Classification and analysis of communication protection policy anomalies

Classification and Analysis of Communication Protection Policy Anomalies

Fulvio Valenza, Cataldo Basile, Daniele Canavese and Antonio Lioy

***Abstract*—This paper presents a classification of the anomalies that can appear when designing or implementing communication protection policies. Together with the already known intra- and inter-policy anomaly types, we introduce a novel category, the inter-technology anomalies, related to security controls implementing different technologies, both within the same network node and among different network nodes. Through an empirical assessment, we prove the practical significance of detecting this new anomaly class. Furthermore, this paper introduces a formal model, based on first-order logic rules that analyses the network topology and the security controls at each node to identify the detected anomalies and suggest the strategies to resolve them. This formal model has manageable computational complexity and its implementation has shown excellent performance and good scalability.**

***Index Terms*—protection policy, policy anomalies, policy conflicts, network security.**

I. Introduction

Enforcing network and communication security in a computer system is a very complex and sensitive task. Security administrators have a hard job to accomplish since it requires very specific skills and a high level of competence. On one hand, several studies confirm the administrators' responsibilities in many security breaches and breakdowns. Wool showed that most of the analysed firewalls contain several problematic policies, such as very lax rules [1]. The 2015 Data Breach Investigations Report states that about 60% of the security breaches are due to internal staff errors [2]. On other hand, administrators often have basic or no tool support to debug the security controls' configurations and check if the enforced policy is compliant with the high-level security requirements.

Configuration analysis methods, often addressed in literature, have seldom been incorporated into industrial tools for various reasons. For instance, some analysis methods may be challenging to use because their computational complexity make them inapplicable or some others make assumptions that are far from reality. The only notable exception is the detection of packet filter anomalies classified by Al-Shaer [3], which is available in some Cisco routers [4].

The main contribution of this paper is a formal model to assist security administrators in configuring and validating security policies. We focus on *communication protection policies* (CPPs), which determine how to protect assets (such as private user data and corporate intellectual properties) when they are transferred over computer networks. CPPs are very sensitive, as incorrect implementations can result in information disclosure that can cause violations of the users' privacy, intellectual properties and, often, huge monetary losses. Moreover, CPP are particularly difficult to manage and enforce since the probability of introducing them increases with the number of entities involved in the implementation, as well as with the size and complexity of the network to configure [5], [1]. Therefore, there is a non-negligible risk of mistakes, such as introducing contradicting configurations and redundant channels.

CPPs originate from legal (e.g. the EU privacy law [6]) and business security requirements. They are often expressed with high-level directives specifying the security properties that the communication must guarantee (e.g. confidentiality and integrity). In some cases, such as companies that host services or provide cloud-based resources, CPPs may be very elaborated and complex. IT managers generally interact with business units to determine what to protect. However, CPPs are enforced by a plethora of security controls implementing different protocols (e.g. IPsec, TLS, WS-Security) that operate at different layers of the OSI stack. Service administrators have to enable channel protection protocols on the services they manage. Finally, *network and security administrators* have to decide the resources accessible through secure protocols and the data that need protection. Consequently, they have to map the selected resources and data to network entities and traffic to protect, decide the OSI layer where to apply the protection, select whether to create end-to-end channels or tunnels, when to enable wireless protection, and so on.

The approach we follow in this paper is to detect and show to the administrators the *anomalies*, which are the presence of redundant or conflicting configuration rules. Anomalies often reveal human errors and thus deserve the explicit attention of the administrators. Detecting anomalies helps in daily CPPs' management activities. On one hand, when designing a CPP implementation, debugging the policy before implementing it can speed-up the deployment time and avoid critical problems (*CPP design validation*). With our approach, the administrator has to provide in input the communications he wants to protect, the security properties to be enforced and the technology he wants to use. The model we developed will output the detected anomalies, such as redundant, insecure, non-enforceable and filtered communications, and propose resolutions that significantly help in reducing human errors. On the other hand, our approach can help in assuring that the current CPP implementation respects its intended semantic. In the

This work has been partly supported by the SECURED and SHIELD project (grant agreements no. 611458 and 700199), co-funded by the European Commission. The authors gratefully thank Dr. Marco Torchiano and Dr. Gabriella Dardanoni for their feedback on the empirical study.

F. Valenza, C. Basile, D. Canavese, and A. Lioy are with the Politecnico di Torino, Dip. Automatica e Informatica, Torino (Italy). Email: {first.last}@polito.it, F. Valenza is also with the CNR-IEIIT, Torino (Italy); e-mail: fulvio.valenza@ieiit.cnr.it.

CPP implementation analysis, the administrator provides the network topology and the configuration of all the protection controls to get as output the anomalies, explanations, and suggested remediation actions.

This paper is structured as follows. Section II lists our contributions to the current state-of-the art. Sections III and IV informally introduce our approach by presenting some background and a motivating example. Sections V and VI are the core of this paper and describe the formal structures and the formulas of our model. Section VII presents the graphical notations for reporting the anomalies. Section VIII contains the complexity and performance analysis of the presented approach, together with an empirical study. Finally, Sections IX and X contain the related works and the conclusions.

II. Contributions

Our work pushes forward the state-of-the-art in several directions. The main contribution is the identification of nineteen types of anomalies that may happen when implementing a CPP. Six anomaly types are already known [5], but all the others are our original contribution[1].

The anomalies we identify arise in the configuration of a single security control (*intra-policy*), between controls of the same type displaced at different network nodes (*inter-policy*), and, our novel contribution, among security controls implementing different technologies, within a single network node or among different network nodes (*inter-technology*). We focus on communication protection controls that work at four network layers: data link, network, session[2], and application. As an example, inter-policy anomalies may appear between the configurations of two IPsec gateways, while inter-technology anomalies may arise between the IPsec and TLS configurations implemented at the same network node. Moreover, we also identify communications that are intrinsically insecure or non-enforceable by the target security controls.

Anomalies are detected by means of a formal model that takes as input the network description, nodes information and the security controls' configurations and the communications to secure, during a CPP design validation, or the communications actually secured, during a CPP implementation analysis. Information in input become part of a knowledge base explored with a set of first-order logic (FOL) formulas to identify and report the detected anomalies to the administrators.

Anomalies have been categorized according to two different classifications: an *effect-based* taxonomy and a *information-centric* one. The first classification divides the anomalies into five macro-categories describing the effects that they have on the network: 1) insecure communications; 2) unfeasible communications; 3) potential errors; 4) suboptimal communications; 5) suboptimal walks. The second one is based on the information needed to be analysed for detecting the anomalies. It divides the anomalies into three classes: 1) anomalies in a single communication channel; 2) anomalies between secure channels that start at or end on the same node; 3) anomalies that are only evident if the full network information (nodes and topology) and high-level security requirements are considered.

Having introduced several new kinds of anomaly, we posed ourselves several questions regarding the impact of our work:

1) is detecting these anomalies important and helpful to improve the security of the current IT infrastructures?
2) are these anomalies actually introduced by the administrators when they implement their policies?
3) is it computationally feasible to identify these anomalies in large networks?

In order to answer the first question, for each anomaly we present the possible consequences on the network and some ways to resolve it for reducing the security impacts on the short and long period (see Section VI). To answer the second question, we prepared an empirical experiment where three categories of administrators (experts, intermediate and beginners) were asked to configure a set of CPPs in a sample network. We noticed that several of the newly introduced anomalies appeared (see Section VIII-A). And finally, to answer the last question, we implemented, and tested in several different scenarios, a tool making use of DL (description logic) ontologies and custom Java-based reasoning rules (see Section VIII-C).

III. Background

A *communication* is any directional data exchange between two network entities. A *secure communication* is a communication 'adequately' protected, that is it fully satisfies a set of security requirements. In this context, the security requirements concern three *security properties*: header integrity, payload integrity and (payload) confidentiality.

A *channel* is a directional data exchange between two nodes protected with some security properties (a *secure* channel) or none (an *insecure* channel). Logically, a secure channel is an association between a source, where the security properties are applied, and a destination, where the security properties are removed or verified. A communication can be thought as a stack of several (secure and/or insecure) channels. For example, an end-to-end TLS communication consists of a single secure channel. However, more complex scenarios exist. For instance, a communication between two hosts in separate networks connected via an IPsec site-to-site VPN is modelled with two channels: an insecure one between the end-points and an IPsec secure channel between the VPN terminators.

In the real world, the secure communications are defined by using a set of configuration settings containing several low-level details. For instance, the configuration of a TLS server contains detailed information about the supported ciphersuites. However, during the design and policy analysis phases, this level of granularity is usually not needed. For our purposes, a secure channel can be represented by specifying: 1) the source and destination entities (they can be network nodes or direct references to an entity lying at a particular OSI

[1]We presented an embryonic and incomplete set of anomalies in a previous work [7]. Note that the "superfluous" keyword was already used in another work about firewall conflicts to denote redundant and shadowed rules. It is a completely different meaning, as it will be evident from this treatment [8].

[2]Protections at transport layer, such as TLS, are sometimes associated to the session layer as they work on top of the TCP/UDP protocols. We do not want to enter a philosophical diatribe as, for our purposes, the important thing is the order of encapsulation of the different protections.

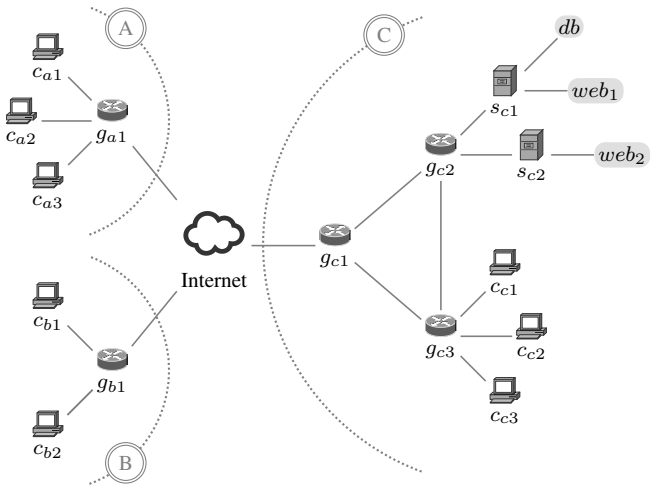


Fig. 1: A simplified network scenario.

layer such IP addresses and URIs); 2) the security protocol to use (our model can be easily extended to new protocols and can support a wide array of technologies at different OSI layers); 3) the required security properties; 4) the crossed gateways and the traffic to protect (meaningful only in case of tunnels). We name *policy implementation* (or *PI* for short) this formal representation of a channel. Note that since a channel is directional, a PI is directional too. This means that, to create a complete request-reply connection, we need at least two PIs. More information on this subject is provided in Section V.

We call a *PI set* a group of policy implementations that belongs to the same node and use the same technology. For instance, a particular server supporting IPsec and SSH will have two PI sets, one for each protocol. We will assume without loss of generality that the policy implementations in the same PI set are ordered according to their priority[3]. Note that our analysis uses other additional sources of information:

- *network reachability data*, as the configurations of filtering controls and NAT devices must be available to determine if the channels can be actually established (e.g. to check if a channel is not dropped by a firewall);
- *supported security protocols* (at various OSI levels), for guaranteeing that it is possible to establish the secure channels;
- *supported cryptographic algorithms*, as some ciphersuites might not be available when actually deploying a PI on the installed security controls.

Finally, we will refer to the *network topology* as a graph where its nodes are potential channel end-points (both sources and destinations) and its edges are physical or virtual connections between them.

IV. Motivating example

Before formally tackling the analysis of the PI anomalies, we begin our discussion by considering the simplified network scenario in Fig. 1. The diagram shows a main corporate network (C) and two branch networks (A and B). The three networks are connected via the Internet and consist of a number of security gateways (denoted by g) that mediate the communications between the servers (s_{c1} and s_{c2}) and the clients (indicated by c). The server s_{c1} hosts two services (web_1 and db), while s_{c2} hosts only one web service (web_2).

We will use the informal notation $s \xrightarrow[pi,c]{\text{t}} d$ to indicate a PI that establishes a channel from the source s to the destination d using the technology t to enforce some security properties. In this simplified example we will only take into account two security properties, (payload) confidentiality and payload integrity, denoted by the symbols c and pi, respectively. For instance, the PI $a \xrightarrow[pi]{\text{IPsec}} b$ indicates an IPsec connection with integrity (but not confidentiality), from a to b.

For the sake of clarity, we grouped the anomalies into five macro-categories, as shown in Fig. 2. These macro categories will be briefly described in the next paragraphs.

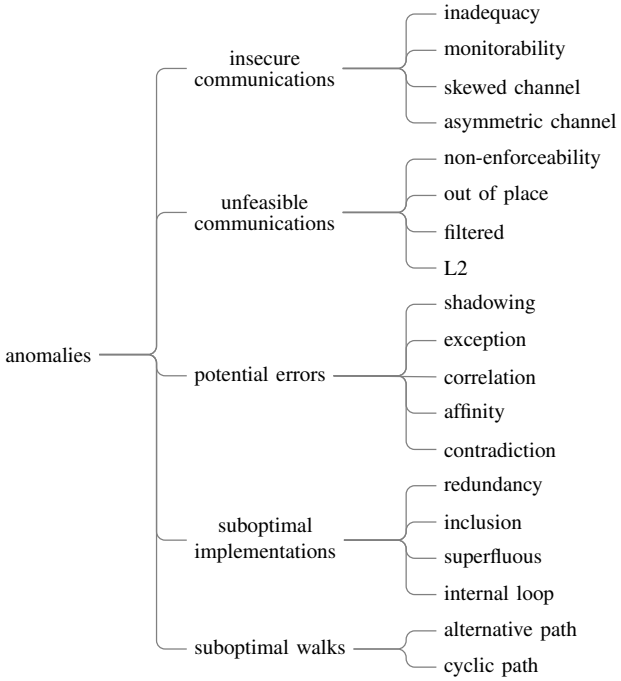


Fig. 2: Effect-based taxonomy.

A. Insecure communications

We have an *insecure communication* when its security level is less than the expected one. For instance, a channel that does not satisfy the minimum security level specified in the corporate policy generates an *inadequacy* anomaly. We have this anomaly if the IT managers require that 'all the data crossing the Internet must be encrypted' and a security administrator creates the policy implementation $c_{a1} \xrightarrow[pi]{\text{TLS}} s_{c1}$.

Another case of insecure communication arises when the security requirements are respected but we have a communication consisting of more than one channel (e.g. remote-access). In this case, the nodes at the channel junctions can 'see' the exchanged data, thus lowering the security of the connection

[3]The work [9] proved that any policy represented as a set of rules can be expressed as an equivalent policy where rules are ordered by their priority.

and creating a *monitorability* anomaly. For instance, the PIs $s_{c1} \xrightarrow[c,pi]{\text{IPsec}} g_{c1}$ and $g_{c1} \xrightarrow[c,pi]{\text{IPsec}} c_{a1}$ create a form of logical communication between s_{c1} and c_{a1} composed of a sequence of two channels interconnected through g_{c1}. This means that, even if everything is encrypted, g_{c1} reads the payload because it decrypts and encrypts the exchanged data (note that this may be the intended behaviour).

Another kind of insecure communication, more subtle but potentially catastrophic, can occur with a wrong tunnel overlapping that removes the confidentiality in a part of the communication and produces a *skewed channel* anomaly (a super-set of the Hamed et al.'s overlapping anomalies [5]). For example, a security administrator can create a tunnel $g_{c3} \xrightarrow[c]{\text{IPsec}} g_{a1}$ and another one with $g_{c3} \xrightarrow[c]{\text{IPsec}} g_{c1}$ (note that the latter tunnel is 'included' in the first one). The trellis diagram in Fig. 3 helps to graphically visualize the problem.

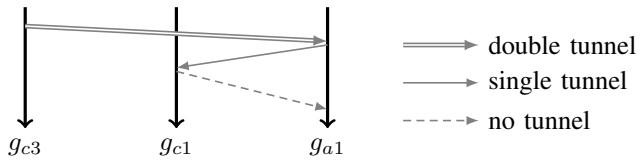


Fig. 3: Diagram of the skewed channel between g_{c3}, g_{c1}, g_{a1}.

When g_{c3} sends some data, it encapsulates the information in two tunnels. Hence when g_{a1} receives the data, it removes the external tunnel encapsulation, but cannot remove the internal one, so g_{a1} sends the data back to g_{c1} which, in turn, removes the last tunnel. Finally, g_{c1} sends the data to g_{a1} with no protection, thus exposing the communication content to a sniffing attacker.

In the real world, most of the connections are bidirectional, since a request usually requires a reply. It may be the case that the request channel has a different security level from the reply one, generating an *asymmetric channel* anomaly. This is not necessarily an issue, but it could be useful to report this inconsistency, so that administrators can check if the security control configurations reflect the intended network behaviour.

B. Unfeasible communications

An *unfeasible communication* is a communication that cannot be established due to a hard misconfiguration. These anomalies are very severe since they completely prevent any data exchange. The simplest example of an unfeasible communication is when the security administrators design a PI with a technology not supported by an end-point or a security level too high to be enforced by the available cipher-suites. We call this situation a *non-enforceability* anomaly. For instance, the policy implementation $web_2 \xrightarrow[c]{\text{TLS}} db$ becomes non-enforceable if the service administrators did not install a TLS module on s_{c2} (where web_2 resides). This PI must obviously be deployed on the source endpoint s_{c2}, however, if the node containing this PI is not s_{c2}, we have generated another problem, an *out of place* anomaly.

We have also a hindered connection if the packets of a channel are dropped by a firewall that lies on the path between the source and destination, thus producing a *filtered channel* anomaly.

Firewalls and bad server configurations are not the only causes of an unfeasible communication. There are also technological incompatibilities between wired and wireless protocols when performing security at level 2 (data link) of the OSI stack. For example, if we choose to create a secure channel using the WPA2 technology, we must be sure that the network frames only cross wireless-enabled nodes. If one or more crossed devices are wired-only, then we have an *L2 anomaly*.

C. Potential errors

Potential errors are a class of anomalies where the original intent of the administrators is unclear. Hence their resolution requires a full human inspection. When working with a large group of PIs, an administrator can create a PI that meddles all the traffic of another one that has different security properties. For instance, the PI $c_{a1} \xrightarrow[pi]{\text{TLS}} web_1$ hides $c_{a1} \xrightarrow[c]{\text{TLS}} web_1$, if the first one has a higher priority. Since the first one shadows the second one we call this anomaly a *shadowing* anomaly. If the second PI instead has a higher priority, we have an *exception* anomaly. Exceptions are useful and are typically exploited by administrators to express an 'all but one' rule, but we report them for verification.

Another kind of potential error is when we have two PIs with the same technology and with the source and destination on the same node. This situation can lead to an ambiguity, since sometimes a piece of data can match multiple PIs, hence making the intended protection level unclear. For example, $web_2 \xrightarrow[c]{\text{TLS}} s_{c1}$ and $s_{c2} \xrightarrow[c,pi]{\text{TLS}} db$ are ambiguous since a packet from web_2 to db can match both PIs. We call this problem a *correlation* anomaly. Analogously, we will have an *affinity* anomaly between two PIs that use different technologies but have the source and destination on the same nodes.

Finally, we have a *contradiction* anomaly when two PIs respectively express that the same communication should be protected and not protected. For instance, let suppose that an IT manager defines a policy where 'all the traffic for the Internet must be inspected' and a security manager enforces the encryption of the traffic exchanged between c_{a1} and s_{c1} via the PI $c_{a1} \xrightarrow[c,pi]{\text{IPsec}} s_{c1}$. This leads to a contradiction, since the policy requires that the data for the Internet should be monitorable, but the PI encrypts them.

D. Suboptimal implementations

Suboptimal implementations arise when one or more PIs can decrease the network throughput by producing some overhead in the nodes. Their existence is usually not problematic, but their resolution can be beneficial since it improves the network performance and makes the PIs less vulnerable to DoS attacks. The simplest kind of suboptimal implementation occurs when an administrator deploys a PI that makes another one useless, as the first one can secure the communication at the same or a higher level of the second, with a more effective protection (e.g. stronger encryption). For example,

two different security administrators may have independently defined the PIs $c_{a1} \xrightarrow[c]{\text{TLS}} web_1$ and $c_{a1} \xrightarrow[c,pi]{\text{IPsec}} s_{c1}$. The former is included in the latter, so that it can be safely removed. In these cases we have a *redundancy* anomaly (if both the PIs use the same technology) or an *inclusion* anomaly (if they use two different protocols).

Another type of suboptimality arises when a tunnel encapsulates other tunnels with a higher security level. This is a *superfluous* anomaly and can be resolved by simply deleting the external, redundant tunnel.

We can also have some channels that can be safely removed without altering the network semantic. This happens in the so called *internal loop* anomalies, where a PI source and destination belong to the same node.

E. Suboptimal walks

A group of PIs can produce a *suboptimal walk* when the path taken by the data is unnecessarily long.

In large networks, a communication between two endpoints can take multiple paths, thus generating an *alternative path* anomaly. This is not necessarily a misconfiguration, but nonetheless we detect and report it to the administrators to stay on the safe side. For example, $g_{c2} \xrightarrow[c]{\text{IPsec}} g_{c3}$ forms an alternative path w.r.t. $g_{c2} \xrightarrow[(c}{\text{IPsec}} g_{c1}$ and $g_{c1} \xrightarrow[c]{\text{IPsec}} g_{c3}$.

Another cause of suboptimality occurs when some data cross a node multiple times during their travel. This *cyclic path* anomaly can be removed by deleting the cycles, thus shortening the network path.

V. PI HIERARCHICAL STRUCTURE

In this section, we formally define what is a policy implementation, its structure, and the relationships between the various network fields that compose it. In addition, we describe the notion of path used to detect several kinds of network anomalies. In our model, a PI i is a tuple:

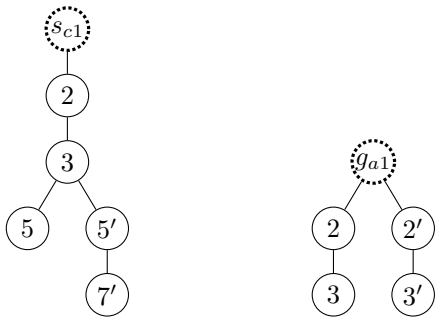
$$i = (s, d, t, C, S, G)$$

where:

- s and d respectively represent the channel source and destination (Section V-A);
- t is the adopted security technology (Section V-B);
- C is an ordered set of coefficients that indicate the required security levels (Section V-C);
- S is a selector, i.e. a tuple of network fields, used to identify the traffic that need to be protected (Section V-D);
- G is the list of the gateways involved in the communication (Section V-E).

A. Sources (s) and destinations (d)

To perform an accurate detection of anomalies, we need to identify the OSI layer where a communication starts and terminates. To this purpose, we use a hierarchical structure that represents the points where the secure communication endpoints can be established. This structure has a very simple tree-like graphical representation as shown in Fig. 4.



(a) Representation of s_{c1}. (b) Representation of g_{a1}.

Fig. 4: Graphical representation of a server and a gateway.

The root represents the network node itself, while all the tree nodes model the available connection end-points, named *network entities*, ordered according to the OSI stack layer (see Fig. 4a). We only focus our attention on the data link, network, session and application layers. The tree levels may also be associated to layer 2 addresses, IP addresses, port numbers and URIs. To avoid ambiguity we will use the notation $s_{c1}.l5'$ to specify the node labelled $5'$ in the s_{c1} tree and so on.

Note that the gateways expose multiple interfaces, one for each network where they are connected to. For instance, in Fig. 4b, the two layer 3 vertices represent the 'internal' interface (network A) and an 'external' interface (the Internet). If a gateway also supports VPNs via TLS tunnels (e.g. OpenVPN), two additional vertices are present at the session level.

Given any two network entities e_1 and e_2, we define the following relationships:

- e_1 is *equivalent* to e_2 ($e_1 = e_2$) if they are exactly the same entity;
- e_1 *dominates* e_2 ($e_1 \succ e_2$) if all the traffic starting from (or arriving to) e_2 passes through e_1. On the graph, e_1 is an ancestor of e_2 in the tree representation. This concept is particularly useful when dealing with security protocols working at different OSI layers. For instance in Fig. 4a, $s_{c1}.l3$ dominates $s_{c1}.l7'$;
- e_1 is a *kin* of e_2 ($e_1 \sim e_2$) if e_1 and e_2 belong to the same network node, but there is no equivalence or dominance relationship amongst them. For example in Fig. 4a, $s_{c1}.l5$ is a kin of $s_{c1}.l5'$;
- e_1 and e_2 are *disjoint* ($e_1 \perp e_2$) if they belong to different network nodes (and hence trees).

Note that if e_1 and e_2 are not disjoint ($e_1 \not\perp e_2$) that means that they are on the same device, hence they are related by an equivalence, dominance or kinship relationship.

B. Technologies (t)

In this paper we take into account a limited set of technologies, but our model is flexible enough to accommodate any security protocol. In particular we will consider only:

- for the data link layer: WPA2 and 802.1AE MACsec;
- for the network layer: IPsec;
- for the session layer: TLS and SSH;
- for the application layer: WS-Security.

In addition we also use the special NULL technology, indicating that a communication should be created without any kind of protection.

Similar to the network entities, two technologies t_1 and t_2 can have different relationships:

- t_1 is *equivalent* to t_2 ($t_1 = t_2$), if they are exactly the same technology;
- t_1 *dominates* t_2 ($t_1 \succ t_2$) if t_1 operates at an OSI level strictly less than the t_2's one. By definition, the NULL technology is dominated by all the other technologies;
- t_1 is a *kin* of t_2 ($t_1 \sim t_2$) if t_1 and t_2 are different and work at the same OSI layer;
- t_1 is *disjoint* from t_2 ($t_1 \perp t_2$) if one technology is NULL and the other one is not NULL.

In general, the following relationships hold:

$$t^{(i)} \sim t'^{(i)}, \quad t^{(i)} \neq t'^{(i)}$$

$$t^{(2)} \succ t^{(3)} \succ t^{(5)} \succ t^{(7)}$$

$$t \perp \text{NULL}, \quad \forall t \neq \text{NULL}$$

Where $t^{(i)}$ represent a technology at the OSI level i.

C. Security coefficients (C)

The tuple of security coefficients consists of several non-negative real values that indicate a required security level for a specific property. The higher a value the stronger the enforcement of a property should be. On the other hand, if a coefficient is zero the related security property must not be enforced. Obviously if the chosen technology is NULL, all the coefficients are zero. These values should be estimated by the administrators with the use of some metrics, for example on the chosen cipher-suite (e.g. taking into account the key length, encryption/hash algorithms and cipher mode).

In this paper we focus our attention only on three properties, which are header integrity (c^{hi}), payload integrity (c^{pi}) and (payload) confidentiality (c^{c}), so that:

$$C = (c^{hi}, c^{pi}, c^{c})$$

The relationships amongst two coefficient sets C_1 and C_2 are:

- C_1 is *equivalent* to C_2 ($C_1 = C_2$) if all the coefficients of C_1 are the same as their C_2's counterparts;
- C_1 *dominates* C_2 ($C_1 \succ C_2$) if at least one coefficient of C_1 is strictly greater than its C_2's counterparts and the other coefficients of C_1 are not less than their C_2's counterparts;
- C_1 is *disjoint* with C_2 ($C_1 \perp C_2$) if there is neither dominance nor equivalence between C_1 and C_2, that is $C_1 \not\succeq C_2 \wedge C_1 \not\preceq C_2$.

D. Selectors (S)

Some security protocols (e.g. IPsec, see RFC-3585) allow the definition of filtering conditions to select the traffic that must be protected. Our model supports such conditions via the *selectors* S of a policy implementation, that are tuples of network fields. In theory (and in our model too), S can be arbitrarily defined with any field, however, in practice, the fields in S are usually the well-known five tuple consisting of a source IP address (ip_{src}) and port (p_{src}), a destination IP address (ip_{dst}) and port (p_{dst}) and a protocol type (prt). We will assume this in the rest of the paper that S at least includes the five-tuple, that is:

$$S = (ip_{src}, p_{src}, ip_{dst}, p_{dst}, prt, \dots)$$

We will use the notation $\overleftarrow{S}$ to indicate a *reverse* list of selectors where source and destination are swapped, that is $\overleftarrow{S} = (ip_{dst}, p_{dst}, ip_{src}, p_{src}, prt, \dots)$. In addition, we use the notation $S|_{f_1 \times f_2 \times \dots}$ to restrict the selector space to the fields f_1, f_2, ... For instance, in the following sections, we will often use the more compact $S|_{ip_{src} \times p_{src}} = (ip_{src}, p_{src})$ instead of the n-tuple $(ip_{src}, *, ip_{dst}, *, *, \dots)$ (where the asterisk symbol $*$ will denote a field matched by any value) to define a selector that matches all the traffic from ip_{src} to ip_{dst} regardless of the port numbers and protocol. Moreover, the all-matching tuple $(*, *, *, *, *, \dots)$ will be also shortened to a single $*$ inside a PI definition.

In addition, we will make use of the following relationships between the selector tuples:

- S_1 is *equivalent* to S_2 ($S_1 = S_2$), if their selectors are exactly the same;
- S_1 *dominates* S_2 ($S_1 \succ S_2$) if the matched traffic of S_2 is a sub-set of the matched traffic of S_1;
- S_1 is a *kin* to S_2 ($S_1 \sim S_2$) if there is at least one communication that matches S_1 but not S_2 and vice-versa;
- S_1 is *disjoint* from S_2 ($S_1 \perp S_2$) if the sets of the traffic matched by S_1 and S_2 are disjoint.

E. Crossed gateways (G)

Tunnel PIs contain an ordered set G that specifies the gateways crossed by the channel traffic. The G sets of tunnel PIs is statically computed from the network topology and the content of the routing tables.

Note that the list of crossed gateways does not contain the channel source and destination nodes. We use the notation G^* to indicate a list containing also the PI end-points, that is $G^* = \{s\} \cup G \cup \{d\}$. We will also denote the list of crossed gateways in reverse order with $\overleftarrow{G}$.

It is worth presenting an example of PIs that use gateways. Given the network in Fig. 1, a communication from c_{a1} to s_{c1} that passes into an IPsec tunnel between the two gateways g_{a1} and g_{c2} is implemented by two PIs:

$$i_1 = (c_{a1}, s_{c1}, \text{NULL}, (0,0,0), *, (g_{a1}, g_{c1}, g_{c2}))$$

$$i_2 = (g_{a1}, g_{c2}, \text{IPsec}, (3,3,3), (ip_{c_{a1}}, *, ip_{s_{c1}}, *, *), (g_{c1}))$$

where i_1 specifies the communication that will be encapsulated in the tunnel defined by i_2.

F. Paths

We introduce now the concept of path, which completes the notions used by our model. The notation P^{e_1,e_n} represents a path starting from the network entity e_1 and terminating into the network entity e_n. Each *path* is a tuple of policy implementations $(i_1, i_2, \dots, i_n)$ where:

- the source of the first PI i_1 is e_1;
- the destination of the last PI i_n is e_n;

- given two consecutive PIs in the path i_j and i_{j+1}, the property $d_j \in S_{j+1}$ holds.

For instance, a path from c_{c2} to s_{c2} is:

$i_1 = (c_{c2}, g_{c3}, \text{NULL}, (0,0,0), *, \varnothing)$

$i_2 = (g_{c3}, g_{c2}, \text{IPsec}, (3,3,3), (subnet_{c_c}, *, subnet_{c_s}, *, *), \varnothing)$

$i_3 = (g_{c2}, s_{c2}, \text{NULL}, (0,0,0), *, \varnothing)$

Since P^{e_1,e_n} is a set, we will use the notation $|P^{e_1,e_2}|$ to indicate its cardinality, that is the number of policy implementations that compose it.

Note that two paths $P_1^{e_1,e_n}$ and $P_2^{e_1,e_n}$ are different ($P_1^{e_1,e_n} \neq P_2^{e_1,e_n}$) if they differ by at least one element and/or if their respective PIs are placed in different orders.

VI. Anomaly analysis and resolution

Having formalized the definition of a policy implementation, we can now express the logic formulas used to detect the various anomalies.

In Section IV, we introduced an anomaly classification based on five macro-categories (Fig. 2), which emphasizes the side effects of an anomaly. However in the following paragraphs, we will use a more technical classification (Fig. 5), better suited for a more formal discussion. In fact, such classification highlights the possible levels of interactions among PIs and, hence, at which level the anomaly is generated. We distinguish three levels of anomalies: 1) the *PI level anomalies* that occur within a single PI; 2) the *node level anomalies*, which come up between two distinct PIs placed on the same node; 3) the *network level anomalies* arising between distinct PIs that belong to different nodes.

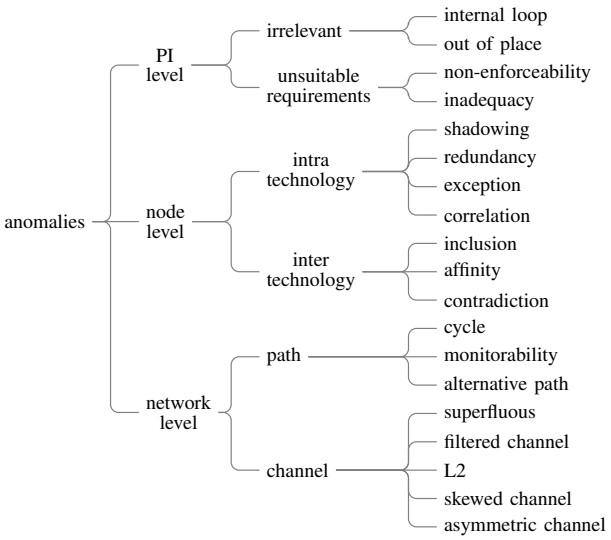


Fig. 5: Information-centric taxonomy of anomalies.

A. PI level anomalies

We can distinguish two families of PI level anomalies: irrelevant and unsuitable requirement anomalies. A PI that generates an *irrelevant anomaly* (which splits into internal loop and out of place) is meaningless for the network semantics, so that their presence does not change how a network exchanges the data. The *unsuitable requirement anomalies* (non-enforceability and inadequacy) instead break some security requisite and they can lead to severe problems.

1) Internal loop – $\mathcal{A}_{il}(i_1)$: There is an *internal loop anomaly* when the source and destination end-points are on the same node, thus creating a communication loop. These anomalies can be inferred by using the formula:

$$\mathcal{A}_{il}(i_1) \Leftrightarrow s_1 \not\perp d_1 \tag{1}$$

The proposed resolution method is to simply delete i_1.

2) Out of place – $\mathcal{A}_{op}(i_1)$: There is an *out of place anomaly* when a PI is deployed on a wrong network node. That means that the source is disjoint with the node where the PI is deployed. To detect these anomalies, we use the function $\mathcal{N}(i_1)$ that returns the node where the PI is actually deployed. The formula is then:

$$\mathcal{A}_{op}(i_1) \Leftrightarrow \mathcal{N}(i_1) \perp s_1 \tag{2}$$

The simplest resolution is to delete i_1. However, a more suitable approach can be to redeploy the PI on the correct node or to appropriately modify its source.

3) Non-enforceability – $\mathcal{A}_{ne}(i_1)$: A PI i_1 is non-enforceable when its technology is not supported by the source, the destination or when its security coefficients are 'too high', and hence cannot be enforced (e.g., due to missing strong encryption algorithms).

We will make use of two functions: $\mathcal{T}(e)$, which returns the set of technologies supported by the node e, and $C_{max}(i_1)$, which returns the set of maximum enforceable coefficients by the PI i_1. These anomalies can be identified with the formula:

$$\mathcal{A}_{ne}(i_1) \Leftrightarrow C_1 \succ \mathcal{C}_{max}(i_1) \vee t_1 \notin \mathcal{T}(s_1) \vee t_1 \notin \mathcal{T}(d_1)$$

To resolve these anomalies, an administrator can choose to upgrade the security libraries/services on the PI source/destination to support the desired technologies or, alternatively, he might modify the PI by changing the protocol or lowering the security coefficients (at his own risk).

4) Inadequacy – $\mathcal{A}_{in}(i_1)$: We have an *inadequacy anomaly* when the security coefficients of a policy implementation establish a channel with a security that is lower than an acceptable threshold. We can use a function $C_{min}(i_1)$ that returns the minimum acceptable coefficients for the channel defined by the PI i_1. This function should be defined a priori by the administrators according to a (corporate) metric or best practice [10], [11], [12]. For example a network administrator could define a function such as:

$$C_{min}(i_1) = \begin{cases} (1,1,1) & \text{if } i_1 \text{ is crossing the Internet} \\ (0,0,0) & \text{otherwise} \end{cases}$$

We can detect these anomalies with the rule:

$$\mathcal{A}_{in}(i_1) \Leftrightarrow C_1 \prec \mathcal{C}_{min}(i_1) \tag{3}$$

In order to fix these issues, the security requirements of the policy implementation must be increased so that the property $C_1 \succeq \mathcal{C}_{min}(i_1)$ holds.

B. Node level anomalies

A *node level anomaly* occurs between two distinct policy implementations laying on the same node.

If the two PIs have the same technology then we have an *intra-technology anomaly* (shadowing, exception, redundancy and correlation), otherwise we have an *inter-technology anomaly* (inclusion, affinity and contradiction). The intra-technology anomaly category has been heavily inspired by the work of Hamed et al.'s [5].

For detecting these anomalies, we assume that the two PIs have the same crossed gateways, that is $G_1 = G_2$. In addition, we will also make use of the function $\pi(i) \in \mathbb{N}$ that returns the priority of a PI in a PI set (the lower the number the higher the priority).

1) Shadowing – $\mathcal{A}_{sh}(i_1, i_2)$: A PI i_2 is *shadowed* when there is another policy implementation i_1 with a higher priority that matches all the traffic of the first one ($s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge S_1 \succeq S_2$) and has disjoint security coefficients. We can detect these anomalies using the formula:

$$\mathcal{A}_{sh}(i_1, i_2) \Leftrightarrow \pi(i_1) < \pi(i_2) \wedge t_1 = t_2 \wedge s_1 \succeq s_2 \wedge$$
$$d_1 \succeq d_2 \wedge S_1 \succeq S_2 \wedge C_1 \perp C_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2 \quad (4)$$

In order to resolve these kind of anomalies, either the shadowed PI is deleted or the two PIs are replaced by another PI i_3 that is an upper bound of the previous ones. In particular, i_3 will have the following fields:

- s_3 is the least upper bound of s_1 and s_2 such that $s_3 \succeq s_1$ and $s_3 \succeq s_2$ hold;
- d_3 is the least upper bound of d_1 and d_2 such that $d_3 \succeq d_1$ and $d_3 \succeq d_2$ hold;
- $C_3 = \{c_{3,i}\}_i$ can be computed as $c_{3,i} = \max(c_{1,i}, c_{2,i})$ where $C_1 = \{c_{1,i}\}_i$ and $C_2 = \{c_{2,i}\}_i$;
- S_3 is the least upper bound of S_1 and S_2 such that $S_3 \succeq S_1$ and $S_3 \succeq S_2$ hold;
- $t_3 = t_1 = t_2$, $G_3 = G_1 = G_2$.

To maintain the semantics of the system, the new PI i_3 should be inserted at the highest priority (i.e. $\pi(i_1)$).

2) Redundancy – $\mathcal{A}_{re}(i_1, i_2)$: A PI i_2 is *redundant* when there is another policy implementation i_1 with a higher priority that matches all the traffic of the first one and its security coefficients are equal or dominates the other PI's coefficients. The following formula can be used to infer these problems:

$$\mathcal{A}_{re}(i_1, i_2) \Leftrightarrow t_1 = t_2 \wedge s_1 \succeq s_2 \wedge$$
$$d_1 \succeq d_2 \wedge S_1 \succeq S_2 \wedge C_1 \succeq C_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2 \quad (5)$$

The proposed resolution is to delete i_2, because it does not add new semantics to the policy.

3) Exception – $\mathcal{A}_{ex}(i_1, i_2)$: A PI i_2 is an *exception* of another policy implementation i_1 with a higher priority if they have disjoint security coefficients and i_2 is a superset match of i_1 ($s_1 \prec s_2 \wedge d_1 \prec d_2 \wedge S_1 \prec S_2$) . The relative detection formula is:

$$\mathcal{A}_{ex}(i_1, i_2) \Leftrightarrow \pi(i_1) \prec \pi(i_2) \wedge t_1 = t_2 \wedge s_1 \prec s_2 \wedge$$
$$d_1 \prec d_2 \wedge S_1 \prec S_2 \wedge C_1 \perp C_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2 \quad (6)$$

Exceptions are analogous to the shadowing anomalies (just the opposite order of precedences) thus they share the same resolution approach.

4) Correlation – $\mathcal{A}_{co}(i_1, i_2)$: A PI i_2 is *correlated* with another policy implementation i_1 if they have disjoint security coefficients, i_1 matches some traffic for i_2 and vice versa. In other words, the source and destination of i_1 and i_2 belong to the same node and there is no other intra-technology anomaly between policy implementations (i.e. shadowing, redundancy or exception). We can detect these anomalies via the formula:

$$\mathcal{A}_{co}(i_1, i_2) \Leftrightarrow s_1 \not\preceq s_2 \wedge d_1 \not\preceq d_2 \wedge t_1 = t_2 \wedge$$
$$S_1 \not\preceq S_2 \wedge G_1 = G_2 \wedge \neg\mathcal{A}_{sh}(i_1, i_2) \wedge$$
$$\neg\mathcal{A}_{ex}(i_1, i_2) \wedge \neg\mathcal{A}_{re}(i_1, i_2) \wedge i_1 \neq i_2 \quad (7)$$

To resolve these anomalies, the two PIs i_1 and i_2 can be replaced with a new PI i_3 with the same fields as described in the shadowing anomaly resolution technique. However, the newly created policy implementation will be inserted with a priority $\pi(i_3) = \min(\pi(i_1), \pi(i_2))$.

5) Inclusion – $\mathcal{A}_{in}(i_1, i_2)$: The PI i_1 *includes* (or dominates) the policy implementation i_2 when all fields of i_1 dominate or are equal to their respective i_2 fields, except one that is strictly dominant. We can detect these anomalies with:

$$\mathcal{A}_{in}(i_1, i_2) \Leftrightarrow s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge t_1 \succeq t_2 \wedge$$
$$C_1 \succeq C_2 \wedge S_1 \succeq S_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2 \quad (8)$$

The simplest way to resolve these anomalies is to delete i_2 (the 'innermost' PI). However, an administrator can also keep both the policy implementations for a security-in-depth approach.

6) Affinity – $\mathcal{A}_{af}(i_1, i_2)$: A PI i_1 is *affine* with another policy implementation i_2 when they share some fields, but none of the PIs includes the other. We can detect these anomalies with the formula:

$$\mathcal{A}_{af}(i_1, i_2) \Leftrightarrow s_1 \not\preceq s_2 \wedge d_1 \not\preceq d_2 \wedge t_1 \not\preceq t_2 \wedge$$
$$S_1 \not\preceq S_2 \wedge \neg\mathcal{A}_{in}(i_1, i_2) \wedge \neg\mathcal{A}_{in}(i_2, i_1) \wedge i_1 \neq i_2 \quad (9)$$

To resolve this type of anomalies, the two PIs should be replaced with a new PI i_3 that is an upper bound of the previous ones:

- s_3 is the least upper bound of s_1 and s_2 such that $s_3 \succeq s_1$ and $s_3 \succeq s_2$ hold;
- d_3 is the least upper bound of d_1 and d_2 such that $d_3 \succeq d_1$ and $d_3 \succeq d_2$ hold;
- t_3 is the least upper bound of t_1 and t_2 such that $t_3 \succeq t_1$ and $t_3 \succeq t_2$ hold;
- $C_3 = \{c_{3,i}\}_i$ can be computed as $c_{3,i} = \max(c_{1,i}, c_{2,i})$ where $C_1 = \{c_{1,i}\}_i$ and $C_2 = \{c_{2,i}\}_i$;
- S_3 is the least upper bound of S_1 and S_2 such that $S_3 \succeq S_1$ and $S_3 \succeq S_2$ hold;
- $G_3 = G_1 = G_2$.

7) Contradiction – $\mathcal{A}_{co}(i_1, i_2)$: Two PIs i_1 and i_2 are in a *contradiction* if their sources/destinations lay on the same node but their technologies are disjoint (that is one PI is using the NULL technology and the other one a security protocol). The formula for detecting these anomalies is:

$$\mathcal{A}_{co}(i_1, i_2) \Leftrightarrow s_1 \not\perp s_2 \wedge d_1 \not\perp d_2 \wedge$$
$$t_1 \perp t_2 \wedge S_1 \not\perp S_2 \wedge i_1 \neq i_2 \quad (10)$$

The resolution is the removal of one PI, however, due to the high ambiguity of the situation is up to the administrator to choose which one (we cannot automatically pick among 'protect' and 'do not protect').

C. Network level anomaly

The network level anomalies occur between distinct PIs that belong to different nodes. We can split these anomalies into

two main categories: path (cyclic path, monitorability and alternative path anomalies), and channel anomalies (superfluous, filtered channel, L2, skewed channel and asymmetric channel anomalies).

1) Superfluous – $\mathcal{A}_{su}(i_1)$: A PI i_1 is *superfluous* if it models a tunnel that protects less than all its inner end-to-end channels. That is, the security coefficient of a superfluous tunnel i_1 are smaller than all the encapsulated channels ($\forall\ i_k : s_k \in S_1|_{ip_{src}\times p_{src}\times\ldots} \wedge G_k^* \supset G_1^*$). This anomalous PIs can be detected by using the formula:

$$\mathcal{A}_{su}(i_1) \Leftrightarrow \nexists\ i_k : s_k \in S_1|_{ip_{src}\times p_{src}\times\ldots} \wedge G_k^* \supset G_1^* \wedge C_k \prec C_1 \quad (11)$$

Since all the data transported in the tunnel are better protected than the tunnel itself, the obvious resolution is to delete i_1 (since it is superfluous). However, as in the inclusion anomaly, an administrator could choose to keep the PI to (slightly) increase the security of the network.

2) Skewed channel – $\mathcal{A}_{sk}(i_1, i_2)$: Two PIs i_1 and i_2 that define two tunnels are *skewed* if their respective channels overlap. This type of anomalies are tricky because in a portion of the network the traffic will be sent without any form of encryption (Fig. 3). We can detect these anomalies with:

$$\mathcal{A}_{sk}(i_1, i_2) \Leftrightarrow s_1 \in S_2|_{ip_{src}\times p_{src}\times\ldots} \wedge (|G_1^* \cap G_2^*|) \geqslant 2 \wedge (G_2^* \setminus G_1^* \neq 0) \wedge c_1^c > 0 \wedge c_2^c > 0 \wedge i_1 \neq i_2 \quad (12)$$

In order to resolve this kind of anomalies, the two PIs must be split in three (or more) non-overlapping policy implementations.

3) Filtered channel – $\mathcal{A}_{fi}(i_1)$: A PI i_1 is *filtered* when there exists at least one node e in its path with a filtering rule that discards all its traffic. Given a function $\mathcal{F}_e(i_1)$, which returns true if the traffic related to i_1 is dropped and false otherwise, we can formally model this anomaly with:

$$\mathcal{A}_{fi}(i_1) \Leftrightarrow \exists e : e \in G_1 \wedge \mathcal{F}_e(i_1) = true \quad (13)$$

In practice, the output of the function $\mathcal{F}_e(i_1)$ can be populated either by means of a network reachability analysis [13] or by using some firewall policy queries [14].

This anomaly is particularly severe since it completely hinders the connectivity between a number of network nodes. To remove the problem, the administrator can choose to delete the PI i_1 or modify accordingly the filtering rule.

4) L2 – $\mathcal{A}_{L2}(i_1)$: An *L2 anomaly* occurs when a PI that uses a data-link layer technology crosses an area using a different layer 2 protocol. For instance, we have a L2 anomaly when a WPA2 policy implementation crosses some Ethernet nodes, so that we cannot use WPA2 for the whole path. We can express this anomaly by using a function $\mathcal{T}^{(2)}(e)$ that returns the set of technologies at layer 2 supported by the node e. We can then write the formula:

$$\mathcal{A}_{L2}(i_1) \Leftrightarrow \exists e : e \in G_1^* \wedge t_1 \notin \mathcal{T}^{(2)}(e) \quad (14)$$

These anomalies are quite hard to resolve since they require a complete edit of the PI, by choosing a technology at a layer strictly greater than 2.

5) Asymmetric channel – $\mathcal{A}_{as}(i_1)$: A PI i_1 is *asymmetric* if does not exist another PI with: 1) the source and destination swapped ($s_1 \not\perp d_2 \wedge d_1 \not\perp s_2$); 2) the same technology and security coefficients; 3) the same list of crossed gateways, but in reverse order. In other words, these problems arise when we have a bidirectional communication with a channel weaker than the other. We can identify these anomalies by using the formula:

$$\mathcal{A}_{as}(i_1) \Leftrightarrow \nexists i_2 : s_1 \not\perp d_2 \wedge d_1 \not\perp s_2 \wedge t_1 = t_2 \wedge C_1 = C_2 \wedge S_1 \not\perp \overleftarrow{S_2} \wedge G_1 = \overleftarrow{G_2} \quad (15)$$

Administrators must check these anomalies to verify if they represent the wanted protection.

6) Cyclic path – $\mathcal{A}_{cy}(P^{e_1,e_2})$: There is a *cyclic path* anomaly between two nodes e_1 and e_2 if there is at least one cycle in the path connecting them. These anomalies can be detected with any of the several very efficient algorithms available in literature to perform cycle detection [15].

The only way to resolve this kind of anomalies is to modify the PIs to remove the cycles.

7) Monitorability – $\mathcal{A}_{mo}(P^{e_1,e_2})$: A path P^{e_1,e_2} is *monitorable* when there is not an end-to-end channel between e_1 and e_2. That means that, even if the connections are protected by encryption, there is at least one node where an encrypt/decrypt operation is performed, thus potentially breaking the confidentiality of the communication. These anomalies can be detected by using the formula:

$$\mathcal{A}_{mo}(P^{e_1,e_2}) \Leftrightarrow \nexists P^{e_1,e_2} : (|P^{e_1,e_2}| = 1 \wedge i_j \in P^{e_1,e_2} : c_j^c > 0) \quad (16)$$

If the network is not trusted, the obvious way to remove this anomaly is to edit the PIs such that there are only end-to-end channels between e_1 and e_2 .

8) Alternative path – $\mathcal{A}_{al}(P_1^{e_1,e_2}, P_2^{e_1,e_2})$: There is an *alternative path* between two nodes e_1 and e_2 if there are two or more different paths that can be taken from the source node to the destination node. These anomalies can be easily found by using the formula:

$$\mathcal{A}_{al}(P_1^{e_1,e_2}, P_2^{e_1,e_2}) \Leftrightarrow \exists P_1^{e_1,e_2}, P_2^{e_1,e_2} : P_1^{e_1,e_2} \neq P_2^{e_1,e_2} \quad (17)$$

To remove this redundancy, the administrators have to choose the 'best' path for the communication and delete the other ones. The choice can be made by using different strategies, such as picking the shortest path or the path containing the PIs with the highest security coefficients.

VII. Graph-based representation of the anomalies

Aiming for a model that also has practical relevance, we investigated the possibility of a user friendly representation of our anomalies, as logical formulas are not easily usable by administrators. In Section V we have already sketched our hierarchical view of a network node. This allows the representation of secure communications by connecting network nodes to form a multi-graph. The obvious advantage of such representation is that it allows a network administrator to visualize the communications at a glance. Our claim is that it allows the intuitive identification of the anomalies and the consequences and the reactions.

For example, Fig. 6 shows an affinity anomaly between two PIs. The first policy implementation (solid line) enforces IPsec in transport mode and requires only confidentiality, while the second one (dashed line) uses TLS and enforces only

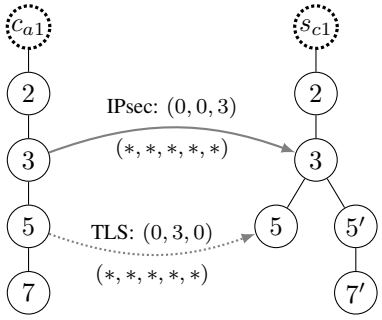


Fig. 6: Graphical representation of an affinity anomaly.

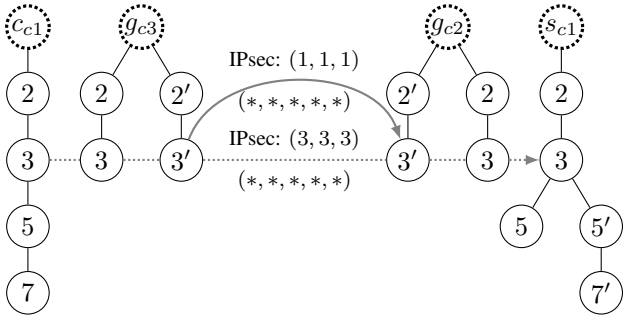


Fig. 7: Graphical representation of a superfluous anomaly.

payload integrity. The graph clearly shows that the two PIs are correlated as there are two "parallel" arrows. These PIs are affine since they share some common aspects, but none of them includes the other one. An administrator may understand that he can alternatively use (1) only the IPsec channel, if he adds also payload and header integrity, (2) only the TLS channel, if he adds confidentiality, or (3) keep both. Another example is shown in Fig. 7, where a superfluous anomaly is depicted. We recall that a channel is superfluous when there is another tunnel that covers at least the same traffic but protects the communication with a higher security level. In this case the IPsec tunnel between g_{c3} and g_{c2} is redundant, so an administrator immediately sees that it can be safely removed.

All the anomalies (but the out of place one) have a corresponding graphical representation. These representations can be built by including the network node trees corresponding to the communication end-points (Fig. 4), that is, the source and the destination of the PI. The PIs that enforce end-to-end channels are represented as single directed edges between two communication vertices, i.e. the proper communication layer nodes. For instance, in Fig. 6 the edge connects the layer 3 nodes as the technology is IPsec. To increase the expressiveness of our representation, each edge is also labelled with the technology, the security coefficients required by the PI and the selectors. To represent the policy implementations that enforce site-to-site and remote-access communications, we add all the network node trees corresponding to the crossed gateways and an edge crossing all the communication parties. For instance, in case of a tunnel (Fig. 7), we introduce an edge crossing the source node, the first gateway, the second gateway and terminating into the destination node.

As anticipated, out of place is the only anomaly that we do not represent graphically. Visualizing this anomaly negligibly boosts practical usefulness of our graphical representation but significantly increases its complexity.

VIII. Model validation

In this section, we present the evaluation of our anomaly analysis model's usefulness and usability.

A. Empirical assessment

In order to evaluate the practical importance of our work, we conducted an empirical assessment. We tried to answer two simple yet interesting research questions:

RQ1. are anomalies presented in this paper actually introduced by the administrators when configuring the CPPs?

RQ2. does the number of anomalies decrease when the administrator expertise grows?

If RQ1 is confirmed, we could deduce that performing the detection can help improving the policy enforcement correctness in real world networks. RQ2 instead can give us insights on users that can benefit the most from our anomaly analysis model.

We mainly focused on the new kinds of anomalies presented for the first time in this paper. For this reason we did not report statistics on anomalies already present in the literature, namely the shadowing, redundancy, exception, correlation, the skewed channel (overlapping sessions) and out of place (irrelevances) whose importance was already proved in other original works [5], [16], [17]. We designed the experiment to be completed by the administrators in one hour, therefore we avoided to provide data link information and kept the size of the network reasonably low. For this reason, the L2, asymmetric channel, cycle and alternative path anomalies were not considered in our study.

In order to answer the research questions, we conducted an experiment by recruiting a set of 30 administrators. We split them into three categories according to their expertise level (high, medium and low), each one containing 10 people. In the test, we have considered as high expertise administrators people with more than two years of experience in the security field, as medium expertise administrators people with more than two years of practice in the (non-security) network field, and as low expertise administrators the remaining ones.

We asked them to enforce five CPPs (e.g. "all the administrators must securely reach the accounting service") by implementing them as a set of PIs. The landscape was a small network (consisting of 5 subnets, 6 servers, 9 clients and 10 gateways). The network description and the CPPs were available online to the participants both as a web page and as a PDF document to be accessed offline. The participants were asked to write all PIs where all the their fields where constrained to valid values (e.g. correct node and protocol names) to avoid uninteresting errors. We did not impose neither a time limit nor a maximum number of PIs.

The analysis of experiments data gave us very interesting information. First of all, 93% of administrators introduced at least one anomaly, regardless of the expertise, as shown in Table I. In addition, all the new anomalies have been intro-

expertise	insecure communications	unfeasible communications	potential errors	suboptimal implementations	at least one type
low	70.00%	60.00%	60.00%	70.00%	100.00%
medium	60.00%	30.00%	50.00%	40.00%	90.00%
high	30.00%	20.00%	20.00%	70.00%	90.00%
average	53.33%	36.67%	43.33%	60%	93.33%

TABLE I: Percentage of administrators that created at least one anomaly in a macro-category.

expertise	internal loop	non-enforceability	inadequacy	inclusion	affinity	monitorability	superfluous	filtered	contradiction
low	20.00%	30.00%	40.00%	30.00%	50.00%	30.00%	30.00%	30.00%	30.00%
medium	10.00%	20.00%	40.00%	20.00%	40.00%	20.00%	30.00%	10.00%	10.00%
high	10.00%	10.00%	10.00%	20.00%	20.00%	20.00%	50.00%	10.00%	0.00%
average	13.33%	20.00%	30.00%	23.33%	36.67%	33.33%	30.00%	16.33%	13.33%

TABLE II: Percentage of administrators that created at least one anomaly.

expertise	insecure communications	unfeasible communications	potential errors	suboptimal implementations	total
low	18.41%	22.39%	15.92%	7.46%	64.18%
medium	16.54%	12.78%	9.77%	9.77%	48.87%
high	12.90%	4.03%	2.42%	12.90%	32.26%
average	16.38%	14.63%	10.48%	9.61%	51.09%

TABLE III: Percentages of anomalies introduced by the administrators grouped in macro-categories.

expertise	internal loop	non-enforceability	inadequacy	inclusion	affinity	monitorability	superfluous	filtered	contradiction
low	1.49%	4.48%	10.95%	2.99%	6.97%	2.99%	7.46%	17.91%	8.96%
medium	1.50%	3.76%	13.53%	4.51%	5.26%	3.01%	3.76%	9.02%	4.51%
high	1.61%	0.81%	4.03%	3.23%	2.24%	8.87%	8.06%	3.23%	0.00%
average	1.53%	3.28%	9.83%	3.49%	5.24%	6.55%	4.59%	11.35%	5.24%

TABLE IV: Percentages of anomalies introduced by the administrators.

duced by at least one administrator (Table II). Interestingly enough, all the anomaly types except contradictions were also introduced by expert administrators. This result successfully answered positively the research question RQ1, that is the anomalies presented in this paper can appear in real world scenario, hence it is useful to look for them.

The RQ2 research question (the more the expertise of administrators the less the anomalies) has been also confirmed for all the anomaly macro-categories except one, the suboptimal implementations (Table III). Obviously having a better understanding of a network and its different security controls, reduces the chance of introducing anomalies. This is particularly evident for the filtered anomalies, as the administrators also have to consider the interactions with firewalls to avoid them, but it is also valid for the non-enforceability, inadequacy, affinity, and contradiction anomalies (Table IV). On the other hand, the suboptimal implementations tend to increases because the expert administrators add more superfluous anomalies, most likely for providing a defense in depth approach, although this was not expressly required in the exercise. Moreover, expert administrators' PIs also contain several monitorability anomalies, since they tend to make an extensive use of tunnels while the less skilled ones mainly use end-to-end channels. In short, the experienced administrators tend to break secure communications to improve the overall network performance. In this sample network, the monitorability anomalies are not the most serious issues (as we had homogeneous security levels in all networks), however, in general it is worth checking them. Finally, there is a number of internal loop anomalies, constant with the expertise, probably due to distraction errors.

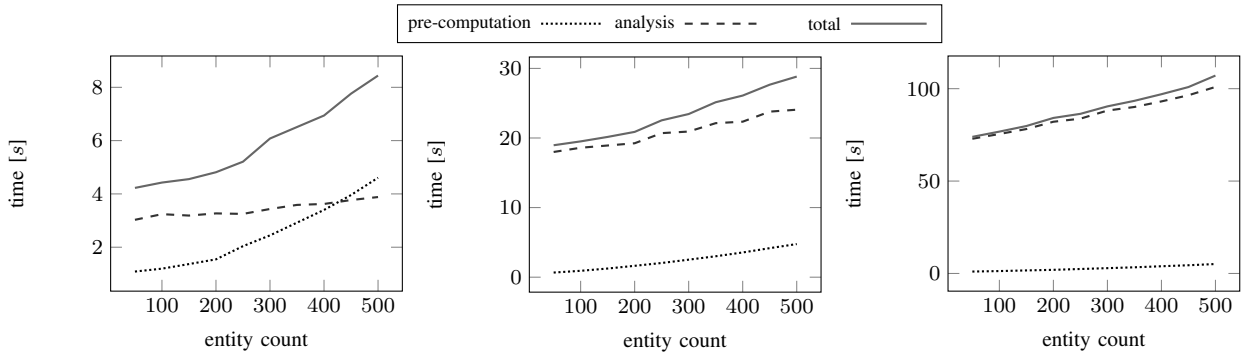
The statistical significance of the differences between the results on the three expertise levels, has been assessed with the analysis of variances (ANOVA) with a significance level of 0.05. We performed the test on the number of error types and errors for each administrator and obtained two P-values of 0.034 and 0.006, thus successfully proving the hypothesis.

We identified two major threats to the validity of our experiment. First, for practical reasons (anonymity, expected number of participants, web-based data collection) we provided a single administration task, therefore, the experiment is subject to the mono-operational bias. This threat has been mitigated as the proposed task was realistic and complex enough to capture the heterogeneity of administrators' tasks. Then, the other threat concerns the generalization of the results, as participants may not approximate well security administrators. However, participants were selected on a voluntary basis, so they were motivated, they performed the task at their place with the most familiar environment, and they had unlimited time; therefore, in ideal conditions. Moreover, the ex post analysis of the correlation with the expertise didn't show inconsistencies.

More information about the experiments, the input policy and network, the observed data, the statistical assessment, and the threats to validity are available in the supplemental material of this paper.

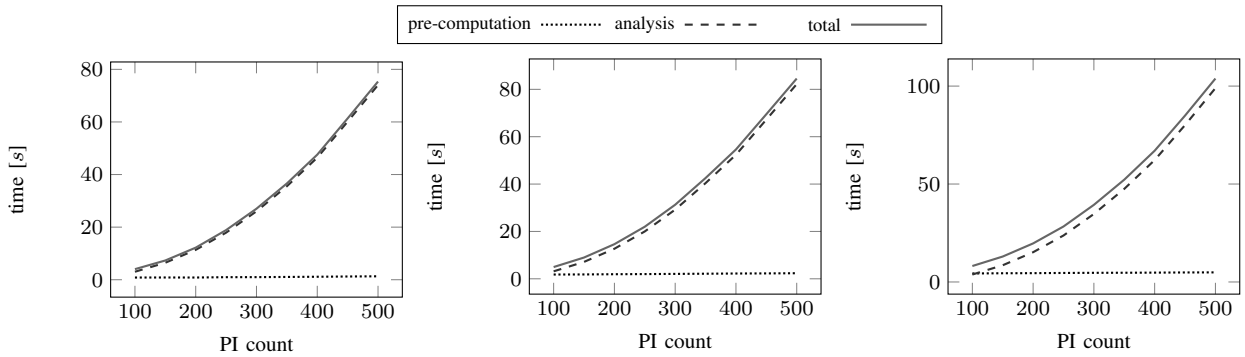
B. Complexity analysis

We will now derive some complexity formulas that prove the theoretical performance of our model. We will start with



(a) With 100 PIs. (b) With 250 PIs. (c) With 500 PIs.

Fig. 8: Performance tests: time to perform the anomaly analysis of a fixed number of PIs depending on the number of entities.



(a) With 100 entities. (b) With 250 entities. (c) With 500 entities.

Fig. 9: Performance tests: time to perform the anomaly analysis on networks of a fixed size depending on the number of PIs.

a simple observation. Our approach can be split in two consecutive phases. The first one is a *pre-computation phase*, where the tree representation of the network and its paths are obtained. The second one is an *analysis phase* that consists of the real anomaly detection phase.

Let's suppose that we have a network consisting of $\mathcal{E}$ entities (IPs, ports, addresses, ...), $\mathcal{I}$ policy implementations and $\mathcal{C}$ connections between the network entities created by the PIs (obviously $\mathcal{C} \geq \mathcal{I}$).

We will start by taking a look at the pre-computation phase. To create the tree representation of the network nodes, we need to check every single entity, so that this process has an exact complexity of $\mathcal{E}$. Finding all the simple paths[4] in an acyclic graph is a NEXPTIME problem with a maximum complexity of $O(e^{\mathcal{E}})$. Note, however, that the real networks are scarcely connected and that multiple paths between two different nodes are quite rare, making these calculations feasible also in large IT infrastructures. In addition, an administrator can choose to limit the number of paths to check to some fixed value $\mathcal{P}$, typically $\mathcal{P} \lll e^{\mathcal{E}}$. Hence, the total complexity of the pre-computation phase is $\mathcal{E} + \mathcal{P}$.

Regarding the analysis phase, we have to take into account the different characteristics of the anomaly detection formulas. In particular, we have that:

- the internal loop, out of place, non-enforceability, inadequacy, filtered channel, L2 and asymmetric channel anomalies algorithms work on a single PI at a time, so that they have a complexity of $\mathcal{I}$;
- the shadowing, redundancy, exception and inclusion anomalies algorithms need an ordered pair of PIs, thus have a complexity of $\mathcal{I}(\mathcal{I}-1)$. Also the superfluous anomaly has a quadratic complexity since it needs to test every PI against all the other ones;
- the correlation, affinity and skewed channel anomalies algorithms work on unordered pairs of PIs, giving a complexity of $\mathcal{I}(\mathcal{I}-1)/2$;
- the monitorability and alternative path anomalies algorithms work on every path, hence their complexity is $\mathcal{P}$;
- the cyclic path anomaly algorithm can be efficiently performed using a proper cycle detection algorithm such as [15], that has a complexity of $O(\mathcal{E}+\mathcal{C})$. Note that its complexity is not necessarily $\mathcal{P}$ since a graph with few loops may have an infinite number of paths.

Summarizing, the total complexity of the analysis phase is:

$$\mathcal{I} + \mathcal{I}(\mathcal{I}-1) + \mathcal{P} + O(\mathcal{E}+\mathcal{C}) \approx \mathcal{I}^2 + \mathcal{P} + O(\mathcal{E}+\mathcal{C})$$

C. Performance analysis

We implemented our anomaly detection model and tested it in several scenarios using a number of synthetically generated networks in order to assess its running time. Our tool was developed using Java 1.8 and the natural graph-based repre-

[4] A simple path is a path with no duplicate vertexes.

sentation of ontologies offered by OWL API 3.4.10 and the reasoner Pellet 2.3.1. We performed all our tests on an Intel i7 @ 2.4 GHz with 16 GB RAM under Windows 7.

Each test was performed on several ad-hoc scenarios consisting of an automatically generated network with a parametric structure where we can specify: 1) the number of network entities; 2) the number of policy implementations; 3) the percentage of conflicting PIs. We choose to fix the number of conflicting PIs to about 50% since, from our empirical analysis, on average, an administrator writes only about half of the policy implementations without any kind of conflict (Table III). We performed two kinds of tests. In the first one we fixed the number of network entities and increased the number of PIs, while in the second one we did the reverse (fixed the number of PIs and changed the entities count). Fig. 8 shows the test results when fixing the number of PIs respectively to 100, 250 and 500, while Fig. 9 shows the graphs when the network entities count is 100, 250 and 500. We kept track of three times: the pre-computation (the dotted lines), the analysis phase (the dashed lines) and the total times (the solid lines).

Our tool proved to be very scalable, achieving a total time of less than two minutes in the worst scenario (500 PIs and 500 network entities). We also noted that the results are aligned with the complexity analysis in Section VIII-B. For instance, the computation times grow when the number of network entities (Fig. 8) increase, while the pre-computation phase time is independent of the number of entities (Fig. 9).

IX. Related works

Anomaly analysis, detection and resolution in policy-based systems and security controls are hot topics. We briefly discuss in the next paragraphs the most notable works in this area.

A. Communication protection policies

The current literature contains several works about anomaly detection in CPPs, however, the research in this area is solely focused on IPsec, and overlooks the effects of multiple overlapping protection techniques.

Zao [18] introduced an approach based on the combination of conditions that belong to different IPsec fields. The same idea was used by Fu et al. [19] to describe a number of conflicts between IPsec tunnels, discovered through a simulation process that reports security requirements violations. In their analysis, the policy anomalies are identified by checking the IPsec configurations against the desired policies written in a natural language. In practice, an anomaly occurs when the policy implementations do not satisfy the desired policies. In addition, Fu et al. proposed a resolution process that finds alternative configurations to satisfy the desired policy.

Hamed et al.'s analyzed the effects of IPsec rules on the protection of networks [5], by proposing a number of ad-hoc algorithms and formulas to detect these problems. They formalized the classification scheme of [19] and proposed a model based on OBDD (Ordered Binary Decision Diagrams) that not only incorporates the encryption capabilities of IPsec, but also its packet filter capabilities. He also identified two new IPsec problems (channel-overlapping and multi-transform anomalies). The first one occurs when multiple IPsec sessions are established and the second session redirect the traffic of the first one (similar to the case depicted in Fig. 3). On the other hand, the multi-transform anomalies occur when a data protection is applied to an already encapsulated IPsec traffic and the second protection is weaker than the first one. The same authors also described a classification system for conflicts between filtering and communication protection policies [16]. Niksefat et al. [20] presented an two improvements over the Hamed et al.'s solution [5], a faster detection algorithm and the possibility to resolve the detected anomalies. Finally, Li et al. classified the IPsec rules in two classes: access control lists (ACL) and encryption lists (EL) [21].

B. Filtering policies

Configuration/policy anomaly detection in networks is not only restricted to communication protection technologies. In literature there exist a rich collection of papers about filtering policy analysis. These works belong to a different domain w.r.t. the approach presented in this paper, however they are interesting background on anomaly analysis. Note that in our approach we perform a full-spectrum CPP analysis that takes also into account the effects of filtering through the filtered channel anomaly, while the works presented in the following paragraphs focus solely on the conflicts between filtering rules.

One of the most influential works in this area is by Al-Shaer et al., which addresses the analysis of filtering configurations via FOL formulas [3]. The authors analyzed both the local anomalies arising in a single firewall and the global ones taking into account several distributed filtering devices.

Liu et al. focused on detecting and removing redundant filtering rules with data-structure named FDD (Firewall Decision Diagram) [22]. The authors distinguish upward redundant rules, which are rules that are never matched, and downward redundant rules, which are rules that are matched but enforce the same action as some lower priority rules.

Basile et al. described a geometric representation of filtering rules (up to the application level) where detection and resolution is based on the intersection of hyper-rectangles [9], [23], [24]. The authors extended the work performed by Al-Shaer by introducing the anomalies between more than two rules and by showing how to transform a policy representation in another form that preserves its semantic. Similarly, Hu et al. suggested to split the classic five-tuple decision space into disjoint hyper-rectangles, induced by the rules, where conflicts are resolved with a combination of automatic strategies [25].

Hu et al. proposed a thoroughly different approach, an ontology-based management framework that detects anomalies in filtering rule sets [26]. Analogously, Bandara et al. proposed the use of logic reasoning, obtaining excellent performance [27]. Alfaro et al. presented a collection of algorithms to remove a series of anomalies between packet filter configurations and NIDS in distributed systems [17] that have been implemented in the MIRAGE tool [28].

X. Conclusions and future work

In this paper we proposed a list of nineteen anomalies (classified in two taxonomies) that can arise during the imple-

mentation of communication protection policies, and devised a formal model based on FOL formulas that is able to detect them. Our approach can be used to find incompatibilities, redundancies and severe errors among policy implementations that use security technologies working at different OSI layers and with different security properties.

We implemented our model in Java by making an extensive use of ontological techniques, and verified in several network scenarios that it is scalable and performs well. Most of the anomalies can be visualized with a graph-based representation that should facilitate their identification.

For the future, we plan to extend the expressivity of our model by adding support for new types of network devices, such as intrusion detection systems (IDS). Furthermore, we are planning to perform other empirical assessments to evaluate if our tool can help administrators to reduce the number of anomalies in real-world scenarios.

REFERENCES

[1] A. Wool, "Trends in firewall configuration errors: measuring the holes in Swiss cheese," *IEEE Internet Computing*, vol. 14, no. 4, pp. 58–65, Jul. 2010.

[2] Verizon, "Data Breach Investigations Report," 2015.

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

[4] Cisco Systems, "User Guide for Cisco Security Manager 4.10," pp. 731–776, Dec. 2015.

[5] H. Hamed, E. Al-Shaer, and W. Marrero, "Modeling and verification of IPsec and VPN security policies," in *13th IEEE Int. Conf. on Network Protocols*, Nov. 2005, pp. 259–278.

[6] European Commission, "Directive 95/46/ec- protection of personal data," Jul. 2000. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32000D0520&from=EN

[7] C. Basile, D. Canavese, A. Lioy, and F. Valenza, "Inter-technology conflict analysis for communication protection policies," in *9th Int. Conf. on Risks and Security of Internet and Systems*, Trento (Italy), Aug. 2014, pp. 148–163.

[8] Amina, "Automatic detection and correction of firewall misconfigurations: A formal approach," in *International Symposium on Symbolic Computation in Software Science*, vol. 45, Mar. 2017, pp. 68–76.

[9] C. Basile, A. Cappadonia, and A. Lioy, "Network-level access control policy analysis and transformation," *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, pp. 985–998, Aug. 2012.

[10] NSA's Information Assurance Directorate, "NSA Mitigation Guidance." [Online]. Available: https://www.nsa.gov/ia/mitigation_guidance/index.shtmls

[11] National Institute of Standard and Technology, "Recommendations of the National Institute of Standards and Technology." [Online]. Available: http://csrc.nist.gov/publications/PubsTC.html

[12] International Organization for Standardization and International Electrotechnical Commission, "ISO/IEC 27033: Information technology – security techniques – network security," 2009.

[13] C. Basile, D. Canavese, A. Lioy, and C. Pitscheider, "Improved reachability analysis for security management," in *Euromicro Int. Conference on Parallel, Distributed, and Network-Based Processing*, Belfast (UK), February 27 - March 1, 2013, pp. 534–541.

[14] A. Khakpour and A. X. Liu, "Quarnet: A tool for quantifying static network reachability," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 551–565, Feb. 2009.

[15] R. Tarjan, "Depth first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[16] H. Hamed and E. Al-Shaer, "Taxonomy of conflicts in network security policies," *IEEE Communications Magazine*, vol. 44, no. 3, pp. 134–141, Mar. 2006.

[17] 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, Mar. 2008.

[18] J. Zao, "Semantic model for IPsec policy interaction," Internet Draft, Mar. 2000.

[19] 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 *Policies for Distributed Systems and Networks*, Jan. 2001, pp. 39–56.

[20] S. Niksefat and M. Sabaei, "Efficient Algorithms for Dynamic Detection and Resolution of IPSec/VPN Security Policy Conflicts," in *2010 24th IEEE Intern. Conference on Advanced Information Networking and Applications*, Apr. 2010, pp. 737–744.

[21] Z. Li, X. Cui, and L. Chen, "Analysis and classification of IPsec security policy conflicts," in *Japan-China Joint Workshop on Frontier of Computer Science and Technology*, Nov. 2006, pp. 83–88.

[22] A. X. Liu and M. G. Gouda, "Complete Redundancy Detection in Firewalls," in *19th Working Conference on Data and Applications Security*, Aug. 2005, pp. 193–206.

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

[24] C. Basile, D. Canavese, A. Lioy, C. Pitscheider, and F. Valenza, "Inter-function anomaly analysis for correct SDN/NFV deployment," *Int. J. of Network Management*, vol. 26, no. 1, pp. 25–43, 2016.

[25] H. Hu, G.-J. Ahn, and K. Kulkarni, "Detecting and resolving firewall policy anomalies," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 3, pp. 318–331, May 2012.

[26] H. Hongxin, A. Gail-Joon, and K. Ketan, "Ontology-based policy anomaly management for autonomic computing," in *7th Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing*, ser. CollaborateCom, Oct. 2011, pp. 487–494.

[27] A. K. Bandara, A. C. Kakas, E. C. Lupu, and A. Russo, "Using argumentation logic for firewall configuration management." in *Integrated Network Management-Workshops*, Jun. 2009, pp. 180–187.

[28] J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, and S. Preda, "Mirage: A management tool for the analysis and deployment of network security policies," in *5th Int. Workshop on Data Privacy Management*, Sep. 2011, pp. 203–215.

Fulvio Valenza received the M.Sc. degree (summa cum laude) in computer engineering in 2013 from the Politecnico di Torino. In 2016, he completed his PhD in Computer Engineering at the Politecnico di Torino. His research activity focused on network security policies. Currently he is a Researcher at the CNR-IEII Torino, Italy, where he works on orchestration and management of network security functions in the context of SDN/NFV-based networks and industrial systems.

Cataldo Basile received a M.Sc. (summa cum laude) in 2001 and a Ph.D. in Computer Engineering in 2005 from the Politecnico di Torino, where is currently a research assistant. His research is concerned with policy-based management of security in networked environments, policy refinement, general models for detection, resolution and reconciliation of specification conflicts, and software security.

Daniele Canavese received the M.Sc. degree in computer engineering in 2010 from the Politecnico di Torino, where he is currently a research assistant and a Ph.D. student. His research interests are concerned with policy-based management systems, models for network analysis, security management via inferential systems and public key cryptography.

Antonio Lioy is full professor at the Politecnico di Torino, where he leads the TORSEC research group active in information system security. His research interests include network security, policy-based system protection, and electronic identity. Lioy received a M.Sc. in Electronic Engineering (summa cum laude) and a Ph.D. in Computer Engineering, both from the Politecnico di Torino.