succ \text{False}$$

Two examples of the CPP detection rules defined in [13] are:

- the *Single-PI Irrelevance anomaly* occurs when pi_i can be removed without changing the network behaviour because it would establish a secure communication between a source and a destination that lay on the same node

$$s_i \not\perp d_i \Rightarrow \text{Irrelevant}(pi_i) \quad (1)$$

- the *Pair-PI Inclusion anomaly* occurs when two *PIs* are not equivalent and all the network fields of pi_i dominate or are equal to the ones of pi_j

$$s_i \succeq s_j \wedge d_i \succeq d_j \wedge t_i \succeq t_j \wedge c_i^h \succeq c_j^h \wedge c_i^p \succeq c_j^p \wedge$$
$$c_i^c \succeq c_j^c \wedge G_i \succeq G_j \wedge pi_i \neq pi_j \Rightarrow \text{Inclusion}(pi_i, pi_j)$$

- the *Pair-PI Alternative anomaly* occurs when all the fields of pi_i are equivalent to the fields of pi_j, but G_i is disjoint with G_j

$$s_i = s_j \wedge d_i = d_j \wedge t_i = t_j \wedge c_i^h = c_j^h \wedge c_i^p = c_j^p \wedge$$
$$c_i^c = c_j^c \wedge G_i \perp G_j \Rightarrow \text{Alternative}(pi_i, pi_j)$$

B. Service Function Chaining Policy

Service Function Chaining (SFC) defines an ordered set of network functions (e.g., NAT, load balancer, web cache, etc.) that processes the incoming traffic. Here forwarding policies can be used to configure the traffic flows that must be processed by a given chain.

In literature, most of the existing works focused on the OpenFlow protocol, which has been a successful SFC implementation. The detection of errors among OpenFlow rules deployed into the OpenFlow switches has been addressed in depth, while the correct forwarding policy definition at the SDN controller layer was overlooked.

In order to validate that the proposed model is able to support administrator-defined policy anomalies, we propose a new definition of *PI* structure and detection rules to detect forwarding policy anomalies at controller layer, in the Service Function Chaining (SFC) domain. The proposed *PI* is structured as follows:

$$pi = (ip\_src,\ ip\_dst,\ t,\ p\_src,\ p\_dst,\ C)$$

where $ip\_src, ip\_dst, t, p\_src, p\_dst$ have the same meaning of the network fields in the filtering *PI*, while the action C specifies the ordered set of network functions to which the incoming traffic should be forwarded. This means that also the relations $\Re$ between the network fields are similar to the filtering policy case, while a policy action C_i of pi_i can be equal, disjointed, correlated to or can dominate the C_j of pi_j.

Finally, we also propose an example of possible *PI* anomalies between SFCPs, which are represented by the following detection rules:

- *Intra-PI Incorrect anomaly* arises when the source and destination of a traffic flow correspond. This means that the ip_src and ip_dst fields of pi_i are equal and, hence, a forwarding loop is generated in the network

$$ip\_src_i = ip\_dst_i \Rightarrow \text{Incorrect}(pi_i)$$

- *Inter-PI Correlation anomaly* occurs if all network fields of pi_i and pi_j are equal, except C_i and C_j that are correlated. Hence there will be ambiguity on which chain should process the traffic:

$$ip\_src_i = ip\_src_j \wedge ip\_dst_i = ip\_dst_j \wedge$$
$$t_i = t_j \wedge p\_src_i = p\_src_j \wedge p\_dst_i = p\_dst_j \wedge$$
$$C_i \sim C_j \Rightarrow \text{Correlated}(pi_i, pi_j)$$

C. Examples of inter-domain policy anomalies

The detection of policy anomalies defined among policies of different domains (i.e., inter-domain anomaly) was generally overlooked by the existing works on the policy conflicts. However the inter-domain anomalies allow to achieve a higher flexibility in the definition of network errors, events and redundancies that administrators require to detect. In order to validate that our model is able to support this new feature, we can consider two examples of inter-domain policy anomalies.

Having presented the Filtering and Communication Protection Policies, a first example can be the *Pair-PI Filtered Anomaly*. In particular such anomaly identifies when a certain traffic flow, belonging to a secured communication, is discarded by a firewall because a rule is installed to deny ($a_j = \texttt{deny}$) such flow. This means that the secure communication (implemented by the CPP pi_i) is interrupted by the FP pi_j installed in the firewall. Hence, in the proposed model, the Pair-PI Filtered Anomaly can be expressed in this way:

$$s_i \not\perp ip\_src_j \wedge s_i \not\perp p\_src_j \wedge d_i \not\perp ip\_dst_j \wedge$$
$$d_i \not\perp p\_dst_j \wedge a_j = \texttt{deny} \Rightarrow \text{Filtered}(pi_i, pi_j)$$

Another example is the Inter-PI Interruption Anomaly that arises between the FP and SFC domains: let us consider a traffic flow (K) that should be processed by a service chain that contains a firewall ($C_i \succ \{f_j\}$), but a filtering policy was defined so that the flow K must be dropped by the firewall.

This means that the flow K will not traverse the entire service chain:

$$ip\_src_i = ip\_src_j \wedge ip\_dst_i = ip\_dst_j \wedge t_i = t_j \wedge$$
$$p\_src_i = p\_src_j \wedge p\_dst_i = p\_dst_j \wedge a_j = \{\texttt{deny}\} \wedge$$
$$C_i \succ \{f_j\} \wedge \Rightarrow \text{Interruption}(pi_i, pi_j)$$

IV. IMPLEMENTATION AND TESTING

The prototype implementation of the proposed model is based on an ontology. In our ontology-based prototype, the model was implemented using the OWL2 [14] language:

- a *class* was defined for each definition of *PI* structure and also for each network field and policy action of the *PI* (an *individual* element is created when a new *PI* is instantiated);

- the network field (or action) relations are represented by a set of *object property* assertions that connect the different classes of that network fields (or actions);
- a *PI* anomaly is established by defining the *object property* assertion that links, in turn, the *PI classes* involved into the anomaly;
- the detection rule that implements a *PI* anomaly is expressed in the SWRL [15] language that allows to specify them in a similar way of the Horn clauses.

An example of SWRL rule is shown below, where the Single-PI Irrelevance anomaly (equation 1) is defined:

```
hasSrc(?pi1,?s1), hasDst(?pi1,?d1),
NotDisjoint(?s1,?d1)−>Irrelevance(?pi1)
```

Finally we tested this prototype implementation in the context of Communication Protection Policy anomalies with a growing number of *PIs*. In addition, the *PIs* have been defined in such a way that 40% of the defined *PIs* present an anomaly with other *PIs*. Moreover, multiple tests have been performed to evaluate the elapsed time for detecting the CPP anomalies, considering different network scenarios: the tests have been run in networks composed of 100, 250 and 500 end-hosts.

As shown in Fig. 1, it is worth to note that in the tests upon the biggest network (that is the 500 end-hosts case), we achieve a reasonable detection time of no more than 45s.
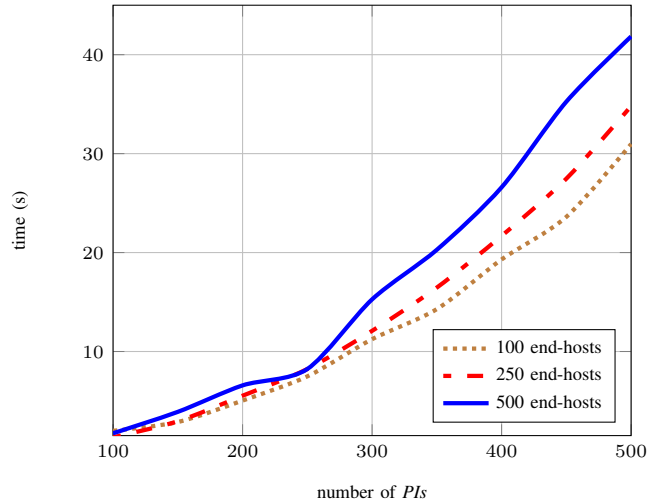

Fig. 1. Detection time vs. number of *PIs* in a CPP domain.

V. RELATED WORK

The network policies have been treated by many works, which analysed anomaly scenarios in different application domains. In this section our target is to examine the seminal works on the three domains of our case studies.

Concerning the filtering policy domain, the main work was presented by Al-Shaer *et al.* that focus on anomalies occurring in a distributed firewall [4]. In spite of our work, authors

treat anomalies between pair of rules, overlooking the case of anomalies generated by one or more than two rules. Moreover, they perform the anomaly detection by checking the packet header fields, without dealing with external information (e.g., algorithms, timestamps, etc...).

Khakpour *et al.* [16] have proposed a completely different approach to detect filtering anomalies based on a query engine system. However, this query-based approach does not find all the anomalies, as it checks only the scenarios specified by the administrator. In this approach, the impact of the human factor is very significant in the selection of the correct set of queries: an anomaly can be overlooked because it was not selected.

Other works, which take Al-Shaer *et al.* works as reference, were presented in literature. Basile *et al.* [17], [18] extend the Al-Shaer's conflict classification and propose a formal policy specification model. The proposed model is based on a geometrical representation: the set of all packet header fields (namely the "selectors") generate an hyper-space, where the policy rules can be represented. Here, an anomaly is seen as the intersection of the space of two or more rules. Even though the authors fill the limitations of the Al-Shaer's work, they do not represent the anomalies generated by a single rule.

On the other hand, Garcia-Alfaro *et al.* [19], [20] detect and resolve filtering anomalies thought the use of specific algorithms in distributed system, where NIDSs, VPN routers and other security controls are considered. However this is a radical different approach from our model, since rule-based inferential systems (formal ontologies, rule-based systems, SMT solver, etc...) do not use algorithms to detect anomalies.

Concerning the communication protection policy, a greater emphasis was given only to IPsec configuration anomaly, starting form the common idea, initially introduced by Zao in [21], to combine conditions that belong to different IPsec fields. An early work has been presented by Fu *et al.* [3], which verify the correctness of the deployed policies comparing them with high-level policies. In this way the authors give another interpretation of the anomaly concept. In fact, in our vision, an anomaly can be seen as an erroneous interaction among policies, while Fu *et al.* support that an anomaly derives from an incorrect deployment of the policy.

Al-Shaer *et al.* [22] formalizes the classification of [3] by proposing a model based on Ordered Binary Decision Diagrams. The work can be seen as the extension of the packet filter classification proposed by Al-Shaer *et al.* [4]. In fact, this model incorporates both the encryption capabilities of IPsec and its packet filter capabilities, but it does not fill the limitations of the previous work [4].

Finally, the last case of network policy domain is the Service Function Chaining (SFC), where the use of forwarding policy is sustained by the active IETF working group [8]. Also the need for a forwarding rule verification is suggested in [23] in order to avoid that an incoming traffic could match more traffic classification rules of different SFCs.

In this field, several studies addressed the detection of possible anomalies between the most common SDN protocol, that is OpenFlow [5]. The existing works analyse the anomalies among the OpenFlow rules installed in each switch of the network. In particular several techniques to detect the anomalies have been used by those works, such us Binary Decision Diagrams by Al-Shaer *et al.* [24], hash-tree data structure in the work of Natarajan *et al.* [25], and First Order Logic by Batista *et al.* [26].

In our vision, the main limitation of these works is the absence of interaction with other policy domains. In fact the traffic forwarding does not depend only by the switch and router configurations, but also it can be influenced by other network functions (e.g., firewall, load balancer, VPN gateway, etc...). Hence, an inter-domain anomaly analysis should be more suited to verify the correctness of the network behaviour, as well as it should be encouraged for all the other policy domain analysis.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we propose a formal model for detecting network policy anomalies, both in case of inter- and intra-domain. This policy model allows the analysis of policy anomalies by using a formal representation for the network policy, named policy implementation (*PI*) and for their anomalies.

The generality of the solution was also validated by applying the model to three case studies of different domains: communication protection, data filtering and service function chaining. Finally we implemented a prototype, using ontological techniques. The prototype was also tested in order to evaluate the impact of a *PI* anomaly detection process in different sized networks, obtaining good results.

As possible future work, we plan to extend the expressivity and capabilities of our model to make it able to solve problems like policy reachability and policy reconciliation. This will require a deeper knowledge of the network state to ensure that the best policies are chosen and to reconcile policy conflicts.

REFERENCES

[1] A. Wool, "Firewall Configuration Errors Revisited," *CoRR*, vol. abs/0911.1240, pp. 103–122, 2009.

[2] J. Strassner, *Policy-Based Network Management: Solutions for the Next Generation (The Morgan Kaufmann Series in Networking)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[3] Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu, "IPsec/VPN security policy: Correctness, conflict detection, and resolution," in *International Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, January 15–17 1999, pp. 39–56.

[4] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 10, pp. 2069–2084, October 2005.

[5] R. Bifulco and F. Schneider, "Openflow rules interactions: definition and detection," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. IEEE, 2013, pp. 1–6.

[6] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[7] R. Soulé, S. Basu, R. Kleinberg, E. G. Sirer, and N. Foster, "Managing the network with merlin," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*. ACM, 2013, p. 24.

[8] S. M. Boucadair, D. Lopez, I. Telefonica, D. J. Guichard, and C. Pignataro, "Service function chaining: Framework & architecture draft-boucadair-sfc-framework-00," Tech. Rep., 2013.

[9] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *Communications Magazine, IEEE*, vol. 51, no. 11, pp. 24–31, 2013.

[10] P. Skoldstrom, B. Sonkoly, A. Gulyás, F. Németh, M. Kind, F.-J. Westphal, W. John, J. Garay, E. Jacob, D. Jocha *et al.*, "Towards unified programmability of cloud and carrier infrastructure," in *Software Defined Networks (EWSDN), 2014 Third European Workshop on*. IEEE, 2014, pp. 55–60.

[11] R.Szabó, B. Sonkoly, and M.Kind, "Unify d2.2: Final architecture," 2015. [Online]. Available: https://www.fp7-unify.eu/index.php/results.html#Deliverables

[12] R.Steinert and W. John, "Unify d4.2: Proposal for sp-devops network capabilities and tools," 2015. [Online]. Available: https://www.fp7-unify.eu/index.php/results.html#Deliverables

[13] C. Basile, D. Canavese, A. Lioy, and F. Valenza, "Inter-technology conflict analysis for communication protection policies," in *CRiSIS 2014 9th International Conference on Risks and Security of Internet and Systems*, Trento, Italy, August 6–8 2014.

[14] W. O. W. Group, "OWL 2 web ontology language document overview," Tech. Rep., Oct. 2009, http://www.w3.org/TR/2009/REC-owl2-overview-20091027/.

[15] "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," World Wide Web Consortium, Tech. Rep., May 2004.

[16] A. R. Khakpour and A. X. Liu, "Quantifying and querying network reachability," in *IEEE Int. Conference on Distributed Computing Systems*, Genova (Italy), June 21–25 2010, pp. 817–826.

[17] C. Basile, A. Cappadonia, and A. Lioy, "Network-Level Access Control Policy Analysis and Transformation," *IEEE/ACM Transactions on Networking*, vol. 20, no. 4, pp. 985–998, August 2012.

[18] C. Basile and A. Lioy, "Analysis of application-layer filtering policies with application to http," *Networking, IEEE/ACM Transactions on*, vol. 23, no. 1, pp. 28–41, Feb 2015.

[19] J. G. Alfaro, N. Boulahia-Cuppens, and F. Cuppens, "Complete analysis of configuration rules to guarantee reliable network security policies," *Int. J. of Information Security*, vol. 7, no. 2, pp. 103–122, April 2008.

[20] J. Garcia-Alfaro, F. Cuppens, N. Boulahia, and P. Stere, "MIRAGE: a management tool for the analysis and deployment of network security policies," in *SETOP 2010: 3rd International Workshop*, Athens, Greece, September 23 2011, pp. 203–215.

[21] J. Zao, "Semantic model for IPsec policy interaction," Internet Draft, Tech. Rep., March 2000.

[22] E. Al-Shaer, H. Hamed, and W. Marrero, "Modeling and Verification of IPSec and VPN Security Policies," in *13th IEEE Int. Conference on Network Protocols*, Boston, MA, November 6–9 2005, pp. 259–278.

[23] S. Lee and M. Shin, "Service function chaining verification draft-lee-sfc-verification-00," Tech. Rep., February 2014.

[24] E. Al-Shaer and S. Al-Haj, "Flowchecker: Configuration analysis and verification of federated openflow infrastructures," in *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*. ACM, 2010, pp. 37–44.

[25] S. Natarajan, X. Huang, and T. Wolf, "Efficient conflict detection in flow-based virtualized networks," in *Computing, Networking and Communications (ICNC), 2012 International Conference on*. IEEE, 2012, pp. 690–696.

[26] B. L. A. Batista, G. A. L. de Campos, and M. P. Fernandez, "Flow-based conflict detection in openflow networks using first-order logic," in *Computers and Communication (ISCC), 2014 IEEE Symposium on*. IEEE, 2014, pp. 1–6.