Expression and Deployment of Reaction Policies

Frédéric Cuppens[1], Nora Cuppens-Boulahia[1], Yacine Bouzida[1], Wael Kanoun[1,2] and Aurélien Croissant[1]

[1] TELECOM Bretagne, Cesson Sévigné, France
[2] Bell Labs - Alcatel Lucent, Nozay, France

Abstract

Current prevention techniques provide restrictive responses that may take a local reaction in a limited information system infrastructure. In this paper, an in depth and comprehensive approach is introduced for responding to intrusions in an efficient way. This approach considers not only the threat and the architecture of the monitored information system, but also the security policy. The proposed reaction workflow links the lowest level of the information system corresponding to intrusion detection mechanisms, including misuse and anomaly techniques, and access control techniques with the higher level of the security policy. This reaction workflow evaluates the intrusion alerts at three different levels, it then reacts against threats with appropriate counter measures in each level accordingly.

1 Introduction

Intrusion Detection Systems (IDSs) are widely used to secure information systems, and became a primary component in modern security architecture solutions. Different intrusion detection techniques have been introduced and implemented in the governmental, academic and commercial information systems. Moreover, Intrusion Prevention Systems (IPSs) are highly used along with the IDSs to counter the detected threats. However, current intrusion prevention devices act only as conventional firewalls with the ability to block, terminate or redirect the traffic when the corresponding intrusion event is triggered. In other words, the intrusion response is statically associated with one (or several) intrusion event(s). Nevertheless, in [11] where the a contextual security policy have been defined, a policy reaction formalism was introduced. This reaction is performed globally allowing a global access control modification in an organization. However, the scalability remains an open issue that was not addressed in [11]. The threat context mechanism was implemented as a set of contextual rules that are triggered when the corresponding threat contexts become active. Only access control rules, i.e. permissions and prohibitions, were considered. We note that prohibitions and permissions are not appropriate to launch some actions, for instance shutting down a server immediately, or redirecting undesirable traffic (e.g syn-flooding packets)

On the other hand, the anti-correlation approach [4] allows an easiest manner to express the reaction activation along with the scalability consideration. However, the reaction within this approach is performed locally without taking into account the global framework where it is implemented. This issue motivated us to improve the contextual security policy not only by using the permissions and prohibitions, but also by focusing on obligations corresponding to actions that are inherent within the whole reaction policy framework. Another objective of our work is to combine in a coherent manner both approaches within a reaction workflow, taking into account different levels of reactions.

A *system oriented* taxonomy is presented in [20] with a classification into degree of automation and activity of triggered response. The automatic response is organized by ability to adjust, time of response, cooperation ability and response selection method. This taxonomy and others (not presented here due to space limitation) do not describe a thorough description of the response including the response strategy, duration, effectiveness and impact of the response. Toth and Kruegel [21] propose a cost sensitive approach that balances between intrusion damage and response cost in order to choose a response with the least impact. Lee et al. [14] also discuss the need to consider the cost of intrusions damage, the cost of manual and automated response to an intrusion, and the operational cost, which measures constraints on time and computing resources.

In this paper, we propose an auto-adaptive model that starts from the security policy management of the monitored information system. The low level tools including intrusion detection and access control mechanisms that are implemented locally to monitor the information system, are configured according to the high level security specifications. Then, whenever it is necessary, some of the generated alerts are forwarded to the upper level , by crossing different levels of reaction. At the upper level, and accordingly to the detected threat, an evaluation of the current system state takes place. Consequently, either direct responses will launched or the whole security policy will be changed. We define three reaction levels; (1) low level reaction, (2) intermediate level reaction, and (3) high level reaction. Each level considers particular security requirements and deploys appropriate security components and mechanisms to react against the detected threats.

The rest of the paper is organized as the following. Section 2 presents the reaction requirements and the reaction policy expression. In particular, we develop our approach to manage the conflicts between the various operational, minimal constraints and contexts of threats. Section 3 describes the reaction deployment framework. Section 4 presents the architecture of the reaction workflow with the different reaction levels. Section 5 presents an illustrative VoIP use case. Finally, Section 6 presents future work and concludes the paper.

2 Reaction policy

We view a security policy as a set of requirements corresponding to permissions, prohibitions and obligations. In the security literature, it is generally considered that these requirements apply to users or processes (i.e. subjects) when they access to resources (i.e. objects) in order to execute services or programs (i.e. actions). The security policy includes requirements that apply in "normal" situations, i.e. when no intrusion occurs. We call this part the *operational* policy, and it typically includes access control requirements.

The *reaction* policy is another part of the policy which specifies security requirements that are activated when an intrusion is detected. It is a set of rules that specify what happens in case of violation (or attempt of violation) of some requirements of the operational security policy. According to these (attempts of) violations and their impacts on the target information system, new permissions, prohibitions or obligations are activated and pushed into the appropriate security components. For instance, if an intrusion occurs, and the alert diagnosis identifies the path of the attack or the targeted equipment pieces by this attack and used to reach the intrusion objectives, (1) some packet flows have to be rejected or at least redirected or, (2) some of the vulnerable equipment used by the attack have to be stopped or at least isolated to contain its spread in the whole system.

Our approach to specify the security policy is based on the an Organizational Based Access Control (OrBAC) model [8]. In the remainder of this section, we shall first recall the basic principles of the OrBAC model, then we present how this model can be used to express the reaction policy. Finally we address the issue of security requirements conflicts.

2.1 The OrBAC model

The security policy specification is based on an expressive security model, the OrBAC model. One of the OrBAC contributions is the abstraction of the traditional triples $\langle subject, action, object\rangle$ into $\langle role, activity, view\rangle$. The entities $subject$, *action* and $object$ are called *concrete entities* whereas the entities $role$, *activity* and $view$ are called *organizational 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. *consult data*) is considered as an operation which is implemented by some actions defined in the organization (e.g. *read* for a file and *select* for a database). This is why they can be grouped within the same activity for which we may define a single security rule. One of the main contributions of the OrBAC model is that it can model *context* that reduces the applicability of the rules to some specific circumstances [8]. Thus, *context* is another organizational entity of the OrBAC model. The OrBAC model defines four predicates[1]:

- *empower*: $empower(s, r)$ means that subject s is empowered in role r.
- *consider*: $consider(\alpha, a)$ means that action α implements the activity a.
- *use*: $use(o, v)$ means that object o is used in view v.
- *hold*: $hold(s, \alpha, o, c)$ means that context c is true between subject s, action α and object o.

Security requirements are specified in OrBAC by quintuples:

- $SR(decision, role, activity, view, context)$

which specifies that the decision (*i.e.* permission, prohibition or obligation) is applied to a given role when requesting to perform a given activity on a given view in a given context. We call these *organizational security rules*. An example of such a security rule is:

$- SR(permission, private\_host,$
$open\_HTTP, to\_Internet, default)$

which corresponds to a filtering requirement specifying that hosts assigned to the role *private_host* are permitted to open HTTP connection with the Internet in the default context (the default context is true every circumstance).

[1]In OrBAC, the organization is made explicit in every predicate but here, to simplify, the organization is left implicit since we consider always only one organization.

Another requirement may correspond to the following prohibition:

– $SR(prohibition, any\_host, send\_IP\_packet,$
$same\_source\_destination, default)$

where any_host is a role assigned to every network host, $send\_IP\_packet$ is the activity of sending IP packets, $same\_source\_destination$ is a view that contains any IP packet with a source IP address equal to its destination IP address. This is actually a security requirement to protect the system against the Land attack.

As suggested in the RBAC model [18], the organizational entity role is associated with a hierarchy called sub_role and security requirements are inherited through this hierarchy. In the OrBAC model, similar hierarchies to the three other organizational entities had been assigned: view, activity and context.

2.2 Using OrBAC to specify a reaction policy

The reaction policy corresponds to security requirements that are activated when intrusions occur. In OrBAC, this is modelled using special contexts called *threat contexts*. For this purpose, intrusion classes are associated with *threat contexts*. *Threat contexts* are activated when intrusions are detected, and are used to specify the reaction policy. The activation of these contexts leads to the instantiation of the policy rules in response to the considered threat. For instance, a Syn-flooding attack is reported by an alert with a classification reference equal to CVE-1999-0116, the target corresponds to some network $Host$ and some $Service$. Then the synflooding context is specified as follows [11]:

– $hold(\_, Service, Host, syn\_flooding) \longleftarrow$
$alert(Time, Source, Target, Classification),$
$reference(Classification, 'CVE-1999-0116'),$
$service(Target, Service), hostname(Target, Host).$

Notice that, since the intruder is spoofing (masquerading) its source address in a Syn-flooding attack, the subject corresponding to the threat origin is not instantiated in the hold predicate. When an attack occurs and a new alert is launched by the intrusion detection system, new $hold$ facts are derived for threat context Ctx. Therefore, Ctx is then active and the security rules associated with this context are triggered to react to the intrusion.

Notice also that we need to define a process that maps the intrusion detection alerts onto the $hold$ predicate. In the above $syn\_flooding$ example, this mapping is voluntary simplified. As shown in [11], it is generally more complex because we need a mapping that has variable granularity, to take into account the different scope of different attacks. For example, a distributed denial-of-service on all areas of the network needs to be handled differently than a targeted brute-force password-guessing attack. By appropriately defining the triples $\langle subject, action, object\rangle$ that are in the scope of a given threat context, it is possible to define such variable context granularity. As suggested in [11], a first form of reaction would be to update the access control policy by activating and deploying new permissions or prohibitions. For instance, a rule:

– R3: $permission(private\_host, open\_TCP,$
$to\_hostObelix, default)$

might be replaced by a new one such as:

– R4: $prohibition(any\_host, open\_TCP,$
$to\_hostObelix, syn\_flooding).$

In the second case, a reaction requirement may be specified by means of obligations. We may actually consider two different kinds of obligations called server-side obligation and client-side obligation. A server-side obligation must be enforced by the security components controlled by the security server and generally corresponds to immediate obligations. R5 is an example of such rules expressed in the OrBAC model:

– R5: $obligation(mail\_daemon, stop,$
$mailserver, imap\_threat)$

Client side obligations generally correspond to obligations that might be enforced after some delay. Several papers have already investigated this problem and suggested models to specify obligation with deadlines [7, 12]. For instance, if there is an intrusion that attempts to corrupt an application server by a Trojan Horse intrusion, then this server must be quarantined by the administrator within a deadline of 10s. R6 provides a specification of this requirement:

– R6: $deadline\_obligation(administrator, quarantine,$
$application\_server, trojan\_horse\_threat, before(10))$

where $deadline\_obligation$ can be used to specify one more attribute that corresponds to the deadline condition $before(10)$.

Obligations with deadline are more complex to enforce than immediate obligation. So, to simplify both the expression and implementation, we shall only consider immediate server-side obligations in the remainder of this paper.

2.3 Security requirements interpretation

Concrete security rules that apply to triples $\langle subject, action, object\rangle$ are modelled using the predicate $sr(decision, subject, action, object)$ and logically derived from organizational security rules by the general derivation rule:

RG: $SR(Decision, R, V, A, C) \wedge empower(Subject, R)\wedge$
$consider(Action, A) \wedge use(Object, V)\wedge$
$hold(Subject, Action, Object, C)$
$\rightarrow sr(Decision, Subject, Action, Object)$

When the security policy contains both permissions, prohibitions and obligations, conflicts between security requirements are inevitable. We can actually consider three different types of conflicts:

Contradiction: A contradiction occurs when it is possible to derive, for some subject, action and object, both $sr(permission, s, a, o)$ and $sr(prohibition, s, a, o)$.

Dilemma: A dilemma occurs when it is possible to derive, for some subject, action and object, both $sr(obligation, s, a, o)$ and $sr(prohibition, s, a, o)$.

Inability: An inability occurs when it is possible to derive, for some subject and object, both $sr(obligation, s, a_1, o)$ and $sr(obligation, s, a_2, o)$ and it is impossible to simultaneously execute both actions a_1 and a_2. For example, a_1 is the action stop a server and a_2 is the action start a server.

However, the approach suggested in the OrBAC model [6] does not include the detection of concrete permissions, prohibitions or obligations conflicts; but it provides means to detect and manage *potential* conflicts between organizational rules. The solution to manage contradictions and dilemmas actually differs from the one used to manage inability.

Management of contradictions and dilemmas. A potential contradiction (resp. dilemma) exists between an organizational permission (resp. an organizational obligation) and an organizational prohibition if these two rules may possibly apply to the same subject, action and object. The approach used to manage such conflicts is based on the definition of *separation* constraints assigned to organizational entities. A separation constraint assigned to two roles specifies that a given subject cannot be empowered in these two roles. Separation constraints for activities, views and contexts are similarly defined.

Thus, a potential contradiction between two organizational security rules is defined as follows (potential dilemma is similarly defined):

Definition: ***Potential contradiction.*** Two security rules $SR(permission, r_1, \quad a_1, v_1, c_1)$ and $SR(prohibition, r_2, a_2, v_2, c_2)$ are potentially conflicting if role r_1, activity a_1, view v_1 and context c_1 are respectively not separated from role r_2, activity a_2, view v_2 and context c_2.

Management of inability. Potential inability is managed using constraints assigned to activities called *antinomic* constraints. We say that two activities are antinomic if it is not possible to execute these two activities simultaneously. Of course, we can use antinomic constraints to manage inability because there is no inability between two organizational obligations if these obligations are associated with antinomic activities.

Combining separation and antinomic constraints, we can now detect every potential conflict. Priorities should be associated with such potentially conflicting security rules in order to avoid situations of real conflict. Prioritization of security rules must proceed as follows [6]: (1) Detection of potentially conflicting rules, (2) Assignment of priority to potentially conflicting rules.

Notice that this process is tractable because each time a new potential conflict is detected, the administrator can decide to insert a new constraint or define a new priority. Notice also that this process must be performed off-line, i.e. before the security policy is actually deployed. We then obtain a set of partially ordered security rules *SR(decision, role, activity, view, context, priority)*. Concrete security rules can be derived from the abstract security rules and are assigned with the same priority. It has been proved in previous works [6] the following theorem.

Theorem: If every potential conflict is solved, then no conflict can occur at the concrete level.

2.4 Strategies to manage conflicts

We observe that most of reaction requirements are in conflict with access control requirements, i.e. the access control policy may specify a permission whereas the reaction policy specifies a conflicting prohibition that applies when an intrusion is detected. For instance, HTTP is permitted when there is no intrusion but prohibited if an intrusion on the HTTP protocol is detected. These conflicts can be solved by manually assigning priorities between requirements as suggested in the previous section. However, it is easier to automatically solve these conflicts by assigning higher priority to the reaction requirement than to access control requirements. In fact, we consider three different types of activation contexts: *threat*, *operational* and *minimal*.

The *operational contexts* aim at describing traditional operational policy [8]. They may correspond to temporal, geographical or provisional contexts (i.e. contexts that depend on the history of previous executed actions). Since access control requirements are associated with operational contexts whereas reaction requirements are associated with threat contexts, we actually consider that threat contexts have higher priority than operational contexts.

However, there are some security requirements such as availability requirements that must be preserved even if an intrusion occurs. For instance, the access to the email server must be preserved even if some intrusions occur. This is modelled as a minimal requirement. *Minimal contexts* then define high priority exceptions in the policy, describing minimal operational requirements that must apply even in case of characterized threat.

Therefore, we consider two parameters to manage conflicting situations called *criticality* and *specificity*. A criticality parameter is used to assess context priority between the three defined categories of contexts *operational*, *threats* and *minimal*. We define an operator $Ł_c$ to assess the level of criticality of contexts, so that if Ctx is a set of well formed contexts: L_c: Ctx $\longrightarrow$ {ope, threat, min} with ope $<$ threat $<$ min. We define the criticality relation as follows: $c_1 <_c c_2 \longleftrightarrow L_{c1} < L_{c2}$. We consider also a specificity parameter that deals with inheritance and context composition, hierarchical specificity context inheritance. For instance, we say that c_2 is more specific than c_1 if $sub\_context(c_2, c_1)$. We define specificity for contexts as follows: $c1 \leq_s c_2 \longleftrightarrow sub\_context(c_2, c_1)$ and $c_1 <_s c_2 \longleftrightarrow c_1 \leq_s c_2 \wedge \neg(c_1 = c_2)$. We have then defined two strategies to assess rule priorities in case of potential conflicts and prove that they are not conflicting strategies, that is we never obtain conflicting decisions when applying them (see [11]).

3 Deployment of the reaction workflow

In Section 2, we gave an overview of the OrBAC model and how to manage the different contexts for a mapping with the current threats and how to resolve the conflicts between the security rules according to the active threat contexts. Deploying the reaction policies within a comprehensive framework needs the definition of each module and its function within the framework. The workflow model must also provide an abstract and global view of the security policy. This is the purpose of the Policy Instantiation Engine (PIE) to manage the global security policy. The PIE will have to clearly separate the global policy from its implementation in the PEPs (Policy Enforcement Points). In particular, the conflicts are to be solved at the abstract level before generating PEPs configurations. The PIE activates threat contexts and extracts a new security policy using the threat information triggered by the ACEs (Alert Correlation Engine) [11]. Concrete security rules are then sent by the PIE to two different modules. The rules corresponding to permissions and prohibitions are sent to the PDP (Policy Decision Point) that dispatches the policy according to PEPs realms and capabilities, and translates the policy rules to PEP-specific scripted commands. Notice that these PEPs correspond to security components such as IDSs and firewalls. Rules corresponding to obligations, are sent to another component called RDP (Reaction Decision Point) that translates the corresponding rules, such as shutting down an HTTP server or redirecting an undesired traffic generated by the syn-flooding attack, into REP (Reaction Enforcement Points) capabilities.

Figure 1 presents the components of the reaction workflow with the different links and information exchange between these modules. We describe, in the following, the different modules and their functions and goals within the suggested reaction workflow deployment framework. The inputs, outputs, configurations and functions of the different modules are summarized in Table 1.

Security policy management. This module describes two different data types. The first deals with the security objectives that are specified as a set of security states. The violated states are then derived using the security objectives defined within the security policy management. These security objectives are expressed using a predefined set of LAMBDA predicates (due to space limitation, we do not develop more this point in this paper) [10]. Such violated states are then used by different modules. In particular, the ACE uses these violated objectives in order to provide a global diagnosis of the current attack scenario performed by the attackers. The second data type deals with the policy definition where two kinds of security rules are distinguished:(1) permissions or/and prohibitions and (2) obligations. Notice that while the instantiated access control policy rules are pushed by the PIE to the PDP the obligations rules are pushed to the RDP.

System management. This module includes the operational context data and the topology information. The working hours context is an example of operational context data. These contexts may be combined with threat contexts (see section 2.4) and are used by the PIE to activate the right security rules. The information topology part describes the topology of the information system under monitoring such as information about network nodes (routers, servers, etc.), security components (IDSs, Firewalls, etc.) and so on. RDPs and PDPs use the topology information provided by the system management for appropriate reactions and PEPs and REPs components choice decisions.

Alert Correlation Engine (ACE). Generally, information produced by detection intrusion probes cannot be considered on their own. Indeed, this information actually comes from many sources and with different formats (ex: a Snort alert, a Netfilter firewall log, etc.). Moreover, there is a strong need for alerts volume reduction, semantics improvement and attacker recognition. Alert correlation, preceded by alert aggregation and fusion, aims at realizing these tasks, thus permitting false positives reduction and producing meta-alerts offering a better semantics and severity levels for more efficient analysis. The ACE constructs ongoing attack scenarios according to the alerts generated by the sensors. We use a semi-explicit ACE as suggested by CRIM (http://www.crim-platinum.org/). Note that other ACEs, such as that presented in [15], may be also used within our workflow. This correlation engine uses a description of elementary attacks in the LAMBDA language [10] to construct ongoing scenarios according to the alerts generated by the sensors. The security objectives are defined in the policy management level and are taken into consideration by the ACE to detect any violations of these objectives. Notice that an explicit ACE can also be used. However, by contrast to a semi-explicit ACE, it cannot handle alerts generated by anomaly detection systems, which correspond to

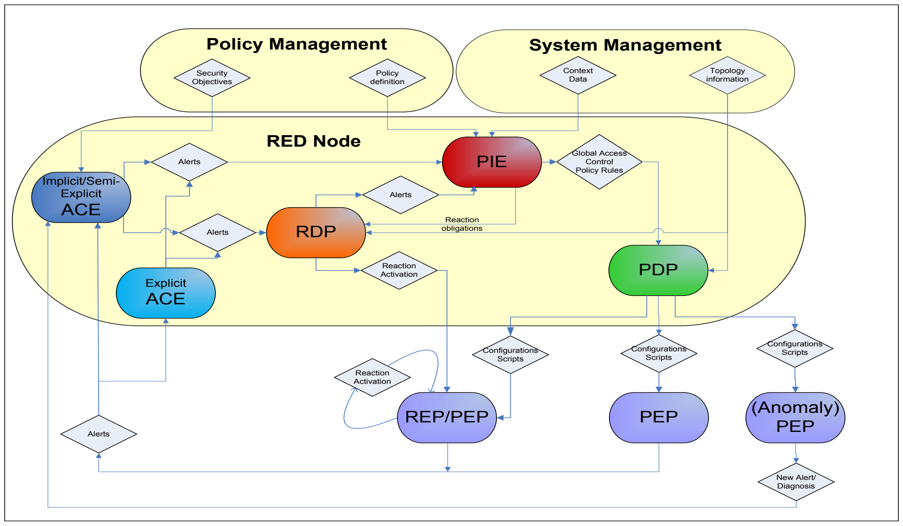


Figure 1. Reaction workflow deployment.

Table 1. Functions of the different modules.

Module	Input	Output	Configuration	Function
ACE	idmef messages	idmef messages	External security reference databases	Verify and update information in idmef messages for threat assessment.
PIE	idmef messages	Policy rules	Policy and context definitions	Activate threat contexts. Extract a new security policy from the active contexts.
RDP	idmef messages, set of local configurations	Reaction activation actions (scripts)		Reinforce the Immediate Reaction with an accurate diagnosis, an impact assessment and a larger choice of possible reactions. Decide on the appropriate reaction and activate reactions according to the REP-specific.
PDP	Policy rules	Config scripts	Policy to script translation rules	Segment the policy according to PEP realms and capabilities, and translate the policy rules to PEP-specific scripted commands.
REP	Reaction activation actions	Reaction actions		Launch the reaction orders received from the RDP in the case of the short term reaction and from PEPs in the case of immediate reaction.
PEP	Config scripts	idmef messages		Apply the configuration script that implements the security policy.

new attacks. The correlated alerts are then sent to the PIE for global reaction and to the RDP for appropriate decision on the possible reaction instances.

Policy Instantiation Engine (PIE). The security policy is specified by a set of rules whose activation depends on contextual data. Thus, a PIE has two major functions: (1) activate contexts which (2) trigger re-evaluation of the security policy (abstract rules). The PIE manages conflict resolution at the policy evaluation level to produce a consistent set of policy instances (concrete rules) to deploy. As suggested in section 2.3 conflict resolution is ensured at the abstract level; i.e. the conflicts are detected and resolved by examining abstract security rules and defining a partial order relationship between conflicting rules.

Policy Decision Point (PDP). It is the element where security policy decisions are made. Policies instantiated by the PIE according to threat contexts are transmitted to one or more PDPs. When it receives a policy instance, in our case an OrBAC concrete rule (permission or prohibition), a PDP has to map each policy instance onto concrete actions to enforce the policy on PEPs. A PDP thus has to be aware of its PEPs, so that it can translate first the rules into generic configurations, considering the kind of PEP (e.g. a firewall), and then the generic configurations into specific configurations, depending on the PEP implementation (e.g. a "Netfilter" firewall) [5]. Note that part of the decisional capability of the PDP relies on the fact that a given concrete policy rule may provide different actions on the PEPs. For instance, depending on the architecture of the information system, reconfiguring access to mail user accounts may be realized on the service itself, (e.g. pop3 service native configuration files) in the case of dedicated services, or at the infrastructure level (e.g. reconfiguration of Active Directory) in the case of federated services environment. A PDP should choose the most appropriate PEPs under its view relying on topology information made available.

Reaction Decision Point (RDP). The RDP decides which reactions should be triggered according to the attack scenarios received from the ACE. It has all the topology information of the network it is charged to react on. It manages the obligation activations received from the PIE without conflicting with the minimal security policy. The obligations are expressed as OrBAC concrete rules. The reaction decision is taken by considering the topology information of the monitored system, the reaction obligations and the impact of the elected reactions [13]. A diagnosis is also sent by the RDP to the PIE and handled at the high reaction level.

Policy Enforcement Point (PEP). The PEPs are the elements that enforce security policy decisions. They are fed with new policy instances that are translated and then sent by the PDP. Expressing new policies may have implications on different PEPs. For instance, an attack targeting a server may lead to a reaction which involves both the server (stopping the service) and a firewall (blocking the corresponding port). Each selected PEP for a policy instance is sent a configuration script considering both its type (for example IDS) and its implementation (such as Snort). In this architecture, we suggest not to create a distinction between PEP and sensors, but to consider that sensors are a special type of PEPs.

Reaction Enforcement Point (REP). This module launches the reactions either immediately or after receiving a reaction activation from the RDP. It can also activate an immediate reaction that might be embedded in the alert signature. In this case, an immediate reaction is directly activated and launched by the REP. We notice that a PEP and REP have two different functions and may be implemented in the same software of hardware device.

4 Reaction workflow architecture

Figure 1 presents the workflow deployment including the different modules. Three different reaction levels coping with the security requirements are determined in this framework; low level, mid level and high level reaction. In the following we describe the implementation and configuration of each level.

4.1 Low level reaction

We define low level reaction as the action that is possible to execute automatically just after an intrusion is detected. In the reaction policy, this corresponds to an obligation to immediately execute this action. We should note here that the reaction must be consistent with the minimal security policy. For example, if a service should be active for any circumstance then a reaction, which consists in stopping this service, should not be launched. Different immediate reactions may be activated. For example, the Snort IDS tool offers different manners to respond to an alert. The following signature is an example of such a reaction: *alert tcp any any <> 192.168.1.0/24 80 (content: " bad.html"; msg: "Not for children!"; react:block)*. A security officer may either add a *tcp-reset* option specified directly in the corresponding rule or use the inline snort capability that is interfaced with *iptables* hence permitting to drop the packet. However, we argue that the inline option is more appropriate since it does not let the targeted party receive the malicious packet when a drop option is used. The DDoS [2] architecture is a good example of this situation when the agents are receiving the order to flood the victim and the reaction only consists in resetting the corresponding connection between the masters and the agents.

The REP and PEP are two modules involved in the low level reaction (Figure 2(a)). The PEP in this level detects the intrusion while the REP performs a direct reaction over the detected intrusion. In this case, the REP and PEP are implemented in the same device. For the snort rule example presented above, a reaction that consists in blocking a forbidden website is launched once the "bad.htm" content is seen. We notice that the low level reaction rules are configured by the different scripts issued from the PDP. However, whether a counter measure is effective at this level is not a shortcoming for the global security vision since the other two reaction types at the upper architecture levels look for an in-depth reaction.

4.2 Mid level reaction

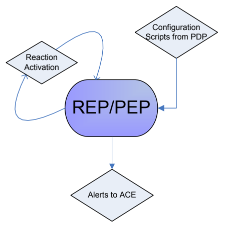
The low reaction defines an appropriate reaction which is generally specified within a signature. At the mid level, intrusion alerts and the different immediate reaction information coming from the lower level are used as an input for aggregation, fusion and correlation functions CRIM in order to construct a local ongoing attack scenario that follows the steps of an attacker(s) and determine his objective(s). These functions are performed by the ACE (Figure 2(b)) using a semi-explicit correlation mechanism [9].

However, at the mid level, we do not push reactions corresponding to obligations explicitly specified in the reaction policy. Instead, anti-correlation [4] is used as a way to find automatically a set of counter measures in order to stop not elementary actions but the global attack scenario. Some correlation and fusion tools, implemented during the last decade, provide a set of counter measures that may be either activated automatically or let the administrator choose the appropriate ones for security agility considerations [16].

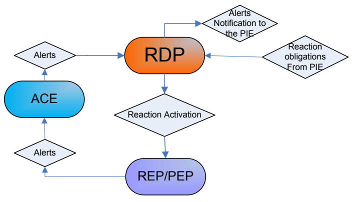
Notice that a diagnosis is derived within this step for improving the reaction process. Activating an automatic or a manual response depends on the confidence level of the diagnosis and the automatic choice may be performed by measuring the impact of the corresponding reaction [13].

4.3 High level reaction

The mid level reaction takes advantage of the low level reaction actions and the selected reactions to stop ongoing attack scenarios. However, these two reaction levels do not consider an in depth revision of the global security. This suggested the introduction of the high level reaction that aims to evolve the security policy of the monitored system according to the current threat. The first input data at this reaction level corresponds to the correlation and intermediate reaction alerts emanating from the intermediate reaction level. A global diagnosis is drawn using these input data. Using the global security policy and the different threat and operational contexts, contextual policy rules are applied when the corresponding contexts become active. Therefore three functions, should be performed at this level; activating contexts [11], triggering generic policy rules accordingly and producing a consistent set of rules to deploy



(a) low level reaction loop.



(b) Mid level reaction loop.

Figure 2. Low and Mid levels reaction workflow.

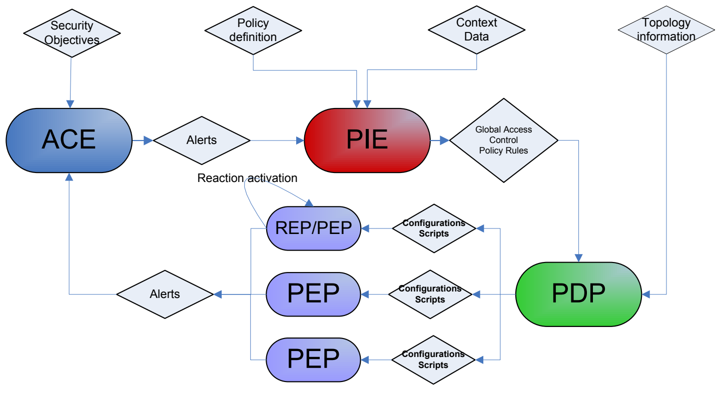


Figure 3. High level reaction loop.

while ensuring conflict resolution with the minimal security requirements. As a result of this level, a new security policy is redeployed as long as the threat or its consequences remain present.

Figure 3 shows the different modules involved in the high level reaction mechanism. We note that the mid level only complies with the minimal security policy and the different counter measures are taken from the vision of the attack scenario issued from the short term level while the high level uses the global security policy and activates policy rules according to threat contexts.

5 VoIP use case

The case study we present is a VoIP framework that illustrates the effectiveness of our reaction process. The considered attacks are those targeting the SIP [17] (Session Initiation Protocol) emerging protocol that has seen an increasing interest from academia and industrial communities. Many soft and hard SIP components are currently developed and widely used by Telco operators who are offering VoIP service mostly with little or no security issues. We give an example for reacting against attacks targeting ISP and VoIP infrastructures.

The architecture that we have used in our use case is the IMS [1] framework. Many Telco operators are interested in IMS, and even some of them had deployed it.

The architecture (see Fig. 4) is composed of different SBCs (Session Border Controllers) that filter all the incoming traffic from Internet or legitimate clients. Since SBCs cannot filter all undesired traffic, VoIP intrusion detection system tools, such as that developed in [3], are deployed behind these SBCs to detect different attacks using some vulnerabilities of the SIP protocol. Other emerging VoIP IDSs [19], based on state machine analysis similar to previous works such as NetStat [22], may be used as a complementary IDS in our framework. Since this tool is not made available, we only consider that presented in [3].

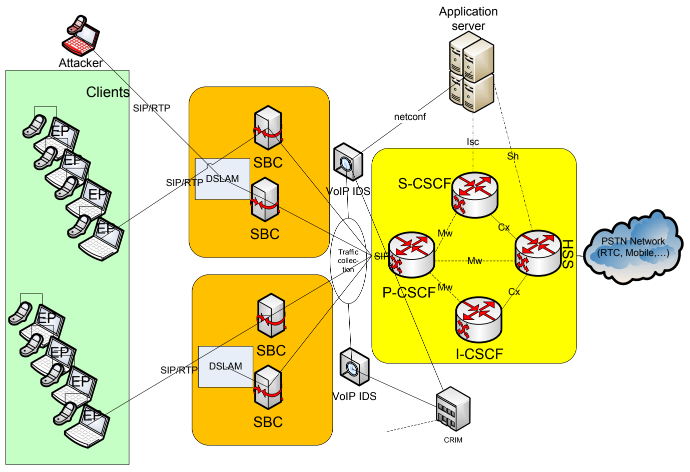


Figure 4. Framework architecture.

One of the attack scenarios we consider is described in Figure 5.

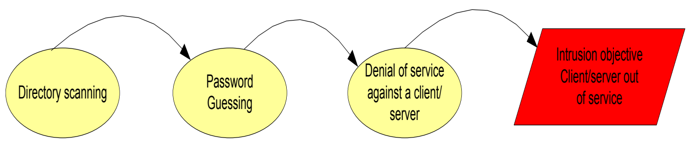

Directory scanning
Password Guessing
Denial of service against a client/ server
Intrusion objective Client/server out of service

Figure 5. VoIP attack scenario.

Generally, an attacker has to perform several actions in order to achieve her goal. In our example, the attack scenario is composed of 3 elementary attacks. In the first step, the attacker tries to collect valid identities corresponding to legitimate clients in the operator database (HSS in our case). This attack may be omitted particularly for those identities that are present in the red lists. However this attack is tested in platforms of different operators and the experiments were successful [3]. The second step consists in guessing the password of the legitimate identity discovered during the first step. The attacker sends REGISTER messages with different passwords using a dictionary. The http-digest mechanism is followed and according to the answer, he may follow up his attack. Once the attacker has guessed the password he may either perform a denial of service against a server such as the P-CSCF, or against a client by sending an infinite loop of INVITE, BYE. Notice that the BYE request is sent when the OK response from the callee (i.e. the target client answers to the call) is received. This will keep the callee phone ringing and unavailable as long as the attacker have not ceased the last attack. However, there is no reaction that could be taken within the current IMS framework [1].

Table 2 gives the mapping between the different logical modules we proposed in the previous sections and their corresponding network components present in our use case framework.

Logical modules	Network component
PEP	VoIP IDS, SBC, P-CSCF
REP	SBC,P-CSCF
ACE	CRIM Correlation engine
RDP	CRIM anti-correlation engine
PDP	Application server
PIE	Application server

Table 2. Logical module and network component module mapping.

We note that all the elementary attacks have been detected by the VoIP IDSs [3] used in our platform. These elementary attacks are collected by these IDSs and forwarded to the CRIM-ACE for building the corresponding attack scenario.

Using our approach, the reaction according to the three levels workflow is described as the following:

Low level reaction. Once the directory scanning attack is detected, the VoIP IDS sends the alert to the CRIM engine and sends, in the mean time, a new reaction to the SBC that consists in adding a rule for decreasing the rate of the requests coming from (1) the IP address of the attacker if this attacker is a client of the our ISP or (2) the couple (IP address, the SIP Via field) when the requests originate from a foreign ISP. This is due to the IP and logical SIP URI spoofing reasons.

Mid-level reaction. If the attacker does not stop at the first stage and continues towards the password guessing attack then the corresponding scenario is constructed thanks to the VoIP IDS alerts and the CRIM engine. At this stage, the RDP implemented aside the CRIM engine takes a reaction decision that consists in reconfiguring the P-CSCF in order to stop answering the requests originating from the attacker IP and logical addresses. The P-CSCF may stop forwarding these requests to the concerned servers and may log these requests for forensic analysis. One possible obligation rule corresponding to stop forwarding the requests by the P-CSCF is as follows:

$$SR(obligation, P\text{-}CSCF, stop\_forwarding, attacker, password\_guessing\_threat).$$

High-level reaction. The attacker may have guessed the password and starts various attacks such as disturbing other legitimate clients by sending infinite INVITE requests followed by the BYE method every time the target client tries to answer. At this stage, the CRIM engine sends the whole detected attack scenario to the application server (AS) that plays the role of the PIE. The “DoS against a client” threat context is activated and the corresponding abstract OrBAC rules are generated. One possible abstract rule related to this context is:

$security\_rule(prohibition, attacker, request, any,$

$dos\_client\_threat)$.

The concrete rule derived from this OrBAC abstract rule consists in blacklisting the attacker at the edge SBC.

6 Conclusion

In this paper, we have presented a novel process for expressing reactions and the corresponding architecture that aim at reacting against threats using different levels. In the low level, reactions are configured a priori within the security components configuration for an immediate reaction. The mid level reaction tries to counter the ongoing attack scenario by launching an appropriate counter measure. Finally, the high level reaction is performed by using the contextual policy that builds upon the OrBAC formalism. The proposed approach is robust since the different security components are configured from a high level point of view and the reactions against the detected threats cross three levels for applying the best reaction issue. The use case example we have developed and tested within a real Telco operator demonstrates the effectiveness of our proposal within the new generation networks. The different attacks were successful against different operators' infrastructures and are detectable by the VoIP IDS. The different reactions applied at each level allowed decreasing the effects of the threat and stopping it at the last stage without the intervention of a human security officer. Our first future work consists in applying the different reactions in a transactional mode since in some situations conflicts may occur between the different reactions that are launched in the different levels. The other direction consists in considering novel attacks within this workflow for a better reaction.

Acknowledgment

This work was partially supported by the European CELTIC ReD project. The authors would like to thank all the other partners for their participation and contribution.

References

[1] The 3rd Generation Partnership Project. Available at: http://www.3gpp.org/, 2007.

[2] Y. Bouzida, F. Cuppens, and S. Gombault. Detecting and Reacting Against Distributed Denial of Service Attacks using Alert Correlation. In *IEEE Intenational Conference on Communications*, 2006.

[3] Y. Bouzida and C. Mangin. Detecting anomalies in VoIP networks. In *3rd International Conference on Avilability, Reliability and Security ARES08*, Barcelona, Spain, 2008.

[4] F. Cuppens, F. Autrel, Y. Bouzida, J. Garcia, S. Gombault, and T. Sans. Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework. *Annales des télécommunications*, March 2006.

[5] F. Cuppens, N. Cuppens, T. Sans, and A. Miége. A formal approach to specify and deploy a network security policy. In *Formal Aspects in Security and Trust FAST*, August 2004.

[6] F. Cuppens, N. Cuppens-Boulahia, and M. B. Ghorbel. High Level Conflict Management Strategies in Advanced Access Control Models. *Electronic Notes in Theoretical Computer Science*, 186:3–26, 2007.

[7] F. Cuppens, N. Cuppens-Boulahia, and T. Sans. Nomad: A Security Model with Non Atomic Actions and Deadlines. In *18th IEEE CSFW*, pages 186–196, 2005.

[8] F. Cuppens and A. Miège. Modelling Contexts in the OrBAC Model. *ACSAC*, page 416, 2003.

[9] F. Cuppens and A. Mige. Alert Correlation in a Cooperative Intrusion Detection Framework. In *IEEE Symposium on Security and Privacy*, Oakland, USA, 2002.

[10] F. Cuppens and R. Ortalo. LAMBDA: A Language to Model a Database for Detection of Attacks. In *Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, France, October 2000.

[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*, 3, 2007.

[12] P. Gama and P. Ferreira. Obligation Policies: An Enforcement Platform. In *6th IEEE Policy*, June 2005.

[13] W. Kanoun, N. Cuppens-Boulahia, F. Cuppens, and F. Autrel. Advanced reaction using risk assessment in intrusion detection systems. In *2nd International Workshop on Critical Information Infrastructures Security*, 2007.

[14] 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*, 10(1/2):5–22, 2002.

[15] P. Ning, Y. Cui, and D. Reeves. Constructing Attack Scenarios Through Correlation of Intrusion Alerts. In *proceedings of the 9th ACM conference on Computer and communication security*, pages 245–254, Washington DC, USA, 2002.

[16] M. Petkac and L. Badger. Security agility in response to intrusion detection. December 2000.

[17] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol, RFC 3261. Available at: http://www.ietf.org/rfc/rfc3261.txt, June 2002.

[18] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 2006.

[19] H. Sengar, D. Wijesekera, H. Wang, and S. Jajodia. VoIP Intrusion Detection Through Interacting Protocol State Machines. In *Proceedings of the 2006 International Conference on Dependable Systems and Networks*, USA, 2006.

[20] N. Stakhanova, S. Basu, and J. Wong. A taxonomy of intrusion response systems. *International Journal of Information and Computer Security*, 1(1/2), March 2007.

[21] T. Toth and C. Kruegel. Evaluating the impact of automated intrusion response mechanisms. In *18th Annual Computer Security Applications Conference ACSAC02*, 2002.

[22] F. Vigna and R. A. Kemmerer. Netstat: A network based intrusion detection system. *Journal of Computer Security*, 7(1):37–71, 1999.