# Risk-aware framework for activating and deactivating policy-based response

Wael Kanoun, Nora Cuppens-Bouhlahia, Frédéric Cuppens, Samuel Dubus

HAL Id: hal-00540839
https://hal.science/hal-00540839

Submitted on 29 Nov 2010

# Risk-aware Framework for Activating and Deactivating Policy-based Response

Wael Kanoun[*†], Nora Cuppens-Boulahia[†], Frédéric Cuppens[†] and Samuel Dubus[*]

[*]Bell Laboratories, Alcatel-Lucent

[†]Telecom Bretagne, Institut Telecom

*Abstract*—**With the growth of modern systems and infrastructures, automated and intelligent response systems become the holy grail of the security community. An interesting approach proposes to use dynamic access control policies to specify response policies for such systems. These policies should be enforced when an ongoing attack, that threatens the monitored system, is detected. However, existing work do not present a clear methodology to specify the Response policies. In particular, the deactivation issue is not yet tackled. In this paper, we first present how to specify response policies. Second, a risk-aware framework is proposed to activate and deactivate response policies. Hence, the success likelihood of the threat, and the cumulative impact of both of the threat and the response, are all considered.**

*Index Terms*—**Response policy, threat and response contexts, activation/deactivation, cumulative impact, success likelihood.**

## I. INTRODUCTION

Information systems, critical services and infrastructure are continuously growing in size and complexity. Network operators are providing a wide range of services such as voice (VoIP) or television (TVoIP) based on these systems. Unfortunately, they remain always an attractive target for hackers and criminal organizations. The attacks may be accurately pointed, or performed at a large scale. Therefore, a proper and efficient response system becomes primordial, to insure the availability and the security of these systems and the provided services.

This interesting issue has been driving the research community to propose several intelligent response models. These response models aim to identify and select the most efficient countermeasure(s), in order to stop the detected attack or mitigate its impact. For instance [1] presents an adaptive intrusion response model that considers the intrusion's spread, using intrusion graphs. On the other hand, models that adopts a cost-benefit approach have been proposed: [2], [3], [4], [5], and [6] consider the cost (or the impact) of the detected attack and the candidate countermeasure(s). [7] and [8] consider service dependencies to calculate more accurately the impact of the attack and the candidate countermeasure(s). Moreover, [9] and [10] propose to consider the risk of the ongoing attack; in other words they combine the impact with the success likelihood of ongoing attacks to assess dynamically the risk(s).

On the other hand, a different approach has been proposed in [11]. The response, as presented in this paper, is based on activating and enforcing dynamic policies to counter an attack. We view this approach more global than the previous models above. While the previous models have a local and limited scope on the monitored system (i.e. tactical), the latter has a far larger scope and acts as 'strategic' response model. [12] and [13] show that the two approaches are complementary, and combine them in a single response workflow. Thus, response may be performed at two levels: intermediate level (i.e. tactical) and high level (i.e. strategic). The former is based upon launching single and local countermeasures, while the latter is policy-based and may affect the entire system.

However, several interesting issues have been raised. First, the aforementioned papers do not consider the deactivation of these policies during specification. Second, these papers propose to activate a strategic response upon receiving and correlating the alerts that correspond to executed attacks. However these alerts may represent ongoing attacks that have a minor (or null) chance to succeed, given the attack progress and the real-time state of the monitored system. Third, previous work raise the deactivation issue, but they do not indicate how and when the strategic (i.e. policy-based) response should be deactivated. For instance, [11] relies on experts to assign static lifetime for the response policies, which is not satisfactory. Moreover, during the deactivation process of a strategic response, the system (or the administrator) may encounter the following cases: (i) some actions once activated, cannot be (simply) deactivated, or (ii) the expert needs to specify some actions that have to be executed only when the response is deactivated, but not before (e.g. notify the users that the service is back to normal).

In this paper, we first present how to specify response policies for complex systems, while considering the deactivation process. Then, we propose an online model with an associated architecture, which activates and deactivates response policies (i.e. strategic responses), using a risk-aware approach: the success likelihood of the ongoing threat/attack, and the cumulative impacts (i.e. cost) of the threat and the response, are all considered before activating/deactivating strategic response.

This paper is structured as follows. Section II summarizes the state of the art of policy models, while Section III presents how response policies may be controlled, by proposing a response taxonomy. Section IV shows how to specify threat and response contexts. Section V defines a risk-aware framework to activate and deactivate threat contexts and response contexts. Section VI proposes an architecture to implement the activation and deactivation framework. Section VII presents a VoIP case study to illustrate our work.

## II. Security and Response Policy Models

To specify security and response requirements, several policy-based models may be used (e.g. RBAC [14], Ponder [15] and OrBAC [16]). However, RBAC and Ponder suffer from several drawbacks. First, they offer only a single dimension of abstraction, using *roles* assigned to *subjects*. Such abstraction is not offered for other entities (i.e. *objects* and *actions*). Obviously, this limitation will multiply significantly the number of security rules for a given policy. Second, RBAC does not consider the concept of *organization*. Therefore, the specification and the management of security policies for large and multi-regional systems, become tedious tasks. Finally, the dynamic nature of the response cannot be modeled with RBAC. For example, once an ongoing attack has been detected, a set of rules should be activated and enforced.

On the other hand, OrBAC does not suffer from the aforementioned limitations. It is expressive enough to specify a large variety of security requirements. In fact, it has been successfully applied to specify network access control policies. Moreover, translation mechanisms were defined to automatically generate firewall configuration rules that are free of inconsistency and redundancy [17]. Thus, we will use OrBAC in the remainder of this paper. We present below a brief description of the OrBAC Model. For more details and formal description, interested readers may refer to [16] and [18].

### A. The OrBAC Model

OrBAC is a security policy model centered on the concept of organization. Using OrBAC, security policies are specified at an abstract level. A concrete level policy may be inferred to enforce them in the monitored system.

Therefore, one of the OrBAC contributions is the abstraction of the traditional triples (*subject, action, object*) into (*role, activity, view*) . The entities *subject*, *action* and *object* are called concrete entities whereas the entities *role*, *activity* and *view* are called organizational (i.e. abstract) entities. A *view* is a set of *objects* that possess the same security-related properties within an organization. Thus, these objects are accessed in the same way. Abstracting them into a *view* avoids the need to write one rule for each of them. Another useful abstraction is that of *action* into *activity*. An *activity* (e.g. call) is considered as an operation which is implemented by one of the *actions* available in the organization (e.g. SIP_call, NoE_call and H323_call)[1]. Therefore, they can be abstracted with a unique *activity* for which we may define a single security rule.

Another main contribution of the OrBAC model is the concept of *context*, that reduces the applicability of the rules to some specific circumstances [18]. Using OrBAC, the expert can specify contextual abstract security rules:

- *permission(org, R, A, V, Ctx)* means that in organization *org*, the role *R* is authorized to perform the activity *A*, on the view *V*, when the context *Ctx* is activated.
- *prohibition(org, R, A, V, Ctx)* means that in organization *org*, the role *R* is NOT authorized to perform the activity *A*, on the view *V*, when the context *Ctx* is activated.

[1]SIP, NoE and H.323 are signaling protocols used in VoIP systems

- *obligation(org, R, A, V, Ctx, Ctx$_v$)* means that in organization *org*, the role *R* MUST perform the activity *A*, on the view *V*. The obligation context *Ctx* represents the set of conditions for which, the obligation holds. The violation context *Ctx$_v$* represents the set of conditions for which, and while the obligation holds, the obligation is violated. Interested readers may refer to [19] for a complete description of obligations.

For example, *prohibition(vodaphone; normal_client, SIP_call, Internet, Response_SIP_DoS)* means that all normal clients of Vodaphone are not authorized to call using SIP protocol, if the SIP server was the target of a DoS attack.

Once the policy has been specified at the organizational level, it can be instantiated by assigning concrete entities to abstract entities and activating the needed context(s). Three predicates have been defined (*empower*, *consider* and *use*) to assign respectively a subject to a role, an action to an activity, and an object to a view:

- *empower(org, S, R)* specifies that in organization *org*, subject *S* is empowered in role *R*.
- *consider(org, A', A)* specifies that in organization *org*, action *A'* implements activity *A*.
- *use(org, O, V)* specifies that in organization *org*, object *O* is used in view *V*.

For example: *empower(vodaphone, 5322@vodaphone.com, normal_client)* means that 5322@vodaphone.com is a normal client in the operator's network.

## III. Usage of OrBAC for Policy-based Response

For any system, OrBAC may be used to specify the Operational Policy (OP), and the Response Policy (RP). The former is used to specify the security rules during 'normal' circumstances. The associated contexts might be for example geographic (e.g. *on_site*, or *outside*), temporal (e.g. *working_hours*, *holidays*), etc. On the other hand, the RP specifies the response rules that must be activated and enforced to counter each detected threat. Thus, each context in RP is specified to handle a potential threat. In the remainder of this paper, we will refer only to the RP. The expert has to associate the appropriate response rules to each response context, which in turn is associated to a threat context. A response context is activated only if the associated threat context has been already activated. However, in a monitored system, a threat could be detected (and therefore the associated context is activated), while the associated response context remains inactive due to other reasons (e.g. the strategic response cost exceeds that of the threat; or to avoid major conflicts because another strategic response has already been activated). This point will be clarified in the remainder of the paper.

In the first section of this paper, we mentioned the deactivation issue: how and when a response policy should be deactivated? It should be noticed that the deactivation of policy-based (i.e. strategic) responses is not a trivial task. In other words, the deactivation of a response policy (or more precisely a response context) does not simply consist of deactivating the associated security rules. This can be due to several reasons:
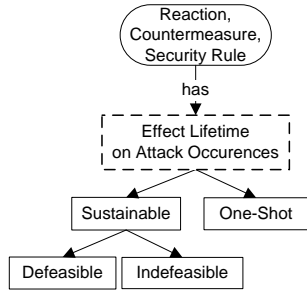
Fig. 1. Temporal Response Taxonomy.

- The expert needs to launch some actions only at the activation (or only at the deactivation) of a response policy (e.g. notify a user to change his password).
- Whether it is impossible, or it is considered a tedious task that may impact the OP (i.e. the expert's intervention is needed to end the response), some of the response rules or countermeasures cannot be deactivated (e.g. patch a software, patch an OS, remove permanently a client from the database).
- Some of the response rules or countermeasures have a very limited lifetime, and therefore are considered as automatically deactivated (e.g. close a malicious connection, restart a server, change a password).

Therefore, a simple yet efficient solution is to specify for each response context, two associated event-based contexts: *Start(response_context)*, and *End(response_context)*. They correspond respectively to the activation and the deactivation of *response context*. [20] propose an operational semantic to specify event-based contexts associated to state-based contexts using the ECA (Event-Condition-Action) formalism [21]. The formal definition of this operational semantic is not in the scope of this paper due to space limitation. Now, we will present first a taxonomy for reactions and countermeasures. This will allow us to classify them with respect to their lifetime, and whether their lifetime is controllable or not.

*A. Response Taxonomy*

There are many papers that present reaction and response taxonomies, like [22] and [23]. However, none of them mentioned the lifetime of the countermeasures, or whether they can be deactivated. We propose a taxonomy that considers the temporal dimension of the response actions. We may refer to this taxonomy (See Figure 1) to classify any reaction or countermeasure. Thus, it may be also used to classify any security rule expressed in the RP, w.r.t. their lifetime and the controllability of their lifetime.

First, considering their lifetime, countermeasures (or reactions) may be divided initially into two major classes: one-shot countermeasures and sustainable countermeasures.

*1) One-Shot countermeasures:* A reaction has a one-shot lifetime when its effectiveness is limited to a single attack occurrence. Once a one-shot reaction is launched, it is automatically deactivated. The effective lifetime of this class of reactions is null (or negligible). That means future attacks may occur, as if this reaction was not launched. Therefore,

a one-shot response has to be launched for each occurrence of the same attack. Examples of this class include: closing a malicious connection, restarting a server, notifying the administrator, etc. Let us consider a perfect system, where $A$ is a set of (the same) attack occurrences, and $CM$ the associated countermeasure occurrence. We define a relation $f : A \rightarrow CM$. This relation can be read as "an element from $A$ is ended by an element from $CM$". In a perfect system, each attack must be ended, and thus $f$ is an application:

*Hypothesis 1:* $\forall\, a \in A, \exists!\, cm \in CM \mid cm = f(a)$

For each occurrence $a$ of the attack $A$, a $cm$ is required to end the attack. Therefore a $cm_i$ that ended the attack occurrence $a_i$, has no effect on future attack $a_j$ $(j > i)$. One-shot countermeasure can be defined with the following proposition which demonstrates that $f$ is injective:

*Proposition 1:* $\forall\, a_i, a_j \in A;\; cm_i = f(a_i)$ and $cm_i = f(a_j) \Rightarrow a_i = a_j$

*2) Sustainable countermeasures:* A reaction has a sustainable lifetime when its effectiveness is not limited to a single attack occurrence. Once a sustainable reaction is activated, it remains active against future attack occurrences. In other words, its lifetime sustains for a period of time (which may be almost infinite). Therefore, the activation of a sustainable reaction may be effective against future attack occurrences, until this reaction is deactivated. Examples of this class include: patching software, blocking a machine, deleting an account, suspending an account, etc. Let us define the same function $f : A \rightarrow CM$. $f$ is not injective because a sustainable countermeasure $cm_i$ can end a sequence of attack occurrences $\{a_i, a_{i+1}, ...\}$ that occur during $cm_i$ lifetime. In other words, we can have several attack occurrences that are ended by the same countermeasure occurrence:

*Proposition 2:* $\exists\, a_i, a_j \in A \mid (cm_i = f(a_i))$ and $(cm_i = f(a_j))$ and $(a_i \neq a_j)$

Moreover, sustainable class may be divided into two sub-classes:

*a) Defeasible:* A defeasible reaction can be deactivated, and therefore its lifetime may be controlled by the administrator or the system. Examples of this class include: blocking a machine, suspending an account, dropping traffic etc.

*b) Indefeasible:* An indefeasible reaction cannot be deactivated, or the deactivation of such reaction requires exceptional effort or policy modification(s). Examples of this class include: deleting an account, patching an OS, patching a software, etc.

IV. POLICY-BASED RESPONSE SPECIFICATION

In [11], response contexts were associated with an alert signatures. This would lead inevitably toward a huge number of contexts to manage. Moreover, the activation of the context is triggered with the detection of an attack, which is surely not very effective considering the high number of failed attempts to attack a complex system. Second, the same attack may be carried out using different exploits and techniques. However, the response policy against all these attacks will be the same. Moreover, even if an attack action (or step in a larger scenario) has been detected, it may be useless to launch the associated

strategic response; because the attacker will not be able to achieve his goal due to other reasons (e.g. another essential attack action is not executable on the monitored system, another response is already activated). Thus a response policy may be activated, only if the associated threat context has been activated. The latter is activated if the success likelihood of the ongoing attack [24] exceeds a predefined threshold.

Now we show how to define the *threat contexts*, and the associated *response contexts*.

### A. Threat Specification

First, the expert identifies the threats that may violate the system's security and operation (using for instance a preliminary risk analysis). For example, a *partial or total DoS*, *SPIT* (Spam over IP Telephony) and *toll fraud* are identified as potential threats in a VoIP environment. For each threat, the expert creates an associated attack objective, in order to be used by alert correlation models based on attack graphs generation [25] [26] [27]. Usually, attack objectives are the terminal nodes in an attack graph, and they violate the security and operational policy of the monitored system. They can be formally defined, using one of the attack description languages (e.g. LAMBDA [27]). LAMBDA uses first-order logic predicates to model the system's state, adopting a pre/postcondition approach. For instance, DNS DoS may be caused if *host* is a DNS server, and *host* is the target of a DoS attack:
*-Precondition: dns_server(host), target(host)*
*-Postcondition: dns_dos(host)*

Finally the expert has to specify, for each attack objective, two event-based contexts:

- *Start(Threat_i)*: This context is triggered when an ongoing threat $i$ is detected with a high success likelihood to accomplish its attack objective.
- *End(Threat_i)*: This context is triggered when an ongoing threat $i$ disappears.

The triggering mechanisms of these two contexts will be defined in Section V-A.

### B. Response Specification

The expert specifies the response rules associated with each *Response_i*, that aims to counter and handle *Threat_i*. Similarly, *Response_i* will be activated and deactivated:

- *Start(Response_i)*: This context is activated when the administrator (or the system) decides to handle and counter *Threat_i*. In consequence, all the response rules associated to *Response_i* will be activated and deployed in the monitored system.
- *End(Response_i)*: This context is activated when the administrator (or the system) decides to cease the response. In consequence, all the response rules that were activated previously (via *Start(Response_i)*) will be deactivated.

The triggering mechanisms of these two contexts will be defined in Section V-B.

The expert may also add explicitly 'new' response rules that will be only activated at the end of the response. These

response rules should be associated to the $\neg Response_i$ context, to express the fact that they should be only enforced when *Response_i* is deactivated (thus when *End(Response_i)* is triggered). Let us suppose that $\alpha$ is a response context, and that $\beta = \neg\alpha$, thus:

$$\beta = \neg\alpha \Rightarrow \left\{ \begin{array}{l} Start(\beta) = End(\alpha) \\ End(\beta) = Start(\alpha) \end{array} \right. \quad (1)$$

As mentioned in Section II-A, there are three types of response rules that can be associated with any response context *Response_i* in the Response Policy (RP): *permissions*, *prohibitions* and *obligations*. When *End(Response_i)* is activated, all the *permissions* and *prohibitions* that were previously activated by *Start(Response_i)*, are deactivated. In other words, we do not need to specify explicitly new response rules that have an opposite 'effect' on the system [20]. However, the case of *obligations* need a deeper examination, and sometimes explicit response rule(s) must be explicitly specified.

In fact, for each *obligation* rule associated with *Response_i*, we have to check first the type of its *activity* or *action*, using the taxonomy presented in Section III-A. Therefore there are three types of *obligations*: (i) one-shot, (ii) defeasible, and (iii) indefeasible.

First, the deactivation process of defeasible obligations is possible. The associated obligations in the context $\neg Response_i$ can be expressed by another obligation *obligation(org, R, $\overline{A}$, V, ¬Response_i, Response_i)*; where $\overline{A}$ cancels the defeasible *activity* (e.g. block/allow traffic, activate/deactivate a backup server)

*Proposition 3: if obligation(org, R, A, V, Response_i, ¬Response_i) then obligation(org, R, $\overline{A}$, V, ¬Response_i, Response_i)*

We notice that *obligation(org, R, A, V, Response_i, ¬Response_i)* holds when *Start(Response_i)* has been activated, and the violation context of this obligation is $\neg Response_i$: this means that this obligation must be fulfilled before the end of response $i$ (i.e. before the triggering of *End(Response_i)*).

On the other hand, *obligation(org, R, $\overline{A}$, V, ¬Response_i, Response_i)* holds when $\neg Response_i$ is activated, while its violation context is *Response_i*: this means that this obligation must be fulfilled before another (future) start of response $i$.

Second, for one-shot obligations (e.g. kill shell, close connection, restart server), the deactivation is 'implicitly' performed, due to the limited lifetime of such obligations:

*Proposition 4: if obligation(org, R, A, V, Response_i, ¬Response_i) then no correspondent obligation has to be associated with ¬(Response_i)*

Finally for indefeasible obligations, no deactivation can be done (or it cannot be specified in the Security and Operational Policy). However the administrator must review (and potentially update) the system policy *POL* that contains the Operational Policy OP and the Response Policy RP, to take in consideration the permanent obligations that have been activated. For example, when a server is patched, the associated threat context may need to be updated. Another example, when a malicious employee *John* is fired, subject *John* is removed from the system. We consider that, so far, these updates can not be automated.

*Proposition 5: if obligation(org, R, A, V, Response$_i$, ¬Response$_i$)) then obligation(org, admin, review, POL, ¬Response$_i$, Response$_i$)*

## V. Context Triggering Framework

In this section, we present a model that handles the activation and deactivation of threat contexts and response contexts. Therefore, we determine for each predefined context, when it should be triggered. A risk-aware approach is adopted: we consider the impact and the Success Likelihood ($SL$) of the ongoing threat, and the impact of the associated response.

### A. Triggering Threat Contexts

The success likelihood ($SL$) of an attack objective $i$ can be calculated in real-time, considering the attack progress and the state of the system. Such model was proposed in [24]: the generated attack graph is first transformed into dynamic Markov Models. Then, the Mean Time to each Attack Objective (MTAO) is calculated using the markovian transition and sojourn matrices derived from the attack graph. Afterward, the success likelihood $SL$ of each attack objective $AO_i$ (and thus of each *Threat$_i$*) is calculated using a logarithmic scale (See [24] for more information):

$$SL_{Threat_i} = -20 \times \log_{10} \left( \frac{MTAO_{AO_i} - 1}{MTAO_{AO_i}} \right) \quad (2)$$

A high $SL$ means that the threat is close to achieve its objective. When the $SL$ of *Threat$_i$*, calculated in real-time, exceeds a predefined threshold, *Start(Threat$_i$)* context should be activated. The expert may tune this threshold (for example using machine learning techniques).

*Proposition 6:* If ($SL_{Threat_i} > \epsilon_{activ}$) then *Start(Threat$_i$)* is true

Every threat or response has an impact on the security properties of the system (i.e. confidentiality, integrity and availability). We rely on existing assessment models, which consider the assets and services dependency in the monitored system, to evaluate the impact of the attack and the response [7] and [8]. Therefore it is possible, for each attack objective, to calculate the variation of the confidentiality level $\delta C$, integrity level $\delta I$ and availability level $\delta A$ in the monitored system. For instance, [7] and [8] model first the monitored system (or service), as a set of resources $\mathcal{R}$. A resource may be a *service instance* or a *user*. The existing dependencies between all the resources in $\mathcal{R}$ may be specified using dependency matrices. All dependency types are considered (mandatory, alternative, combined, etc.). If one resource in the system is targeted by an attack, the direct impact on this resource will propagate to other resources through the existing dependencies (thus inducing an indirect impact). These models calculate, using Breadth First Search (BFS) algorithms, the indirect impact of a given attack on the entire system. Therefore the overall impact (which is the sum of direct and indirect impacts) on confidentiality $\delta C$, integrity $\delta I$ and availability $\delta A$ incurred by the entire system, can be calculated. Interested readers may refer to these paper for further details. Nevertheless, we may simply rely on experts' knowledge to assign, for each threat, static (i.e. *a priori*) $\delta C$,

$\delta I$ and $\delta A$. Thus, we may consider that $\delta C$, $\delta I$ and $\delta A$ for each threat, as input for our triggering framework.

The impact of an attack (or a response rule), is not constant: the longer the attack lasts (or the response is enforced), the higher is its impact. Therefore when a *Threat$_i$* is detected (thus *Start(Threat$_i$)* has been activated), its Cumulative Impact $CI$ starting from the activation time $T_{start\ threat}$, can be calculated at any time $t$:

$$CI_{Threat_i}(t) = \int_{T_{start\ threat}}^{t} (\delta C(u) + \delta I(u) + \delta A(u))\, du \quad (3)$$

On the other hand, *End(Threat$_i$)* should be activated if the $SL$ drops below another threshold $\epsilon_{deactiv}$. $\epsilon_{deactiv}$ may be equal to $\epsilon_{activ}$; however there is the risk of quick switching between the activation and deactivation when the $SL$ varies continually around $\epsilon_{activ}$. Therefore, it is advised to set $\epsilon_{deactiv}$ slightly lower than $\epsilon_{activ}$ (hysteresis effect): this guarantees a fixed delay between the activation and the deactivation.

*Proposition 7:* if [(Start(Threat$_i$) is true) and (SL$_{Threat_i}$ < $\epsilon_{deactiv}$)] then *End(Threat$_i$)* is true

### B. Triggering Response Contexts

In this Section, we define when a policy-based response should be activated or deactivated. Therefore, we have to specify, for each response context *Response$_i$* associated with *Threat$_i$*, when the two contexts *Start(Response$_i$)* and *End(Response$_i$)* should be activated.

First, a *Response$_i$* should be activated, when *Threat$_i$* is active and the associated impact exceeds a predefined impact threshold $\kappa_{min}$ that the expert may tune. A very low $\kappa_{min}$ will cause the activation of a response context even against minor attacks which may be handled by tactical response. On the other hand, a very high $\kappa_{min}$ will turn the strategic response useless or rarely applied. A good practice is to set $\kappa_{min}$ as the deployment cost of the response policy.

*Proposition 8:* if [(Start(Threat$_i$) is true) and (CI$_{Threat_i}$ > $\kappa_{min}$)] then *Start(Response$_i$) is true*

Now, we have to evaluate the policy-based response impact on the system. Given that a policy-based response is a set of response rules, we can assess the $\delta C$, $\delta I$ and $\delta A$ induced by each response rule, using one of the aforementioned impact assessment models (See Section V-A). Therefore, the $CI$ of a given security rule $u$, starting from the activation time $T_{start\ response}$, can be calculated at any time $t$:

$$CI_{SR_u}(t) = \int_{T_{start\ response}}^{t} (\delta C(u) + \delta I(u) + \delta A(u))\, du \quad (4)$$

For example, in a VoIP service environment, let us consider an ongoing attack that threatens the users' hardphones. A simple response may consists of two response rules:

- *prohibition(client, hardph_call, Internet, hardph_threat_resp)*
- *permission(client, softph_call, Internet, hardph_threat_resp)*

The first rule prohibits the clients from using the hardphone, and therefore induces a sharp fall in the availability level $\delta A_{rule1} < 0$, while there is no impact on the confidentiality or

integrity level ($\delta C_{rule1} = \delta I_{rule1} = 0$). The second rule permits the clients to use their softphone, therefore this response rule induces a raise in the availability level $\delta A_{rule2} > 0$. However, the second rule cannot fully compensate the first rule, because not all the clients have an installed softphone. It is obvious that the impact metrics $\delta C$, $\delta I$ and $\delta A$ are proportional to the deployed response rules, which is proportional to the number of subjects (e.g. clients) affected by the response. Moreover, $\delta C$, $\delta I$ and $\delta A$ depend on the type of the *subjects* (i.e. *roles*); for instance the cost incurred by a loss of confidentiality is not the same for a premium user and a normal user.

We define the overall Cumulative Impact at time $t$, for *Response$_i$*, as the summation of cumulative impact of the response rules deployed and enforced by *Start(Response$_i$)*:

$$CI_{Response_i}(t) = \sum_{SR_u \in Response_i} CI_{SR_u}(t) \qquad (5)$$

Finally, *End(Response$_i$)* should be launched, when the associated threat $i$ had vanished (or its *SL* is negligible) or the impact of response $i$ exceeds that of the threat $i$:

*Proposition 9:* if *(Start(Response$_i$) is true) and [(End(Threat$_i$) is true) or ($CI_{Response_i}(t) > CI_{Threat_i}(t)$)]* then *End(Response$_i$) is true*

## VI. PROPOSED ARCHITECTURE

We propose an online architecture, to activate and deactivate policy-based response using the presented model. Figure 2 illustrates the proposed architecture:
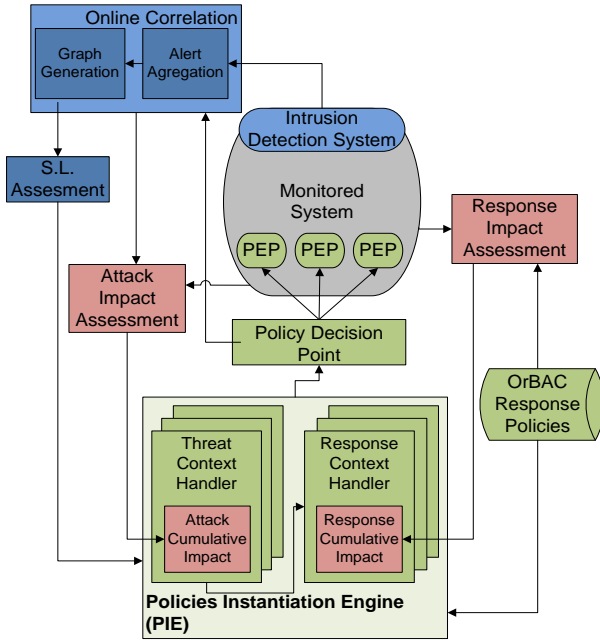


Fig. 2. Proposed Architecture to Activate and Deactivate Policy-based Response.

*a) Online Alert Correlation:* This module collects the generated alerts and correlates them, to construct the graph of ongoing threats and attacks.

*b) Success Likelihood Assessment:* This module evaluates in real-time the success likelihood of ongoing threats, using the attack graphs generated by the Alert Correlation module. For instance, the model presented in [24] can be used.

*c) OrBAC Security and Response Policy:* This database stores the response rules of the OP and RP, which are specified as presented in Section IV.

*d) Attack Impact Assessment:* This modules assesses the variations of confidentiality $\delta C$, integrity $\delta I$, and availability $\delta A$ levels induced by the attack. The values may be statically defined, or measured online. For instance, the models presented in [7] or [8] can be used.

*e) Response Impact Assessment:* Similarly to the previous module, this module assesses the variations of confidentiality $\delta C$, integrity $\delta I$, and availability $\delta A$ levels due to the activation of a response policy.

*f) Policy Instantiation Engine:* This module instantiates abstract response rules (specified with *roles*, *views* and *activities*) into concrete rules (with *subjects*, *objects* and *actions*).

*g) Threat Context Handler:* This module activates and deactivates the threat contexts (i.e. *Start(Threat$_i$)* and *End(Threat$_i$)*), and calculates the cumulative impact of the threat (See Section V-A).

*h) Response Context Handler:* This module activates and deactivates the response contexts *Start(Response$_i$)* and *End(Response$_i$)*, and calculates the cumulative impact of the response (See Section V-B).

*i) Policy Decision Point:* This module selects the optimal set of PEPs in order to deploy the activated response rules. Then, the module generates the appropriate scripts to reconfigure these PEPs, in order to enforce the activation (or deactivation) of response rules in the monitored system [28].

*j) Policy Enforcement Point:* These modules are reconfigured, using the scripts generated by the previous module, in order to enforce OrBAC rules, and thus to activate or to deactivate response polices in the monitored system [12], [13].

## VII. CASE STUDY: VoIP SERVICE

We consider the VoIP service of a mid-size company with 1000 clients as a case study. SIP is used as the Signaling protocol, and the media is transported using RTP or SRTP protocols. We suppose that each client is subscribed with one of the two available account types: premium accounts (20% of clients) for business use, and normal account (80% of clients) for personal use. In normal circumstances (i.e. *nominal* context), premium and normal accounts are allowed to make standard or secured calls. Thus, the Operational Policy[2] OP will be:

- *permission(premium, call[3], all, nominal)*
- *permission(premium, secure_call, all, nominal)*
- *permission(normal, call, all, nominal)*
- *permission(normal, secure_call, all, nominal)*

First, Section VII-A shows threats identification, and the specification of one of these threats. Second, Section VII-B presents

---

[2]The organization has been omitted since the case study treats a single organization

[3]For simplicity, we consider the activities *call* and *secure_call* includes making a call, and receiving a call.

the associated response policy specification. Finally, Section VII-C presents how the latter response policy is activated and deactivated, in three different scenarios.

### A. Threat Specification

First, the expert identifies the following potential attack objectives:

1) SPIT (Spam over IP Telephony) Storm
2) Conversation Tapping (Internal Attack)
3) Audio Injecting - Mixing (Internal Attack)
4) Server partial Denial of Service
5) Client Toll Fraud
6) Client Account highjack

The first attack consists of sending Spam over IP Telephony (SPIT), which definitely annoys the clients. Regular SPIT attacks with no proper response may cause the clients to change the operator. The second attack violates the confidentiality of the calls; while the third attack consist of injecting audio to an established call between two parties (and thus violates the integrity of the calls). It is obvious that the second and third attacks are not tolerable especially for the premium clients. The objective of the forth attack is to cause a partial (or total) DoS on SIP servers, and thus violates the availability of the VoIP service for all the clients. The fifth and sixth attack objectives aim a victim user whose account had been already compromised.

We decide to retain the first four attack objectives to be handled as strategic threats, and therefore we proceed to define the associated contexts. The remaining two attack objectives will be handled by the administrator with local countermeasure(s). Therefore, there is no need to specify the associated strategic responses. Due to space limitation, we will proceed with the specification of the last retained threat (i.e. Server partial Denial of Service):

- if $(SL_{Threat_4} > 5)$ then *Start(Threat₄)*
- if [*Start(Threat₄)* and $(SL_{Threat_4} < 4)$] then *End(Threat₄)*

### B. Response Specification

The expert must specify the Response Policy RP using Or-BAC, to enforce a set of response rules for each of the previous four threats. The expert may specify the same response for several threats at the same time. In this case study, we decide that the second and third threats should be associated with the same response. Due to space limitation, we will proceed with the specification of the last response policy associated to the retained threat in Section VII-A (i.e. Server partial Denial of Service).

*1) $Resp_C$:* This strategic response is used when the system is under considerable (or even near total) DoS. According to this response, the system must allow only the premium clients to perform and receive calls. However, they are prohibited to perform secure calls, due to their greater resources consumption. Moreover, during these harsh conditions, normal clients are prohibited to perform or receive calls. Therefore, the response rules that must be enforced are $Resp_C$:

- *Prohibition(premium, secure_call, all, $Resp_C$)*
- *Obligation(admin, start$_C$_notify, premium, $Resp_C$, ¬$Resp_C$)*
- *Prohibition(normal, call, all, $Resp_C$)*

- *Prohibition(normal, secure_call, all, $Resp_C$)*
- *Obligation(admin, start$_C$_notify, normal, $Resp_C$, ¬$Resp_C$)*

On the other hand, when the administrator has to end the activated response, the following rules have to be enforced in the monitored system. We notice that we do not need to explicitly specify the counterpart of *permissions* and *prohibitions*, because they are automatically deactivated when *End($Resp_C$)* is triggered. (See Section IV-B):

- *Obligation(admin, end$_C$_notify, premium, ¬$Resp_C$, $Resp_C$)*
- *Obligation(admin, end$_C$_notify, normal, ¬$Resp_C$, $Resp_C$)*

The second and fifth rules in $Resp_C$ are defeasible *obligations*, and thus the counterpart rules in ¬$Resp_C$ are also defeasible *obligations*. Finally, the triggering conditions of these response contexts are:

- if [*(Start(Threat)₄ is true)* and $(CI_{Threat_4} > 500)$] then *Start($Resp_C$)* is true
- if *(Start($Resp_C$) is true)* and [*(End(Threat₄) is true)* or $(CI_{Resp_C}(t) > CI_{Threat_4}(t))$] then *End($Resp_C$) is true*

Figure 3 depicts the mapping between the attack objectives, the threat contexts and the response contexts. The generated attack graph is used to calculate the success likelihood of ongoing threats. Interested readers may refer to [24]. The attack graph contains elementary attack steps that lead to the pre-identified attack objectives.

### C. Activating and Deactivating the Response Policy

Once the threat and response contexts are specified, they can be used by the activation/deactivation framework and architecture, as presented in Section V-B and Section VI. Let us suppose that the attacker begins to flood the SIP server using several infected machines. This attack incurs a heavy DoS for the VoIP service (85%). Therefore, the success likelihood of the attack objective *Server DoS* rises dramatically. The associated threat *Threat₄* becomes active, and thus the associated context *Start(Threat₄)* is triggered. The impact of this threat is a decrease in the availability of the VoIP service for all clients ($\delta A \neq 0$; $\delta C = \delta I = 0$). We consider that the incurred cost for each premium user is $\delta A = -1.5/min$, while the cost for a normal user is $\delta A = -1/min$ with a delay of thirty minutes. Consequently, normal users tolerate a thirty minutes service downtime, which is not acceptable for premium users.

To counter this ongoing attack, the associated response context *Response₃* is activated at time $T_{activ} = 0\ min$ (by activating *Start(Response₃)*). In consequence, the remaining resources are exclusively dedicated to the premium users, while the normal users are denied of making or receiving calls. The deactivation process of *Response₃* would be performed in one of the following cases:

*a) The Threat Vanishes:* The deactivation of *Threat₄* is due to a decrease of its success likelihood. For instance, if the flooding bots are disinfected (See the attack graph in Figure 3), the success likelihood of the ongoing threat will decrease significantly (below 4), and therefore *End(Threat₄)* is activated: The evanishment of the ongoing threat leads to the deactivation of the response.
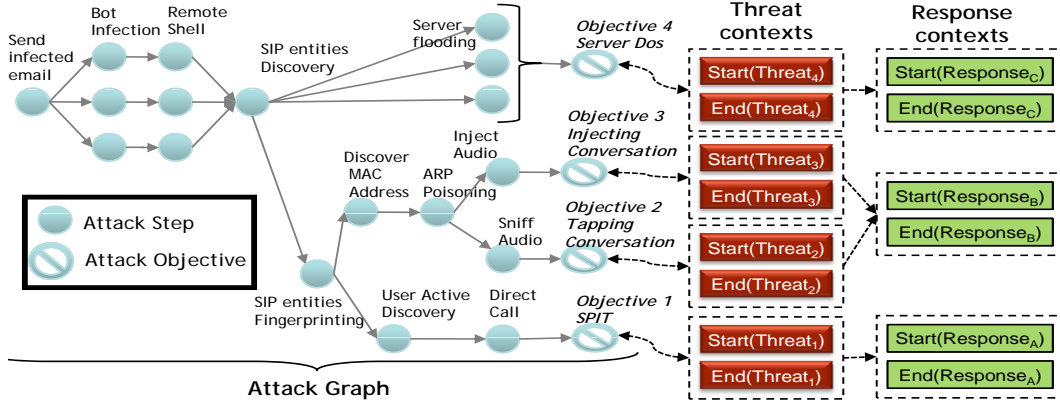
Fig. 3. The mapping between attack objectives, threat contexts and response contexts of the VoIP case study.

*b) The Threat Impact Decreases:* In the second scenario the threat remains present; in other words its success likelihood does not decrease. However, its impact on the monitored system decreases: for example the DoS magnitude falls from 85% to only 20% at $t = 40\ min$. The threat impact may decrease for several reasons, for instance:

1) The DoS level may decrease due to other tactic or strategic response(s) that has been activated by the monitored system (e.g. blocking traffic from attacker domain, activation of backup servers)

2) The DoS level may decrease due to third party actions. For instance, another operator blocked the attacking machines after detecting the high level of outbound traffic, or upon request from the monitored system (i.e. the victim system, operator)

3) The attacker might have decided, for unknown or hidden reasons, to reduce the DoS magnitude. For instance, the attempt was undertaken in order to blackmail the monitored system.
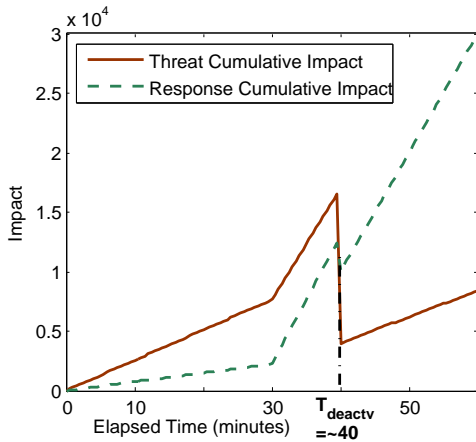
becomes larger than the cost of the threat $Threat_4$. Therefore, the response policy must be deactivated immediately: the associated event-based context $End(Response_C)$ is launched at $T_{deactiv} \approx 40min$. Despite the response deactivation, the threat context $Threat_4$ remains active, because the threat is always present. Moreover, this ensures that $Start(Resp_C)$ could be re-activated if the impact of $Threat_4$ rises again.

*c) The Response Impact exceeds the Threat Impact:* In the third scenario the threat remains present, and its success likelihood does not decrease. Moreover, its impact remains steady over time: the DoS level of the threat remains at 85%. Figure 5 shows that at $t = 68\ min$, the cumulative impact of $Resp_C$ becomes higher than the one of $Threat_4$. Thus, $Resp_C$ is no more effective, and $End(Resp_C)$ is launched at $T_{deactiv} \approx 68\ min$. Even though the response was deactivated, the threat context $Threat_4$ remains active to signal that the threat is always present.



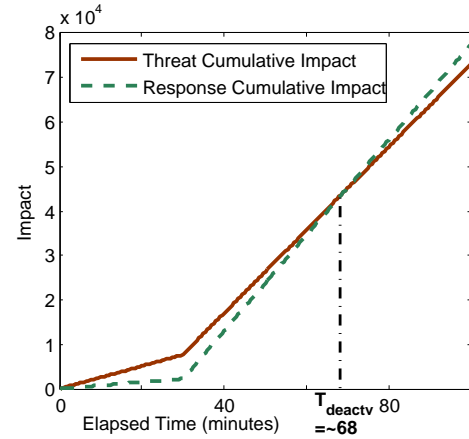Fig. 4. The cumulative impact of $Threat_4$ drops below the cumulative impact of $Response_3$.



Fig. 5. The cumulative impact of $Resp_C$ exceeds the cumulative impact of $Threat_4$.

Figure 4 shows the evolution of the cumulative impact of $Threat_4$ and $Response_C$ over time. We can notice also a rise of the slopes of the two curves at $t = 30\ min$, which is due to the addition of normal users' cost after their tolerance delay had expired. The cost incurred by the activated $Response_C$

## VIII. Conclusion

In this paper, we presented first how to specify response requirements as response policy with associated threat and response contexts. We distinguished threat contexts from response contexts, which was not done in previous work.

The specification considers the security and response requirements at both the beginning and the end of the response. In other words, the response policy indicates how responses should be deployed when activated and deactivated, using $Start(Response_i)$ and $End(Response_i)$. Moreover, a temporal reaction and countermeasure taxonomy was proposed to help the expert specify the aforementioned response contexts. Second, a dynamic risk-aware framework to activate/deactivate the responses was described. The framework considers the two fundamental components of a risk: the success likelihood and the impact of the threat. Moreover, the impact of the response is also considered. Additionally, we introduced the cumulative impact metric for threats and responses, in order to assess the cumulative impact over time due to variation in confidentiality, integrity and availability levels of monitored systems and services. Third, an online architecture was proposed to activate/deactivate and deploy response policies in real-time. Finally, our work was validated using a VoIP service case study.

However, we did not consider a dynamic enforcement of response rules in the monitored system. Each response rule was statically mapped into a generic script: if a response rule is activated, the associated script is launched. [28] shows how to enforce security rules while minimizing configuration changes and reducing resource consumption. Such model can optimize the enforcement of activated/deactivated response rules.

On the other hand, potential conflicts were handled by assigning priorities to threat and response contexts. When a conflict occurs, the rule with the highest priority is enforced. As a first approach to resolve the issue, the prioritization derived from the preliminary risk analysis (See Section IV-A) was used: the response associated with the threat with a higher risk, possesses a higher priority. Future work will focus on a risk-aware transactional management of multiple responses which may be activated/deactivated simultaneously.

## REFERENCES

[1] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, and E. Spafford, "Adepts: Adaptive intrusion response using attack graphs in an e-commerce environment," *Dependable Systems and Networks, International Conference on*, vol. 0, pp. 508–517, 2005.

[2] H. Wei, D. Frinke, O. Carter, and C. Ritter, "Cost-benefit analysis for network intrusion detection systems," 2001.

[3] T. Toth and C. Kruegel, "Evaluating the impact of automated intrusion response mechanisms," in *ACSAC'02: Proceedings of the 18th Annual Computer Security Applications Conference*, 2002, p. 301.

[4] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok, "Toward cost-sensitive modeling for intrusion detection and response," *Journal of Computer Security*, vol. 10, no. (1/2), pp. 5–22, 2002.

[5] I. Balepin, S. Maltsev, J. Rowe, and K. Levitt, "Using specification-based intrusion detection for automated response," in *In Proceedings of the 6th Intl Symp on Recent Advances in Intrusion Detection*, 2003, pp. 136–154.

[6] N. Stakhanova, S. Basu, and J. Wong, "A cost-sensitive model for preemptive intrusion response systems," in *AINA '07: Proceedings of the 21st International Conference on Advanced Networking and Applications*, 2007, pp. 428–435.

[7] M. Jahnke, C. Thul, and P. Martini, "Graph based metrics for intrusion response measures in computer networks," in *LCN '07: Proceedings of the 32nd IEEE Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1035–1042.

[8] N. Kheir, H. Debar, N. Cuppens-Boulahia, F. Cuppens, and J. Viinikka, "Cost evaluation for intrusion response using dependency graphs," in *IFIP International Conference on Network and Service Security (N2S'09)*, June 2009.

[9] A. Gehani and G. Kedem, "Rheostat : Real-time risk management," in *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, 2004, pp. 15–17.

[10] W. Kanoun, N. Cuppens-Boulahia, F. Cuppens, and F. Autrel, "Advanced reaction using risk assessment in intrusion detection systems," in *Second International Workshop on Critical Information Infrastructures Security (CRITIS07)*, Springer, Ed., Spain, 2007.

[11] H. Debar, Y. Thomas, N. Boulahia-Cuppens, and F. Cuppens, "Enabling automated threat response through the use of a dynamic security policy," *Journal in Computer Virology*, vol. 3, no. (3), 2007.

[12] F. Cuppens, N. Cuppens-Boulahia, Y. Bouzida, W. Kanoun, and A. Croissant, "Expression and deployment of reaction policies," in *SITIS '08: Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*. IEEE Computer Society, 2008, pp. 118–127.

[13] F. Cuppens, N. Cuppens-Boulahia, W. Kanoun, and A. Croissant, *Web-Based Information Technologies and Distributed Systems*. Paris: Atlantis Press, May 2010, ch. 8: A Formal Framework to Specify and Deploy Reaction Policies, pp. 159–188.

[14] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, 2001.

[15] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in *POLICY '01: Proceedings of the International Workshop on Policies for Distributed Systems and Networks*. London, UK: Springer-Verlag, 2001, pp. 18–38.

[16] A. Abou El Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin, "Organization Based Access Control," in *4th IEEE Policy*, June 2003.

[17] F. Cuppens, N. Cuppens, T. Sans, and A. Miège, "A formal approach to specify and deploy a network security policy," in *Formal Aspects in Security and Trust FAST*, Toulouse, France, August 2004.

[18] F. Cuppens and A. Miège, "Modelling contexts in the or-bac model," in *ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference*, Las Vegas, USA, 2003, p. 416.

[19] Y. E. Rakaiby, F. Cuppens, and N. Cuppens-Boulahia, "Formalization and management of group obligations," in *POLICY '09: Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks*, 2009, pp. 158–165.

[20] S. Preda, F. Cuppens, N. Cuppens-Boulahia, J. G. Alfaro, L. Toutain, and Y. Elrakaiby, "Semantic context aware security policy deployment," in *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*. New York, NY, USA: ACM, 2009, pp. 251–261.

[21] C. Baral, J. Lobo, and G. Trajcevski, "Formal characterizations of active databases: Part II," in *5th International Conference on Deductive and Object-Oriented Databases (DOOD'97)*, Montreux, Switzerland, 1997, pp. 247–264.

[22] N. Stakhanova, S. Basu, and J. Wong, "A taxonomy of intrusion response systems," *International Journal of Information and Computer Security*, vol. 1, no. (1/2), March 2007.

[23] C. Carver and U. Pooch, "An intrusion response taxonomy and its role in automatic intrusion response," in *The 2000 IEEE Workshop on Information Assurance and Security*, June 2000.

[24] W. Kanoun, N. Cuppens-Boulahia, F. Cuppens, S. Dubus, and A. Martin, "Success likelihood of ongoing attacks for intrusion detection and response systems," in *The IEEE International Conference on Computational Science and Engineering*, vol. 3. Vancouver, Canada: IEEE Computer Society, 2009, pp. 83–91.

[25] P. Ning, D. Xu, C. G. Healey, and R. S. Amant, "Building attack scenarios through integration of complementary alert correlation methods," in *in Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS04*, 2004, pp. 97–111.

[26] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 273–284.

[27] F. Cuppens, F. Autrel, and A. M. et S. Benferhat, "Recognizing malicious intention in an intrusion detection process," in *Second International Conference on Hybrid Intelligent Systems*, Santiago, Chili, December 2002, pp. 806–817.

[28] S. Preda, N. Cuppens-Boulahia, F. Cuppens, J. García-Alfaro, and L. Toutain, "Reliable process for security policy deployment," in *SECRYPT*, 2007, pp. 5–15.