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 introduces a novel category, the inter-technology anomalies, which arise among configurations of security controls implementing different technologies, both within the same network node and among different network nodes. Through an empirical assessment, we proved the practical significance and the impact on security of detecting the novel category of anomalies. Furthermore, this paper introduces a formal model, based on first-order logic rules, that analyses the network topology and security controls at each node and identifies, categorize, and reports users on the detected anomalies and the strategies to resolve them. The formal model has manageable computational complexity and its implementation has showed excellent performances and promising scalability properties.**

***Index Terms*—security policies, policy conflicts, policy analysis**

I. Introduction

Enforcing the security in a system is obviously a very complex and delicate task. Security administrators have a hard work to accomplish, a work that requires very specific skills and a high level of competence. However, in the last years, several studies have confirmed their responsibilities in most of the security breaches and breakdowns. Wool [1] showed that most of the firewalls he analyzed contained several problematic policies such as very lax rules. The Data Breach Investigations Report [2] states that about the 60% of the security breaches are due to errors and mistakes made by the internal staff.

On one hand, when looking for the security problem causes, a US government funded study reported years ago that the skills and competence of security administrators were strongly decreased [3]. On the other hand, administrators often have basic or no tool support to debug the security controls configurations in order to check if the enforced policy is correct and compliant to the high-level security requirements.

Configuration analysis methods, which may be very important to detect and correct errors, have been seldom incorporated into industrial grade tools. The only notable exception is the detection of the packet filter anomalies classified by Al-Shaer [4]. In some cases, porting some of these methods is hard because computation complexity makes them unusable or because they consider scenarios that are too far from the reality.

The main contribution of this paper is a formal model that can be used to assist the security administrators when configuring and validating security policies, thus bridging the gap between theory and practice. In particular, we focus on *communication protection policies* (CPPs), that is regulations that determine how to protect corporate assets, such as private user data and corporate intellectual properties, when they are transferred over computer networks. CPPs originate from laws (e.g., the EU privacy law [5]) and business security requirements. In some cases, like companies that host services or provide cloud-based resources, they may be very complex and articulated. CPPs are often expressed with very high-level directives that specify what are the security properties that the communications must guarantee (e.g. confidentiality and integrity). However, in the end, they are enforced by means of a plethora of security controls, which implement different protocols (e.g. IPsec, TLS, WS-Security) that operate at different layers of the ISO/OSI stack. Therefore, the administrators manually determine the end-to-end channels and tunnels to establish, the resources that need to be accessed through secure protocols and the data that need to be protected with message protection techniques. Incorrect implementation of CPPs can result in information disclosure that can cause, for instance, violations of the users privacy, loss of intellectual properties and, eventually, huge money losses.

In the CPPs enforcement scope, several actors are authorized to their enforcement: service administrators, which have to enable channel protection protocols like SSL/TLS on the services they manage, network administrators, which have to configure secure tunnels and end-to-end channels at network and data link level and to enable wireless protection, and other high level roles, sometimes referred to as IT managers (or IT security managers), which interact with business units to determine the communications to protect.

The fact that several entities are involved in the implementation of the CPPs together with the large size and complexity of the networks to configure increases the risk of mistakes, like conflicting implementations of the policies, and redundant channels.

The approach we follow in this paper is to detect and show the administrators the *anomalies*, that are, as defined by Al-Shaer, the presence of redundant or conflicting configuration rules. Anomalies are particular situations that can be the evidence of a human error and thus deserve the explicit attention of the administrators.

Our model can be used in two different scenarios: both

Fulvio Valenza, Cataldo Basile, Daniele Canavese and Antonio Lioy are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, Italy. Email: {fulvio.valenza, cataldo.basile, daniele.canavese, antonio.lioy}@polito.it

design validation and implementation analysis. For the *design validation*, the administrator has to provide the communications he has to protect, the security properties to be enforced and the technology he wants to use. For the *implementation analysis*, the administrator instead has to provide the configurations of all the protection controls. After having determined the communications to be protected, the model then is able to detect several kind of anomalies such as redundant, insecure, non-enforceable and filtered communications.

This paper is structured as follows. Section II lists our contributions to the current state-of-the art. Section III and IV informally introduce our approach by presenting a brief background and a motivating example. Section V and VI are the core of this paper, by describing the formal structures and the formulas of our model. Section VII describes 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 respectively contains the related works and the conclusions.

II. Contributions

Our work pushes the state of the art in several directions. The main contribution of this paper is the identification of nineteen types of anomalies that may happen when configuring the CPPs. Six anomaly types were already known in literature [6], however, most of them are original contributions of this paper[1]. It is worth noting that our classification includes all the data protection anomalies identified in previous works that, however, find a place in a bigger landscape. The anomalies we identified arise in the configuration of the same security control (*intra-policy*), between configurations of the same type displaced in different network nodes (*inter-policy*), and, our novel contribution, among configurations of security controls implementing different technologies, both within the same network node and 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 layer.

As an example, inter-policy anomalies may appear between the configurations of two IPsec VPN gateways, while inter-technology anomalies may appear between the IPsec and TLS configurations of a web server on the same network node, or amongst the WS-Security configuration of a web service and the OpenVPN gateway that tunnels the communications between corporate networks. Moreover, we identify also communications that are intrinsically insecure or non-enforceable by the security control to which they are intended to.

Anomalies are detected by means of a formal model that takes as input the network topology, the available security controls at each node, the communication to secure (or the communications actually secured in case of the implementation analysis). This information is included in a knowledge base that is explored by a set of first-order logic (FOL) formulas used to identify and reported to the users the detected anomalies.

Anomalies have been categorized according to two different classifications: an *effect-based* and *information-centric* taxonomies. The first one divides the anomalies into five macro-categories describing the effects that they have on the network, that is: 1) insecure communications; 2) unfeasible communications; 3) potential errors; 4) suboptimal communications; 5) suboptimal walks. The second classification is based on the information that need to be analysed to detect the anomalies. It split the anomalies in three classes: 1) anomalies in a single communication channel; 2) anomalies between secure channels that starts 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 kind of anomalies, 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 computationally effective to compute them also in large networks?

In order to answer the first question, we have presented for each of the anomalies the possible consequences on the network and some ways to resolve them to reduce 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, 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 a tool making use of DL (description logic) ontologies and some custom Java-based reasoning rules. We performed an extensive testing of its speed in several different scenarios (see Section VIII-C).

III. Background

We introduce here the main concepts and terms that we will use throughout the rest of the paper.

A *communication* is any directional data exchange between two network entities. Amongst them, we define a *secure communication* as a communication that is '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.

We call a *security gateway*, or simply a *gateway*, a network node that is able to terminate a secure communication. Note that clients and servers may also act as gateways, thus, the name 'gateway' does not imply necessarily a custom device.

A *channel* is an atomic directional data exchange with no security property checks occurring between its end-points (no

[1]An embryonal and incomplete set of anomalies has been published in a previous work [7]. This paper is a much more comprehensive, precise, and coherent extension of the previous work.

[2]Protections at transport layer, such as SSL/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 different protections.

decryption or integrity verification). A communication can be thought as an ordered sequence of several (secure and/or insecure) channels.

In the real world, the various communications are defined by using a set of configurations settings containing several low-level details. For instance, the configuration of a TLS server contains detailed information about the supported cipher-suites, often listed in a preference order. During the design and policy analysis phases, however, such level of granularity is usually not needed. For our purposes, a secure channel can be represented by specifying: 1) the source and the destination entities. They can be network nodes or direct references to an entity lying at a particular ISO/OSI 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 ISO/OSI layers; 3) the required security properties; 4) the crossed gateways and the source protected nodes, meaningful only in case of tunneling. We name this formal representation of a channel a *policy implementation*, or *PI* for short. Note that since a channel is directional, a PI is directional too. That means that to create a complete request-reply connection we need at least two PIs. More information on this subject is available in Section V.

We call a *PI set* a group of policy implementations that belong to the same node and have the same technology. For instance, a particular server supporting IPsec and SSH will have two PI sets, one for each protocol. We will assume that the policy implementations in the same PI set are ordered according to their priority.

Note that in order to perform our analysis, we will make use of other additional sources of information such as:

- network reachability data. 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);
- the supported security protocols (at various ISO/OSI levels). This is needed to guarantee that it is possible to establish the secure channels;
- the supported cryptographic algorithms. Depending on the installed security controls, some cipher-suites might not be available when actually deploying a PI.

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 will begin our discussion in a more informal way by reasoning on the simplified network scenario sketched in Fig. 1. The diagram shows a main corporate network (C) and two branch networks (namely A and B). The three networks are connected through the Internet and consist of a number of security gateways (denoted by the g letter) that mediate the communications between the servers (s_{c1} and s_{c2}) and the clients (indicated by the c symbol). The server s_{c1} hosts two services (web_1 and db), while s_{c2} hosts only one web service (web_2).

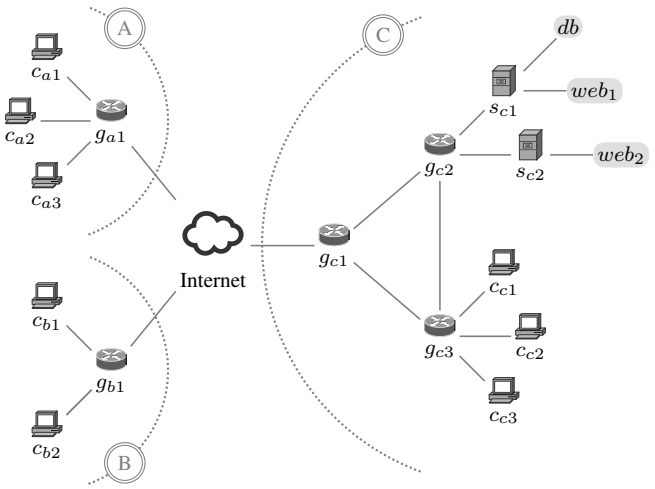


Fig. 1: A simplified network scenario.

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

For the sake of clarity, we grouped the anomalies in five macro-categories. They are sketched in Fig. 2 and they will be briefly described in the following paragraphs.

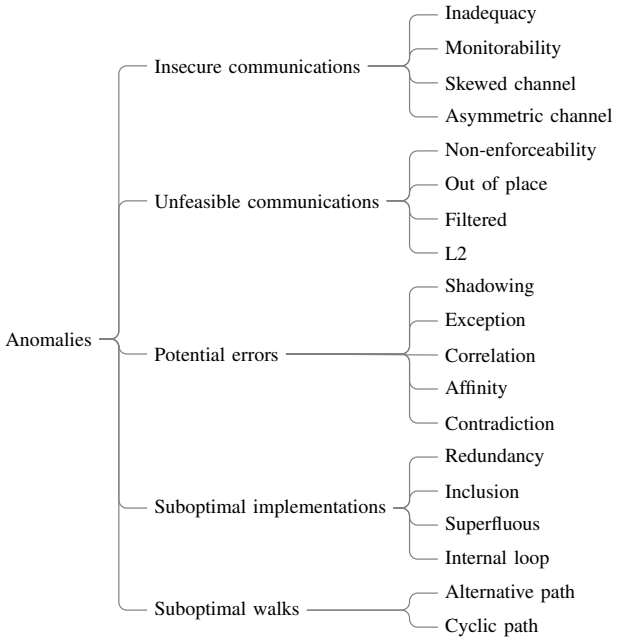


Fig. 2: Effect-based taxonomy.

A. Insecure communications

We have an *insecure communication* when its security level is less than the expected one.

For instance, we can have a channel that does not respect the minimum security levels specified in the corporate policies, thus creating an *inadequacy* anomaly. For example, we have an anomaly if the IT managers state that 'all the data crossing the Internet must be encrypted' and a security administrator creates the policy implementation $c_{a1} \xrightarrow[\{p\}]{\text{TLS}} s_{c1}$.

Another case of inadequacy arise when the security requirements are respected, but we have a communication consisting of more than one channel (e.g. remote access). Here, the nodes in the channel junctions can 'see' the exchanged data, thus lowering the security of the connection and creating a *monitorability* anomaly. For instance, the two PIs $s_{c1} \xrightarrow[\{c,p\}]{\text{IPsec}} g_{c1}$ and $g_{c1} \xrightarrow[\{c,p\}]{\text{IPsec}} c_{a1}$ form a logical connection between s_{c1} and c_{a1} by breaking it into two channels interconnected through g_{c1}. This means that, even if everything is encrypted, g_{c1} can sniff it since g_{c1} must re-encrypt the packet received from s_{c1}.

Another kind of insecure communication, more subtle but potentially catastrophic, can happen with a wrong tunnel overlapping that removes the confidentiality in a part of the communication and produces a *skewed channel* anomaly. For example, a security administrator can create the tunnel via the PI $g_{c3} \xrightarrow[\{c\}]{\text{IPsec}} g_{a1}$ 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.

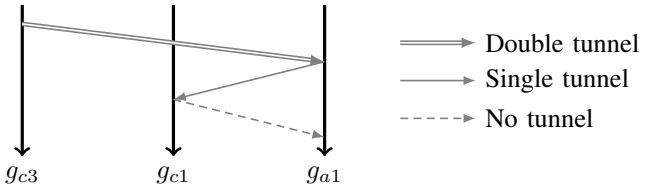


Fig. 3: Diagram of the skewed channel between g_{c3}, g_{c1}, g_{a1}.

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

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

B. Unfeasible communications

An *unfeasible communication* is a communication that cannot be established due to a hard misconfiguration. This anomalies are very severe since they completely hinder a data exchange.

The simplest example of an unfeasible communication is when the security administrators write a PI with a technology not supported by an end-point or a security level that is too high to be enforced by the available cipher-suites. We call this situation a *non-enforceability* anomaly. For instance, the PI $web_2 \xrightarrow[\{c\}]{\text{TLS}} db$ becomes non-enforceable if the service administrators did not install TLS on s_{c2} (where web_2 resides). This PI must be obviously deployed on the source endpoint s_{c2}, however, if the node containing it is not s_{c2} we have generated another problem, that is an *out of place* anomaly.

We have also a hindered connection if a channel is completely 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 the level 2 (data link) of the ISO/OSI stack. For example, if we choose to create a channel by using the WPA2 technology, we must be sure that the network frames does cross wireless-enabled nodes. If one or more crossed nodes are wired-only, then we have a *L2 anomaly*.

C. Potential errors

The *potential errors* are a class of anomalies where the original intent of the administrators is highly unclear. Hence their resolution require a full human inspection.

When working with a large group of PIs, it can happen that an administrator can create a PI that intercepts all the traffic for another one that has different security properties. For instance, the PI $c_{a1} \xrightarrow[\{p\}]{\text{TLS}} web_1$ hides $c_{a1} \xrightarrow[\{c\}]{\text{TLS}} web_1$, if the previous 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 used by administrators to express an 'all but one' rule, but we report them for a 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 data can match multiple PIs, hence making the intended protection level unclear. For example, $web_2 \xrightarrow[\{c\}]{\text{TLS}} s_{c1}$ and $s_{c2} \xrightarrow[\{c,p\}]{\text{TLS}} db$ are ambiguous since a packet from web_2 to db can match both the policy implementations. We call this problem a *correlation* anomaly. Analogously, we will have an *affinity* anomaly between two PIs that have different technologies but with the source and destination on the same node.

Finally, we have a *contradiction* anomaly when an administrator wants to protect a communication that should not be protected. For example, if the IT manager defines the policy 'all the traffic towards the Internet must be inspected', while the security manager enforces that the traffic exchanged between c_{a1} and s_{c1} must be encrypts (PI $c_{a1} \xrightarrow[\{c,p\}]{\text{IPsec}} s_{c1}$). This

leads a contradiction since the policy requires that the data for the Internet should be monitorable, but its implementation encrypts them.

D. Suboptimal implementations

A *suboptimal implementation* arises when one or more PIs can decrease the network throughput producing some overhead in the nodes. Their existence is usually not problematic, but their resolution can be beneficial since it improves the network performances and makes them less vulnerable to DDOS attacks.

The simplest kind of suboptimal implementation happens when an administrator deploys a PI that makes another PI useless, as the first one can secure the communication at the same or higher level of the second, but with a more effective protection (e.g., longer encryption key) . For example, two different security administrators may have independently defined the PIs $c_{a1} \xrightarrow[\{c\}]{\text{TLS}} web_1$ and $c_{a1} \xrightarrow[\{c,p\}]{\text{IPsec}} s_{c1}$. The first one is included in the latter, so that it can be safely removed. In these cases we have a *redundancy* anomaly if both the PIs have the same technology or an *inclusion* anomaly if they have two different protocols.

Another type of suboptimality can arise when a tunnel encapsulates other tunnels with a higher level of security. This problem, a *superfluous* anomaly, 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 unnecessary long.

In large networks, a communication between two end-points can take multiple paths, thus generating an *alternative path* anomaly. These are not necessarily a misconfiguration, but we can detect and report them to the administrators as a safety measure. For example, the PI $g_{c2} \xrightarrow[\{c\}]{\text{IPsec}} g_{c3}$ forms an alternative path w.r.t. the PIs $g_{c2} \xrightarrow[\{c\}]{\text{IPsec}} g_{c1}$ and $g_{c1} \xrightarrow[\{c\}]{\text{IPsec}} g_{c3}$.

Another cause of suboptimality happens when some data crosses a node multiple times during its transit. This *cyclic path* anomaly can be removed by removing the cycles, thus shortening the network path.

V. PI HIERARCHICAL STRUCTURE

In this section, we will formally define what is a policy implementation, its structure and the relationships between the various network fields that compose it. In addition, we will also describe the notion of path, used to detect several kind of network anomalies.

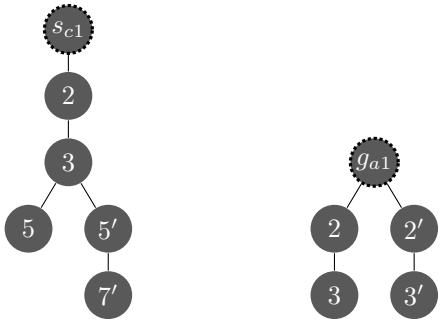
In our model, a PI i is a tuple:

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

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 and G are respectively a list of network nodes used in tunnels and the list of the gateways crossed by the traffic (Section V-D).

A. Sources (s) and destinations (d)



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

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

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

The root represents the network node, while all the other remaining nodes model the available network entities in the ISO/OSI stack. In this paper we will focus our attention only on the data link, network, session and application layers. The tree levels may be also associated respectively to the layer 2 addresses, IP addresses, port numbers and URIs. To avoid ambiguity we will use the notation $s_{c1}.5'$ to specify the node labeled $5'$ in the s_{c1} tree and so on.

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

Given any two network entities e_1 and e_2, we can have 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 pass through e_1. Graphically that means that 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 ISO/OSI layers. For instance in Fig. 4a, $s_{c1}.3$ dominates $s_{c1}.7'$;

- 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 relationships amongst them. For example in Fig. 4a, $s_{c1}.5$ is a kin of $s_{c1}.5'$;
- 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 will 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 will also make use of the special NULL technology, indicating that a communication should be created without any kind of protection.

Similarly 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 the ISO/OSI layer of t_1 is strictly less than the t_2's one. 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 they work at the same ISO/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:

$$t^{(i)} \sim t'^{(i)}, \qquad t^{(2)} \succ t^{(3)} \succ t^{(5)} \succ t^{(7)}$$
$$t \perp \text{NULL} \quad \forall t \neq \text{NULL}$$

Where $t^{(i)}$ and $t'^{(i)}$ represent two different technologies at the ISO/OSI level i and t is a generic security technology different from NULL.

C. Security coefficients (C)

The set of security coefficients consist 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 will 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 lest one coefficients of C_1 are strictly greater than their C_2's counterparts and the other coefficient of C_1 are not minor than their C_2's counterparts;
- C_1 is *disjoint* with C_2 ($C_1 \perp C_2$) if there is nor dominance nor equivalence relationships between C_1 and C_2, that is $C_1 \not\succeq C_2 \wedge C_1 \not\preceq C_2$.

D. Protected nodes (S) and crossed gateways (G)

For non end-to-end channels, a policy implementation contains a non-empty list S of network entities (usually a sequence of IP addresses) representing the node interfaces whose traffic must be protect when encapsulated into a tunnel. On the other hand, in end-to-end channels this field is empty since there are no tunnels.

In addition, each PIs contains an ordered set G that specifies the list of gateway nodes 'potentially' crossed by the channel traffic without taking into account the anomalies' side effects. This data is statically retrieved by knowing the network topology and the content of the various routing tables. Note that the list of crossed gateways does not contain the channel source and destination nodes. In the following sections, we will use the notation G^* to indicate a list containing 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 $\overline{G}$.

For instance, a site-to-site connection from c_{a1} to s_{c1} that makes use of IPsec in tunnel mode between the two gateways g_{a1} and g_{c2} is implemented by the two PIs:

$$i_1 = (c_{a1}, s_{c1}, NULL, \langle 0,0,0\rangle, \varnothing, \langle g_{a1}, g_{c1}, g_{c2}\rangle)$$
$$i_2 = (g_{a1}, g_{c2}, IPsec, \langle 3,3,3\rangle, \langle c_{a1}\rangle, \langle g_{c1}\rangle)$$

Where i_1 specifies the data encapsulated in the tunnel defined by the PI i_2.

E. Paths

We will now introduce the concept of path, which is strictly related to the notion of policy implementation. The notation P^{e_1,e_n} represents a possible path starting from the network entity e_1 and terminating into the network entity e_n. Each path is a list of policy implementations $(i_1, i_2, \ldots, i_n)$ 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} can be implemented by the three PIs:

$$i_1 = (c_{c2}, g_{c3}, NULL, \langle 0,0,0\rangle, \varnothing, \varnothing)$$
$$i_2 = (g_{c3}, g_{c2}, IPsec, \langle 3,3,3\rangle, \langle c_{c1}, c_{c2}, c_{c3}\rangle, \varnothing)$$
$$i_3 = (g_{c2}, s_{c2}, NULL, \langle 0,0,0\rangle, \varnothing, \varnothing)$$

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.

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 (see Fig. 2), which emphasizes the side effects of an anomaly. In the following paragraphs, however we will use a more technical oriented classification (see 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 is possible to generate an anomaly. We have distinguished 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 laying on the same node; 3) the *network level anomalies* arise between distinct PIs that belong to different nodes.

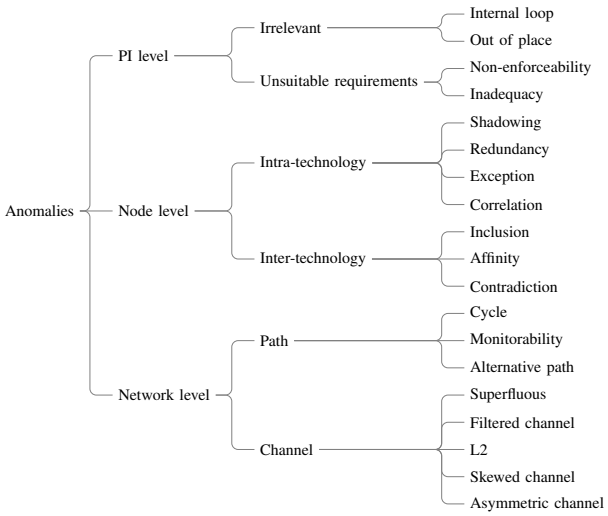


Fig. 5: Information-centric taxonomy.

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* (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 prerequisite and they can lead to very 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 data loop. These anomalies can be inferred using the formula:

$$s_1 \not\perp d_1$$

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 return the node where the PI is actually deployed. The formula is then:

$$\mathcal{N}(i_1) \perp s_1$$

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 neither the source nor the destination or when its security coefficients are 'too secure', and hence cannot be enforced.

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

These anomalies can be identified with the formula:

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

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.

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 some sort of metric or best practice [8], [9], [10] For example a network administrator could define a function such as:

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

We can detect these anomalies with the rule:

$$C_1 \prec \mathcal{C}_{min}(i_1)$$

To fix these anomalies 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 anomalies), otherwise we have an *inter-technology anomaly* (inclusion, affinity and contradiction anomalies). The intra-technology anomaly category has been heavily inspired by the work of Al-shaer et al. [6].

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 ($s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge S_1 \supseteq S_2$) and has disjoint security coefficients. We can detect these anomalies using the formula:

$$\pi(i_1) < \pi(i_2) \wedge t_1 = t_2 \wedge s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge S_1 \supseteq S_2 \wedge C_1 \perp C_2 \wedge G_1 = G_2$$

In order to resolve these kind of anomalies, the two PIs should be replaced by another PI i_3 that is a sort of 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 $C_2 = \{c_{2,i}\}_i$;
- $t_3 = t_1 = t_2$, $G_3 = G_1 = G_2$ and $S_3 = S_1 \cup S_2$.

To maintain the semantics of the system, the new PI i_3 should be inserted with the same priority of i_1 (that is $\pi(i_1)$).

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 has equal or dominant security coefficients. The following formula can be used to infer these problems:

$$t_1 = t_2 \wedge s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge S_1 \supseteq S_2 \wedge C_1 \succeq C_2 \wedge G_1 = G_2$$

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

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 ($s_1 \prec s_2 \wedge d_1 \prec d_2 \wedge S_1 \supset S_2$) . The relative detection formula is:

$$\pi(i_1) \prec \pi(i_2) \wedge t_1 = t_2 \wedge s_1 \prec s_2 \wedge d_1 \prec d_2 \wedge S_1 \supset S_2 \wedge C_1 \perp C_2 \wedge G_1 = G_2$$

Exceptions are very similar to the shadowing anomalies, and in fact they share the same resolution technique.

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 shadowing, redundancy or exception between the policy implementations. We can detect these anomalies via the formula:

$$s_1 \not\perp s_2 \wedge d_1 \not\perp d_2 \wedge t_1 = t_2 \wedge G_1 = G_2 \wedge \neg\mathcal{A}_{sh}(i_1, i_2) \wedge \neg\mathcal{A}_{ex}(i_1, i_2) \wedge \neg\mathcal{A}_{re}(i_1, i_2)$$

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, but one that is strictly dominant. We can detect these anomalies with the formula:

$$s_1 \succeq s_2 \wedge d_1 \succeq d_2 \wedge t_1 \succeq t_2 \wedge C_1 \succeq C_2 \wedge S_1 \supseteq S_2 \wedge G_1 = G_2 \wedge i_1 \neq i_2$$

The simplest way to resolve these anomalies is to delete i_2 (the 'innermost' PI). However, an administrator can also choose to keep both the policy implementations, following 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:

$$(s_1 \not\perp s_2 \wedge d_1 \not\perp d_2 \wedge t_1 \not\perp t_2) \wedge \neg\mathcal{A}_{in}(i_1, i_2) \wedge \neg\mathcal{A}_{inc}(i_2, i_1)$$

To resolve this type of anomaly, the two PIs should be replaced with a new PI i_3 that is a sort of 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 $C_2 = \{c_{2,i}\}_i$;
- $S_3 = S_1 \cup S_2$ and $G_3 = G_1 = G_2$.

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:

$$s_1 \not\perp s_2 \wedge d_1 \not\perp d_2 \wedge t_1 \perp t_2$$

Due to the high ambiguity of the situation (we cannot distinguish between a 'protect' and a 'do not protect' requirement), the resolution is the removal of one policy implementation, for instance, the one using the NULL technology.

C. Network level anomaly

The network level anomalies occur between distinct PIs that belong to different nodes. We can split these anomalies in 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 ($S_1 \neq \varnothing$) and all the policy implementations transported in this tunnel ($\forall\ i_k | s_k \in S_1 \wedge G_k^* \supset G_1^*$) have a security level greater than i_1. This anomalous PIs can be detected using the formula:

$$S_1 \neq \varnothing \wedge \{\forall\ i_k | (s_k \in S_1 \wedge G_k^* \supset G_1^*) \Rightarrow (C_k \succeq C_1)\} \neq \varnothing$$

Since all data transported in the tunnel are better protected than the tunnel itself, the obvious resolution is to delete i_1 (since it is redundant). However, as in the inclusion anomaly, an administrator could choose to keep the PI to 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 send without any form of encryption (see Fig. 3). We can detect these anomalies with the formula:

$$s_1 \in S_2 \wedge (|G_1^* \cap G_2^*|) \geqslant 2 \wedge (G_2^* \setminus G_1^* \neq 0) \wedge c_1^c > 0 \wedge c_2^c > 0$$

In order to resolve this kind of anomalies, it is needed to split the two PIs in three (or more) not overlapping policy implementations.

3) Filtered channel – $\mathcal{A}_{fi}(i_1)$: A PI i_1 is *filtered* when 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:

$$\exists e : e \in G_1 \wedge \mathcal{F}_e(i_1) = true$$

In practice, the output of the function $\mathcal{F}_e(i_1)$ can be populated either by means of a network reachability analysis [11], [12] or by using some firewall policy queries [13].

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- $\mathcal{A}_{L2}(i_1)$: We have a *L2 anomaly* when a PI that makes use of 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 making use of a function $\mathcal{T}^{(2)}(e)$ that returns the set of technologies at layer two supported by the node e. We can then write the formula:

$$\exists e : e \in G_1^* \wedge t_1 \notin \mathcal{T}^{(2)}(e)$$

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 using the formula:

$$\nexists i_2 : s_1 \not\perp d_2 \wedge d_1 \not\perp s_2 \wedge t_1 = t_2 \wedge C_1 = C_2 \wedge G_1 = \overline{G_2}$$

The simplest way to resolve these anomalies is to make sure that both the two PIs have the same security coefficients.

6) Cyclic path – $\mathcal{A}_{cy}(P^{e_1,e_2})$: There is a *cyclic path* anomaly between two network nodes e_1 and e_2 if there is at least one cycle in the path connecting them. In literature several algorithms have been proposed to perform the cycle detection in graphs in a very efficient way [14].

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:

$$\nexists P^{e_1,e_2} : (|P^{e_1,e_2}| = 1 \wedge i_j \in P^{e_1,e_2} : c_j^c > 0)$$

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 o more different paths that can be taken from the first node to the last one. This anomalies can be easily found by making use of the formula:

$$\exists P_j^{e_1,e_2}, P_k^{e_1,e_2}, j \neq k$$

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 with 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. It is evident that logical formulas are not easily usable by administrators. In Section V we already sketched our hierarchical view of a network node. By means of this view, we can depict 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, intuitively identify the anomalies, and immediately see the consequences and the proper 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 both payload and header integrity. The graph clearly shows that the two PIs are correlated as there are two "parallel" arrows. Our tool also explains these PIs are affine since they share some common aspects, but none of them includes the other one. An administrastor 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 ha adds confidentiality, or keep both (provided he adds at least payload integrity to the IPsec channel that is independently highlighted as Inadequacy anomaly).

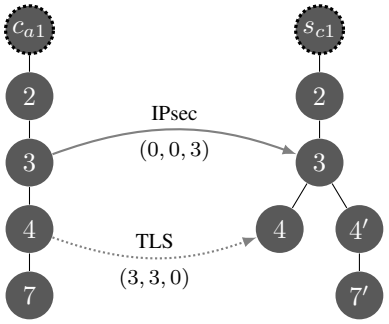


Fig. 6: Graphical representation of an affinity anomaly.

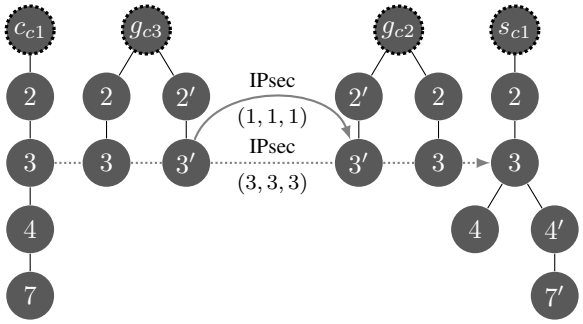


Fig. 7: Graphical representation of a superfluous anomaly.

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 superior 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 anomaly type have a corresponding graphical representation. Indeed, graphical anomaly representations can be built by including the network node trees corresponding to the communication end-points (see Fig. 4), that is, the source and the destination of the PI. Then, on one hand, policy implementations that enforce end-to-end channels are represented as a single directed edges that connect together 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 both the technology and the security coefficients required by the PI.

On the other hand, to represent policy implementations that enforce site-to-site and remote-access communications we add all the the network node trees corresponding to the crossed gateways and a set of edges that connect all the communication parties. For instance, in case of a tunnel (see Fig. 7), we introduce five edges: one undirected edge between the source node and the first gateway, one undirected between the first and the second gateway, one directed between the second gateway and the destination node, and two undirected edges to connect the communication interfaces of the gateways. Graphically, the resulting effect is a single arrow that connects source to destination. To tell them from end-to-end communications, we use dashed lines for site-to-site communications. In this case, the label is added to the only protected communication, that is, the one between the gateways.

As anticipated, the only anomaly that we did not represent graphically is the out of place anomaly type. The increase of complexity to the graphical representation (i.e. associate addresses, possibly ports and other parameters to the nodes) to visualize this anomaly type was not worth the increase of practical usefulness. Indeed, out of place anomalies are easily understood.

VIII. Model validation

In this section, we present an evaluation of the suitability of the our anomaly analysis model.

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?

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 literature, namely the shadowing, redundancy, exception, correlation and the skewed channel (overlapping sessions) and out of place (irrelevances) whose importance was already proved in other original works [6], [15], [16]. Moreover, we designed the experiment to be completed by expert administrators in one hour, therefore we avoided to provide data link information and kept the size of the network reasonably low. Therefore it was not possible to generate L2 anomalies, asymmetric channel, cycle and alternative path.

If our first research question ("the anomalies listed in this paper arise when enforcing the CPPs") were confirmed we could deduce that performing the detection can help improving the policy enforcement correctness in real world networks.

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 expertise), each one containing 10 people. In the test, we have considered as high-level expert administrators who have more than two years of experience in the security field, as mid level experts the administrators with more than two years of practice in the (non-security) network field and as low level experts 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 policy implementations. The landscape was a small network (composed of 5 subnets, 6 servers, 9 clients, 10 gateways and 76 PI sets). 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 the PIs where all the their fields where constrained to valid values

Experience	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 have done at least one anomaly in a macro-category.

Experience	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 have done at least one anomaly.

Experience	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 III: Percentages of anomalies introduced by the administrators.

Experience	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 IV: Percentages of anomalies introduced by the administrators grouped in macro-categories.

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

The second 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 (see Table IV). 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 (see Table III). On the other hand, the suboptimal implementations tend to increase because the expert administrators add more superfluous anomalies then inexpert ones. This is due because highly experienced administrators tend to add several level of protections in the communications (defense in depth), although this was not expressly required in the exercise. Moreover, the expert administrators' PIs also contains several monitorability anomalies, since they tend to make an extensive use of tunnels while the less experienced ones mainly use end-to-end channels. In short, they tend to break secure communications to improve the overall network performances. In this sample network the monitorability anomalies are not a particular serious issue (as we had homogeneous security levels in all networks), however, it is certainly better to double check them. Finally, there are a number of internal loop anomalies, more likely due to mere distraction errors, that are constant regardless of the expertise level.

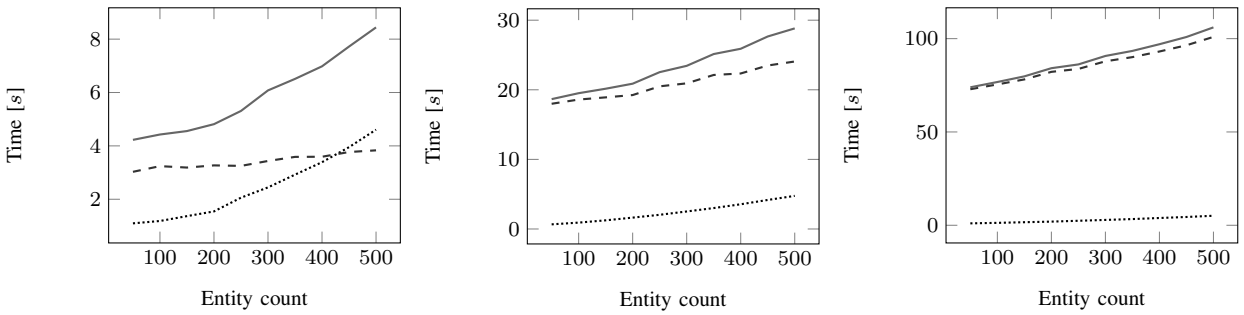
B. Complexity analysis

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

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 $\mathcal{C} \geq \mathcal{I}$).

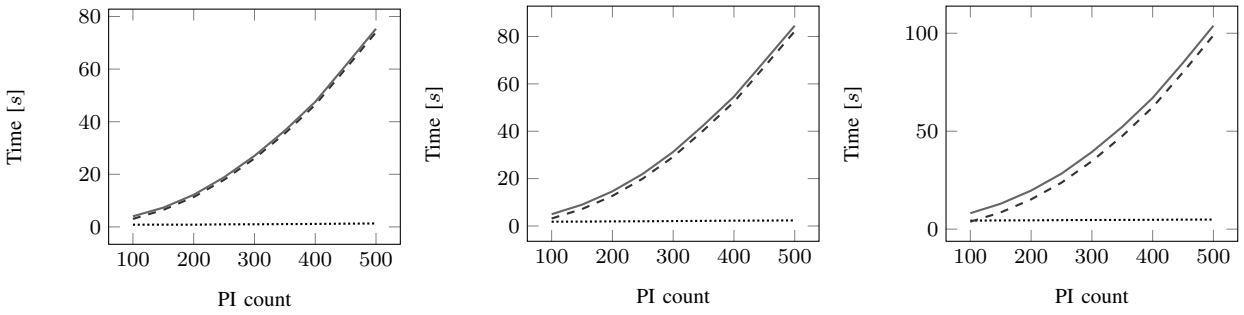
We will start by taking a look at the pre-computation phase. To create the tree representation of the network nodes, we need to check every single entity, so that this process has an exact complexity of $\mathcal{E}$. Finding all the simple paths[3] in an acyclic

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



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

Fig. 8: Performance tests with a fixed number of PIs.



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

Fig. 9: Performance tests with a fixed number of network entities.

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 achieving 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, making them with a complexity of $\mathcal{I}(\mathcal{I}-1)/2$;
- the monitorability and alternative path anomalies algorithms work on every path, hence they have a complexity of $\mathcal{P}$;
- the cyclic path anomaly algorithm can be efficiently performed using a proper cycle detection algorithm such as [14], that has a complexity of $O(\mathcal{E}+\mathcal{C})$. Note that its complexity is not necessarily $\mathcal{P}$ since a graph with some loops has 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 exploiting the natural graph-based representation 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 GiB 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 (see Table IV).

We have performed two main kind of tests. In the first one we kept fixed the number of network entities and increased the number of PIs, while in the second one we did the reverse (kept 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. For each test we kept track of three times: the pre-computation phase (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). In addition, the results are aligned with the complexity analysis discuessed in Section VIII-B. For instance, by increasing the number of network entities (Fig. 8), we may observe that the times tend to grow up, while by fixing the number of entities (Fig. 9) the pre-computation phase time is completely unaffected.

IX. Related works

Anomaly analysis, detection and resolution in policy-based systems and security controls are hot topics in the modern research. The current literature proves this by having several notable works, which we will briefly discuss in the following paragraphs.

A. Communication protection policies

The current literature contains several works about the anomaly detection between communication protection policies. Apparently the research in this area is solely focused on the IPsec technology.

An approach introduced by Zao in [17] is to combine conditions that belong to different IPsec fields. This was the basis used also in [18], where Fu et al. described a number of conflicts between IPsec tunnels, through a simulation process that reports any violation of the security requirements. In their analysis, the policy anomalies are identified by verifying the IPsec configurations against the desired policies written in a natural language. In practice, there is an anomaly when the policy implementations do not satisfy the requirements of the desired policies. In addition, Fu et al. proposed a resolution process that tries to find alternative configurations in order to satisfy the desired policy.

Al-Shaer et al. analyzed the effects of the IPsec rules on the protection of networks [6], by proposing a number of ad-hoc algorithms and formulas to detect these problems. Al-Shaer formalized the classification scheme of [18] 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 in [15].

Niksefat and Sabaei, in [19], presented an improved version of Al-Shaer's solution proposed in [6]. The two main improvements over the Al-Shaer's work are a new, faster detection algorithm and the possibility to resolve the detected anomalies.

Another interesting paper is [20], due to Li et al., where the authors classified the IPsec rules in two classes: access control lists (ACL) and encryption lists (EL).

B. Filtering policies

Network configuration/policy anomaly detection is not only restricted to communication protection technologies. In literature a quite rich collection of papers about filtering policy analysis is also available. Although these works are not directly related to the approach presented in this paper, they can be very useful as a general background on network anomaly analysis. In the following paragraphs we present a brief selection of several relevant works in this area.

One of the most influential works in this area is due to Al-Shaer et al., which addresses the analysis of filtering configurations in [21] via a set of FOL formulas. 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., in [22], focused on detecting and removing redundant filtering rules. The authors categorized these rules into upward redundant rules and downward redundant rules. The first ones are rules that are never matched, whether the downward redundant ones are rules that are matched, but enforce the same action as some other rules with a lower priority. The presented model is based on a data-structure named FDD (Firewall Decision Diagram).

Basile et al. describe a geometric representation, detection and resolution of filtering configurations, based on the intersection of hyper-rectangles in [23]. 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. in [24] suggested to split the classic five-tuple decision space of packet filtering rules into disjoint hyper-rectangles, where the conflicts are resolved using a combination of automatic strategies.

A thoroughly different approach for detecting conflicts between a set of filtering configurations is proposed again by Hu et al. in[25] by making used of an ontology-based anomaly management framework. Similarly, Bandara et al. proposed the use of logic reasoning, obtaining excellent performances [26].

Alfaro et al. presented a collection of algorithms to remove a series of anomalies between packet filter configurations and NIDS in distributed systems [16]. These techniques were more recently implemented in the MIRAGE tool[27].

X. Conclusions and future work

In this paper we proposed a list of nineteen anomalies (grouped in two taxonomies) that can arise during the implementation of communication protection policies and 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 between policy implementations that use security technologies working at different ISO/OSI layers and with different security properties. Most of the anomalies can be represented graphically with a simple yet natural graph representation, thus providing the administrators with a more intuitive way to visually identify the anomalies.

We implemented our model in Java by making an extensive use of ontological techniques, and proved on several network scenarios that it is scalable and performs well.

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

ACKNOWLEDGMENT

The research described in this paper is part of the SECURED project, co-funded by the European Commission (FP7 grant agreement no. 611458). The authors gratefully thank Dr. Marco Torchiano for his feedback on the empirical study described in this paper.

REFERENCES

[1] A. Wool, "Trends in firewall configuration errors: Measuring the holes in swiss cheese," *IEEE Internet Comput.*, vol. 14, no. 4, pp. 58–65, July-August 2010.

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

[3] The SANS Institute , "Consensus roadmap for defeating distributed denial of service attacks," Tech. Rep., 2000.

[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] European Commission, "Directive 95/46/ec of the european parliament and of the council on the adequacy of the protection provided by the safe harbour privacy principles and related frequently asked questions issued by the us department of commerce," Tech. Rep., 2000. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32000D0520&from=EN

[6] H. Hamed, E. Al-Shaer, and W. Marrero, "Modeling and verification of IPsec and vpn security policies," in *13th IEEE International Conference on Network Protocols*, ser. ICNP '05. IEEE Computer Society, Nov. 2005, pp. 259–278.

[7] C. Basile, D. Canavese, A. Lioy, and F. Valenza, "Inter-technology conflict analysis for communication protection policies," in *Risks and Security of Internet and Systems*. Springer, 2014, pp. 148–163.

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

[9] National Institute of Standard and Technology , "Recommendations of the national institute of standards and technology," Tech. Rep. [Online]. Available: http://csrc.nist.gov/publications/PubsTC.html

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

[11] 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, February 2009.

[12] 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.

[13] A. X. Liu and M. G. Gouda, "Firewall policy queries," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 6, pp. 766–777, 2009.

[14] R. Tarjan, "Depth first search and linear graph algorithms," *SIAM Journal on Computing*, 1972.

[15] 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.

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

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

[18] 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*, ser. POLICY'01. Springer-Verlag, January 2001, pp. 39–56.

[19] S. Niksefat and M. Sabaei, "Efficient Algorithms for Dynamic Detection and Resolution of IPSec/VPN Security Policy Conflicts," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, WA, April 20–23 2010, pp. 737–744.

[20] 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*, ser. FCST '06. IEEE Computer Society, Nov. 2006, pp. 83– 88.

[21] 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, Sep. 2006.

[22] A. X. Liu and M. G. Gouda, "Complete Redundancy Detection in Firewalls," in *IFIP WG 11.3 : 19th Working Conference on Data and Applications Security*, Storrs, CT, August 7-10 2005, pp. 193–206.

[23] C. Basile, A. Cappadonia, and A. Lioy, "Geometric interpretation of policy specification," in *Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on*. IEEE, 2008, pp. 78–81.

[24] 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.

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

[26] A. K. Bandara, A. C. Kakas, E. C. Lupu, and A. Russo, "Using argumentation logic for firewall configuration management." in *Integrated Network Management-Workshops, 2009*, ser. IM'09. IEEE Computer Society, Jun 2009, pp. 180–187.

[27] 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 International Workshop on Data Privacy Management, and 3rd International Conference on Autonomous Spontaneous Security*, ser. DPM'10/SETOP'10. Springer-Verlag, set 2011, pp. 203–215.

APPENDIX

Table V lists all the mathematical notations and symbols used in this paper.

Network entities	Technologies	Security coefficients	
$e_1 = e_2$	$t_1 = t_2$	$C_1 = C_2$	Equivalence
$e_1 \succ e_2$	$t_1 \succ t_2$	$C_1 \succ C_2$	Dominance
$e_1 \sim e_2$	$t_1 \sim t_2$		Kinship
$e_1 \perp e_2$	$t_1 \perp t_2$	$C_1 \perp C_2$	Disjointness
Crossed gateways			
G^*			Crossed gateways with end-points
$\overline{G}$			Crossed gateways in reverse order
Paths			
$P^{a,b}$			A path from a to b
Auxiliary functions			
$\mathcal{N}(i)$			Node where the PI i is deployed
$\mathcal{T}(e)$			Technologies supported by node e
$\mathcal{C}_{max}(i)$			Maximum coefficients supported by the PI i
$\mathcal{C}_{min}(i)$			Minimum coefficients acceptable for the PI i
$\pi(i)$			Priority of the PI i
$\mathcal{F}_e(i)$			Filtering of node e for the PI i
$\mathcal{T}_e^{(2)}(i)$			Data-link incompatibility for the node e and the PI i

TABLE V: Mathematical notations.