J Comput Virol (2007) 3:195–210
DOI 10.1007/s11416-007-0039-z

INVITED PAPER

Enabling automated threat response through the use of a dynamic security policy

Hervé Debar · Yohann Thomas · Frédéric Cuppens · Nora Cuppens-Boulahia

Received: 20 December 2006 / Accepted: 7 March 2007 / Published online: 30 March 2007

Abstract Information systems security issues are currently being addressed using different techniques, such as authentication, encryption and access control, through the definition of security policies, but also using monitoring techniques, in particular intrusion detection systems. We can observe that security monitoring is currently totally decorrelated from security policies, that is security requirements are not linked with the means used to control their fulfillment. Most of the time, security operators have to analyze monitoring results and manually react to provide countermeasures to threats compromising the security policy. The response process is far from trivial, since it both relies on the relevance of the threat analysis and on the adequacy of the selected countermeasures. In this paper, we present an approach aiming at connecting monitoring techniques with security policy management in order to provide response to threat. We propose an architecture allowing to dynamically and automatically deploy a generic security policy into concrete policy instances taking into account the threat level characterized thanks to intrusion detection systems. Such an approach provides means to bridge the gap between existing detection approaches and new requirements, which clearly deal with the development of intrusion prevention systems, enabling a better protection of the resources and services.

H. Debar · Y. Thomas (✉)
France Télécom R&D, 42 rue des Coutures, 14066 Caen, France
e-mail: yohann.thomas@orange-ftgroup.com

H. Debar
e-mail: herve.debar@orange-ftgroup.com

Y. Thomas · F. Cuppens · N. Cuppens-Boulahia
GET/ENST Bretagne, 2 rue de la Châtaigneraie,
35512 Cesson-Sévigné, France
e-mail: frederic.cuppens@enst-bretagne.fr

N. Cuppens-Boulahia
e-mail: nora.cuppens@enst-bretagne.fr

1 Introduction

Managing information systems requires to make a compromise between multiple parameters, one of them being security. Although security is of crucial interest, constraints such as performance and convenience are to be strongly considered. In particular, being able to serve large numbers of users concurrently or to maintain acceptable response times, while keeping capital (hardware and software) and operations (manpower) expenses under control, is a major issue. Moreover, ease of use and automation are frequent requirements to provide better service to users.

Nowadays, the compromise between these multiple adjustment variables is generally defined statically at design time. However, security is not static, since new vulnerabilities, new users and usages, and new attackers continually appear. This is also true for the other variables. In particular, it is essential to reflect the evolution of the information system through an up-to-date view of hardware and software, which impact both performance and convenience, and thus maintain a dynamic balance between the different requirements.

Consequently, the compromise needs to change over time, in particular to respond to threat. This paper describes a mechanism for threat management at the security policy level. The security policy is dynamically updated with respect to current threats. Our mechanism enables management and deployment of dynamic security policies with safeguards, and we provide an architecture to deploy such policies, which can be dynamically enforced to ensure that the best compromise is always met.

1.1 Intrusion prevention and threat response

Intrusion detection systems now belong to the arsenal of mainstream security tools and are deployed within organizations to monitor the information system operations and report security threats. While many issues have been highlighted with the diagnosis proposed by intrusion detection systems, the technology has matured sufficiently to tackle the problem of intrusion prevention. In particular, correlating alerts with the inventory of the hosts [1] allows to better characterize intrusions, through correlation with vulnerabilities, alert severity mitigation, and false positive recognition. The objective of intrusion prevention is not only to detect threats but also to block them, to prevent the attacker from building upon its advantage and further propagating within the information system, and this has been forecasted for quite some time [2].

Intrusion prevention currently means that when an alert is triggered, a mechanism is activated to terminate the network connection or the process associated with the event. Network-based intrusion prevention devices effectively act as application level firewalls, adding the capability to block traffic based on packet content in addition to headers and connection context. Host-based intrusion prevention software has the capability to terminate a process that is trespassing or abusing its privileges, as shown by [3], but is limited to a single machine. In many cases, the time to react is so small that the threat response mechanism is implemented very close to the detection mechanism, to ensure that the response is effective in dealing with the threat.

However, response is statically associated with each alert, which leads to undesirable side effects [4]. Previous network-based threat response mechanisms based on connection termination by TCP reset injection have shown that they have undesirable side effects in certain contexts, as shown in RFC 3360 [5] and that including response mechanisms online is a requirement for timely and successful response.

We argue that while threat response in itself is a desirable goal, the implementation of threat response at the intrusion prevention system level yields undesirable side effects. First of all, the response is based on an event analyzed by the intrusion prevention device. This means that for every malicious event, the threat response must be applied; unfortunately, this results in a *default permit* (or *open*) security policy, where only events that trigger an alert during the analysis process will be blocked. More generally, the decision on which the threat response is based is a local decision, which does not take into account other operational constraints. This has two undesirable side effects, (1) operators lacking the global vision of the behaviour of the information system will be reluctant to activate threat response mechanisms, and (2) local responses may interfere with global desired behaviour.

1.2 Comprehensive approach to threat response

The objective of the paper is to propose a more comprehensive approach to threat response. We observe that the deployment of modern information systems and networks is associated with access control technologies, located at critical points of the network. We therefore would like to link the threat detection performed by intrusion detection/prevention systems and the access control mechanisms, to provide an adaptive security policy capable of dynamically adjusting to threats. This comprehensive approach does not compete with the immediate application of threat response mechanisms by intrusion prevention systems, but should take over the application of threat response once the threat is properly characterized.

We assume in this approach that intrusion detection systems and alert correlation techniques allow a clear identification of the threat, including the threat type (typically represented by a set of signatures and references to vulnerability databases), the threat origin (represented in most cases by an IP address), and the threat victim (represented by a host under our control, a process, or any set of components of our information system), as in [6] for example. As shown in [7], it is indeed possible to use configuration information to adapt the detection mechanism to its environment, thus ensuring that contextual information in the alerts is exhaustive and correct. While this assumption may be considered strong given the history of false positives and negatives that has plagued intrusion detection research, we do believe that current intrusion detection systems, both commercial and research prototypes, allow a reasonable identification of the threat, and that they will make sufficient progress that the three parameters on which we rely will be filled with appropriate values.

A lot of work has also been undertaken in the research community to reliably identify attack sources, such as identifying stepping stones, or various trace-back mechanisms. Our approach will be able to use more accurate source information if available, but can also concentrate on the protected assets of the information system, that are also the victims of the threat. Several approaches have been proposed for intrusion response [8,9], but they require the deployment of additional systems; our approach leverages existing security policy enforcement mechanisms, limiting the need for new devices. Finally, threat response has been studied repeatedly in the context of denial-of-service attacks, where the threat impact is related to system availability and not system compromise. While we do not consider availability threats at this stage, as shown in Table 2, we should be able to use DDoS filtering mechanisms as policy enforcement points.

Our proposed approach is based on defining a contextual security policy. The threat response mechanism is implemented as contextual policy rules, which are then applied to the information system when contexts become active. The

aforementioned alert management and correlation platform should therefore, in addition to obtaining synthetic alerts, instantiate the appropriate contexts. We describe the particular security policy followed in our approach in Sect. 2, and then give examples in Sect. 3 of how such a policy can ensure response to threat. We then explain how contexts, and especially threat contexts, are managed in Sect. 4. Section 5 provides the architecture of the threat response system, and we explain in Sect. 6 the information workflow allowing to obtain new policies from alerts characterizing threats. We then present an application of the threat response system to an email environment in Sect. 7 and conclude by discussing issues and future work in Sects 8 and 9.

This paper is an extended version of [10]. Additional content mainly deals with context management and the way alerts are processed to obtain new policy instances responding to threat. In particular, a context algebra is provided to express composed contexts, allowing the definition of fine-grained policy rules. A workflow explaining how new policy instances are obtained from alerts is also given, and response strategy is discussed.

2 Security policy formalism

2.1 Choice of a security policy formalism

Most of current security models such as DAC [11] or RBAC [12] can only be used to specify *static* security policies. When an intrusion occurs, the security administrator has to manually update the policy by removing obsolete security rules or inserting new security rules. Unfortunately, the time required for such a manual update is generally too long to represent an effective way to react to an intrusion. The administrator has also to update the policy again once the intrusion is circumvented to restore the policy in a state corresponding to a non intrusive context. Note that in this paper, we will use the terms *policy rule* and *security rule* indifferently to specify security policy statements.

Our objective is to design a method to help the administrator in these tasks of updating the policy. For this purpose, we need a model to specify security policies that dynamically change when some intrusion is detected. In the absence of intrusion, the policy to be applied corresponds to a *nominal* context. Other contexts must be defined to specify additional security rules to be triggered when intrusions are detected. In fact, a parallel could be drawn with provisional authorizations [13]; contexts are linked to the history of reported intrusions, and activate provisional security rules. Some of these security rules may correspond to *permissions* (positive authorizations) but more often they will represent *prohibitions* (negative authorizations). The prohibitions will be automatically deployed over the information system as a reaction to the intrusion. For instance, this may correspond to automatically insert a new deny rule in a firewall.

Thus, the model to be used must provide means to manage conflicts between permissions and prohibitions. In particular, the policy associated with a nominal context can include *minimal* security requirements. These minimal requirements must not be overridden, even when an intrusion is detected. For instance, they may include minimal availability requirements. Of course, these minimal requirements may conflict with contextual rules associated with the detection of a given intrusion. In this case, simple strategies such as prohibition takes precedence or permission takes precedence will not be appropriate to solve the conflict. Instead, the model must include the possibility to specify high level conflict management strategies to find the best compromise between conflicting rules [14].

The model must also provide an abstract and global view of the security policy. This is the purpose of the Policy Instantiation Engine (PIE, see Sect. 5.1 below) to manage this 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. Unfortunately, most security models do not provide such a clear separation.

In this paper, we suggest using an approach based on the Or-BAC model [15]. In the following section, we briefly present the main concepts used in Or-BAC to specify a security policy and explain why this model is a good candidate to manage the kind of contextual security policies we need to support our proposal.

2.2 The Or-BAC formalism

The concept of *organization* is central in the Or-BAC model [16]. Intuitively, an organization is any entity that is responsible for managing a security policy. Thus, a company is an organization, but concrete security components such as a firewall may be also viewed as an organization.

The objective of Or-BAC is to specify the security policy at the *organizational* level, that is abstractly from the implementation of this policy. Thus, instead of modeling the policy by using the concrete and implementation-related concepts of subject, action and object, the Or-BAC model suggests reasoning with the roles that subjects, actions or objects play in the organization. The role of a subject is simply called a *role* as in the RBAC model. On the other hand, the role of an action is called an *activity* whereas the role of an object is called a *view*.

Each organization can then define security rules which specify that some roles are permitted or prohibited to carry out some activities on some views. These security rules do not apply statically but their activation may depend on contextual

conditions. For this purpose, the concept of *context* is explicitly introduced in Or-BAC. Thus, using a formalism based on first order logic, security rules are modeled using a 6-places predicate:

- $security\_rule(type, org, role, activity, view, context)$ where $type$ belongs to $\{permission, prohibition\}$.
  For instance, the following security rule:
- $security\_rule(prohibition, corp, pop\_user, read\_pop, mail\_server, pop\_attack)$.

means that, in organization $corp$, a pop user is forbidden to use the pop service to consult his or her mail in the context of pop attack.

All these concepts, organization, role, activity, view and context, may be structured hierarchically. Permissions and prohibitions are both inherited through these hierarchies (see [17] for more details).

Since a given security policy may include permissions and prohibitions, conflict management strategies have to be defined to solve the possible conflicts. In Or-BAC, such a strategy consists in assigning a priority to each security rule [14]. Priorities define a partial order on the set of security rules so that when a conflict occurs between two rules, preference is given to the rule with the higher priority. Priority assigned to security rules must be compatible with hierarchies defined on entities such as organization, role, activity, view and context. Thus, in case of conflict, if a given security rule is inherited by a given entity, this rule will have lower priority than another security rule explicitly assigned to this entity.

Once the organizational security policy is defined, it is possible to check if the conflict management strategy is *effective*, that is it will solve every conflict at the concrete level (see [15] for further details). Since the Or-BAC model abides to the Datalog restrictions [18], we can prove that it is possible to decide in polynomial time that a conflict management strategy is effective.

The organizational policy is then used to automatically derive concrete configurations of PEPs. For this purpose, we need to assign to subjects, actions and objects, the roles they play in the organization. In the Or-BAC model, this is modeled using the three following 3-places predicates:

- $empower(org, subject, role)$: means that in organization org, $subject$ is empowered in $role$.
- $consider(org, action, activity)$: means that in organization org, $action$ is considered an implementation of $activity$.
- $use(org, object, view)$: means that in organization org, $object$ is used in $view$.

For instance, the fact $empower(corp, alice, pop\_user)$ means that organization $corp$ empowers Alice in role pop_user.

Notice that, instead of enumerating facts corresponding to instances of predicate $empower$, it is also possible to specify role definitions which correspond to logical conditions that, when satisfied, are used to derive that some subjects are automatically empowered in the role associated with the role definition. Activity and view definitions are similarly used to automatically manage assignment of action to activity and object to view. For instance, in a network environment, we can use a role definition to specify that every host in the zone 111.222.1.0/24 are empowered in the role DMZ.

Notice that we shall use Prolog notation to specify Or-BAC security policies. On this purpose, the only important Prolog constructs to remember are that constant values start with a lowercase character, that variables start with an uppercase character, and that _ denotes any value.

2.3 Or-BAC contexts

Regarding contexts, we have also to define logical conditions to characterize when contexts are active. In the Or-BAC model, this is represented by logical rules that derive the following predicate:

- $hold(org, subject, action, object, context)$:
  means that in organization org, $subject$ performs $action$ on $object$ in context $context$.

We say that context c is active in organization org when it is possible to derive $hold(org, s, a, o, c)$ for some subject s, action a and object o.

Using the model, one can then derive concrete authorizations that apply to subject, action and object from organizational security rules. For instance, let us consider Listing 1. In an organization Org, the security rule expresses a $permission$ for a given $Role$ to make a given $Activity$ on a given $View$ in a given $Context$. The predicates $empower$, $consider$ and use indicate that $Role$, $Activity$ and $View$ are respectively abstractions of $Subject$, $Action$ and $Object$ in the considered organization. When the considered $Context$ is being held for $Subject$, $Action$ and $Object$ through the $hold$ predicate, we can thus derive the fact that it is permitted for $Subject$ to make $Action$ on $Object$.

Listing 1 Derivation of concrete authorizations

```
is_permitted(Subject, Action, Object) :-
    security_rule(permission, Org, Role, Activity, View, Context),
    empower(Org, Subject, Role),
    consider(Org, Action, Activity),
    use(Org, Object, View),
    hold(Org, Subject, Action, Object, Context).
```

This general principle of derivation of concrete authorizations from organizational authorizations is used to automatically generate concrete configurations (see [19] for further details in the case of network security policies).

3 Examples

Or-BAC is used to define attack classes thanks to threat contexts. We propose here two examples of threats and explain how response is managed considering activated *hold* predicates and security rules describing the policy to apply in such cases.

3.1 Syn-flooding attack

Imagine a Syn-flooding attack towards a webserver. We use IDMEF messages as explained in Sect. 4.3.1 to say that if a given alert message is received with (1) a classification reference equal to CVE-1999-0116 (corresponding to the CVE reference of a Syn-flooding attack) and (2) the target is attacked through a service whose name is $http$ (or port is $tcp/80$) and (3) the target corresponds to a network node whose name is ws, then the $syn\_flooding$ context is active for $http$ action on ws object. The corresponding translation in Prolog can be found in Listing 2.

Listing 2 Syn_flooding context definition

```
hold(corp, _, http, ws, syn_flooding) :-
    alert(CreateTime, Source, Target, Classification),
    reference(Classification, 'CVE-1999-0116'),
    service(Target, http),
    hostname(Target, ws).
```

Notice that, since in a Syn-flooding attack, the intruder is spoofing its source address, the subject corresponding to the threat origin is not instantiated in the *hold* predicate which is represented by "_".

When an attack occurs and a new alert is launched by the intrusion detection process, (a) new fact(s) $hold(org, s, a, o, c)$ is (are) derived for some threat context c. So, c is now active and the security rules associated with this context are triggered to react to the intrusion.

Notice that our approach provides *fine-grained* reaction. For instance, let us consider a network where a given host ws is assigned to the role web_server. Let us assume that a Syn-flooding attack is detected against this host on port $tcp/80$, which corresponds to service $http$. In this case, we shall derive the following fact:

- $hold(org, \_, http, ws, syn\_flooding)$: means that host ws is now in the threat context $syn\_flooding$ through $http$.

Since the $syn\_flooding$ context is now active, security rules associated with this context are triggered. For instance, let us assume that there is the following security rule:

- $security\_rule(prohib, org, internet, tcp\_service, web\_server, syn\_flooding)$: means that, in the threat context $syn\_flooding, internet$ is prohibited to perform $tcp\_service$ activity on the web_server.

This security rule is triggered once the $syn\_flooding$ context is active. However, only host ws (whose role is web_server) is in the context of $syn\_flooding$ through $http$ (which is a tcp service). As a consequence, the reaction will not close every tcp service from the Internet to every web server. Instead, the reaction in this case will be limited to close $http$ from the Internet to host ws.

3.2 Pop reconnaissance attack

Imagine now that an internal attacker is attempting a reconnaissance attack on a pop3 server in order to determine valid users. The reference CVE-2005-1133 is an instance of such an attack for a pop3 server in IBM iSeries AS/400.

The definition of the pop_attack context says that if a given alert message is received with (1) a classification reference equal to CVE-2005-1133 (corresponding to the CVE reference of a pop reconnaissance attack) and (2) the target is attacked through a service whose port is $tcp/110$ (or name is $pop3$) by (3) a source that corresponds to a mail user whose name is $charlie$ and (4) the target corresponds to a network node whose name is ms, then the pop_attack context is active for $charlie$ subject making $tcp/110$ action on ms object. Notice that we face here an internal attack and we consider that the diagnostic has revealed that the source is not a decoy, so we are able to instantiate the subject being the source in the *hold* predicate. The corresponding translation in Prolog can be found in Listing 3.

Listing 3 Pop_attack context definition

```
hold(corp, charlie, 'tcp/110', ms, pop_attack) :-
    alert(CreateTime, Source, Target, Classification),
    reference(Classification, 'CVE-2005-1133'),
    hostname(Source, charlie),
    process(Target, 'tcp/110'),
    hostname(Target, ms).
```

In this case, we shall derive the following *hold* fact:

- $hold(org, charlie, tcp/110, ms, pop\_attack)$: means that host ms is now in the threat context pop_attack through port $tcp/110$, the attacker being user $charlie$.

Since the pop_attack context is now active, security rules associated with this context are triggered. For instance, let us assume that there is the following security rule:

- $security\_rule(prohib, org, mail\_user, read\_pop, mail\_server, pop\_attack)$: means that, in the threat context pop_attack, a $mail\_user$ is prohibited to perform $read\_pop$ activity on the $mail\_server$.

This security rule is triggered once the pop_attack context is active. However, only host ms (whose role is $mail\_server$) is in the context of pop_attack through port $tcp/110$ (or $pop3$ service, which are $read\_pop$ actions) for subject user $charlie$ (which belongs to the $mail\_user$ role). Alike the previous example, the reaction in this case will be limited to forbid port $tcp/110$ to host ms, but for user $charlie$ only.

These two examples illustrate the fact that, in our approach, we can associate threat contexts with *general* security rules. However, fine-grained instantiation of the intrusion can be used to limit the reaction to those entities that are involved in the attack (as an intruder or a victim). Notice that the presented listings could be generalized by replacing constants by variables. For instance, in Listing 2, it is possible to replace the constant ws by a variable, and similarly for constants $charlie$ and ms in Listing 3.

4 Context management

The central idea of our proposal is based on using contexts to model how to dynamically update the security policy when a threat is detected. Therefore, the core of our proposal is to manage contexts according to threat information. Note that we generically talk about *threat* contexts to refer to contexts used to characterize threats and to provide threat response. Thus, examples of threat contexts may in fact refer to attacks or intrusions (successful attacks). For instance, *syn_flooding* is considered as a threat context, and *pop_attack* is considered as a threat context.

We present in this section how we define atomic and composed contexts, and how we aim at activating and deactivating these contexts according to threat level.

4.1 Atomic contexts

We consider that contexts may belong to three categories: *operational*, *threat* and *minimal*. Let C be a set of contexts. We consider a set $OC \subseteq C$ of *operational* contexts. For the sake of simplicity, we consider that, in the absence of characterized threat, that is in the absence of attack or intrusion, the organizational policy is defined using a single *nominal* context. Thus, we assume that $nominal \in OC$. However, in a more realistic setting, this policy may depend on other contexts, for instance temporal contexts. Thus, we assume that OC may contain additional sub-contexts, and that for example, $working\_hours \in OC$. Additional details about *operational* contexts may be found in [20]. Note that $c \in OC$ *is active* does not mean that there is no attack or intrusion, but that *it is possible that there is no attack or intrusion*. Indeed, *operational* contexts do not provide any information about threats. For example, $nominal$ is always active, and $working\_hours$ only relies on time. We then consider a set $TC \subseteq C$ of *threat* contexts. A context $c \in TC$ is activated when a given threat is detected. This means that $c \in TC$ *is active* necessarily implies that *there is an attack or an intrusion*. It is associated to a set of new security rules that apply to fix the threat. Finally, we consider the set $MC \subseteq C$ of *minimal* contexts. Minimal contexts aim at defining high priority exceptions in the policy, allowing to describe minimal security requirements that must apply even when intrusions occur.

Contexts are organized hierarchically so that, when a conflict occurs, security associated with contexts higher in the hierarchy will override security rules associated with lower contexts. We assume that *operational* contexts are lower than *threat* contexts which are in turn lower than *minimal* contexts. However, potential conflicts may still remain between rules associated with contexts belonging to the same category. In such cases, a partial order has to be defined between concerned rules, in order to ensure conflict resolution at the policy evaluation level (see Sect. 5.3).

If c is an *threat* context, then subject s, action a and object o must be correctly mapped onto information available from threats, including threat source, threat classification and threat target. So, in that case, the context definition associated with c is a logical condition that matches the alert message generated by the intrusion detection process.

4.2 Composed contexts

Providing the possibility to express fine-grained contexts is of major interest, in particular to characterize threats. However, managing specific atomic contexts would rapidly become difficult since it would result in a huge number of definitions. On this purpose, a context algebra is defined to provide a way to combine atomic contexts through a boolean algebra. The following basic functions are provided on this purpose to manipulate composed contexts:

$$\begin{aligned} Negation &: n(c) \leftrightarrow context\ c\ is\ \textbf{not}\ active \\ Conjunction &: \&(c1, c2) \leftrightarrow context\ c1\ \textbf{and}\ context\ c2\ are\ active \\ Disjunction &: v(c1, c2) \leftrightarrow context\ c1\ is\ active\ \textbf{or}\ context\ c2\ is\ active \end{aligned}$$

This algebra allows the expression of composed contexts based on the composition of atomic contexts, ensuring thus an easy way to define fine-grained security rules.

However, managing security rules with composed contexts requires to be able to deal with such context priorities. On this purpose, we now analyze the possible combinations, giving examples for a better understanding of composed context priorities. Contexts entering in the composition of *composed* contexts are simply named *composing* contexts.

Definition 1 The negation of a context, whatever its category, is an operational context because it is possible that there is no attack or intrusion in a negative context.

Property 1 *The priority of a negative context is equal to the priority of an operational context. Consequently, the priority of a negative context is lower than the priority of a threat context, and lower than the priority of a minimal context.*

Let us consider $c1 \in OC$, $c2 \in TC$ and $c3 \in MC$. According to Definition 1, one can state that $n(c1) \in OC$, $n(c2) \in OC$ and $n(c3) \in OC$. Now, according, to Property 1, one can state that $n(c1)$, $n(c2)$ and $n(c3)$ have a priority of an operational context. Thus, they have a lower priority than threat and minimal contexts.
Ex. *$n(working\_hours)$, like $working\_hours$, is an operational context; $n(pop\_attack)$, negation of pop_attack, is an operational context. Thus, $n(working\_hours)$ and $n(pop\_attack)$ have both a lower priority than pop_attack, which is a threat context.*

Definition 2 The conjunction of two contexts belonging to the same category belongs to this category.

Property 2 *The priority of the conjunction of two contexts belonging to the same category is the priority assigned to this category.*

Let us consider $c1 \in OC$ and $c2 \in OC$. According to Definition 2, one can state that $\&(c1, c2) \in OC$. Now, according to Property 2, one can state that $\&(c1, c2)$ has the priority of an operational context.
Ex. *$\&(working\_hours, in\_dmz)$ is the conjunction of a temporal (thus, operational) and a spatial (thus, operational) context. Consequently, $\&(working\_hours, in\_dmz)$ is an operational context.*
Now, let us consider $c3 \in TC$ and $c4 \in TC$. One can state that $\&(c3, c4) \in TC$ and that its priority is higher than operational, but lower than minimal.
Ex. *$\&(pop\_attack, syn\_flooding)$ is the conjunction of two threat contexts. Consequently, $\&(pop\_attack, syn\_flooding)$ is a threat context.*

Definition 3 The conjunction of two contexts belonging to different categories belongs to the category of the composing context having the highest priority.

Property 3 *The priority of the conjunction of two contexts belonging to different categories is the highest priority of the composing contexts.*

Let us consider $c1 \in TC$ and $c2 \in OC$. According to Definition 3, one can state that $\&(c1, c2) \in TC$. Now, according to Property 3, one can state that $\&(c1, c2)$ has the priority of a threat context.
Ex. *$\&(pop\_attack, working\_hours)$ is the conjunction of a threat context and an operational (temporal) context. Since an threat context has a higher priority than an operational context, $\&(pop\_attack, working\_hours)$ is a threat context, and thus it has the priority of a threat context.*

Dealing with the disjunction is not so trivial, in particular with two contexts belonging to different categories. Indeed, let us consider $c1 \in OC$ and $c2 \in TC$. Determining to which category $v(c1, c2)$ belongs requires to consider which composing context among $c1$ and $c2$ is activating $v(c1, c2)$, since $c1$ and $c2$ do not have the same priority. Indeed, if $v(c1, c2)$ is active because $c1$ is active, this means that $v(c1, c2)$ is an operational context, like $c1$. On the contrary, if $v(c1, c2)$ is active because $c2$ is active, this means that $v(c1, c2)$ is a threat context, like $c2$. Moreover, it is possible that $v(c1, c2)$ is active because $c1$ and $c2$ are both active. In this case, $v(c1, c2)$ belongs to the category of the composing context having the highest priority.

In order to avoid the issue of active context determination, we make the choice of automatically splitting the security rules defined with a disjunctive context into a set of equivalent rules, each one being defined for each composing context of the disjunction. For this purpose, we have simply to observe that a security rule defined with a disjunctive context $v(c1, c2)$ is logically equivalent to the conjunction of two security rules respectively defined with context $c1$ and with context $c2$. Therefore, we first convert contexts to Disjunctive Normal Form (DNF), that is as a disjunction of conjunctions, and then write the set of equivalent rules.

Composed contexts are not necessarily composed of atomic contexts. Based on the defined algebra, it is possible to envision not only the composition of atomic contexts, but also the composition of composed contexts, so that one can define rules triggered by fine-grained contexts expressing accurately the security requirements. For instance, one could express a prohibition for a role $user$ to make the activity $read\_pop$ on the view $mail\_server$ in the context:

$$\&(v(remote\_access, \&(internal\_access, n(working\_hours)), pop\_attack))$$

that is either in a context of pop attack **and** remote access, **or** in a context of pop attack **and** internal access on non-working hours.

4.3 Context activation

Activation of threat contexts raises two major points: (1) which information is available to characterize threats, and (2) what do we do with this information to characterize threats at the policy level. We should insist on the fact that when we talk about context activation, we deal in fact with the activation of complete *hold* facts, that is context, but also organization, subject, action and object. This allows a full characterization of the threat, that is not only which kind of threat (e.g. context *pop_attack*), but also which subject, action and object it deals with, and within which organization.

4.3.1 Information about threat

IDMEF (Intrusion Detection Message Exchange Format [21]) messages generated by intrusion detection sensors naturally carry threat information. Even outside intrusion detection, IDMEF provides an appropriate format for describing log events, as shown for example by the Prelude IDS framework.[1] Therefore, we use IDMEF messages to select contexts and policy rules to activate. Among the IDMEF message attributes, we particularly use:

CreateTime The CreateTime timestamp indicates the time at which the alert was created and is mostly relevant for context activation.

Assessment The Assessment attribute carries information related to the risk of the attacker's actions.

Classification The Classification provides information about the mechanism of the attack. This is important to relate the alert to the views and activities of the Or-BAC policy rules, and to activate contexts.

Target The Target attribute carries information about the victim. This is important to relate the alert to the views and activities of the Or-BAC policy rules, and to activate contexts.

Source The Source attribute carries information about the attacker. This may be relevant for roles in the Or-BAC policy rules if the attacker is an insider, and to activate contexts.

We use the two first attributes to compute a context lifetime, as shown in Table 2. Attributes are also translated into contexts through the use of mapping functions, as shown in Sect. 4.3.2.

[1] http://www.prelude-ids.org/.

4.3.2 Mapping alert information on hold predicates

Mapping alert information to context requires creating transformations from alert content to instantiated triples ($Subject$, $Action$, $Object$) by writing the appropriate *hold* predicates. Unfortunately, the naive mapping from *IDMEF.Source* to *Subject*, from *IDMEF.Classification* to *Action*, and from *IDMEF.Target* to *Object*, is far from sufficient, and this for three reasons:

1. 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.
2. Alert information is sometimes incomplete; sources can be inexistent, incomplete or wrong. Multiple classifications may provide inconsistent information, such as conflicting attack references, may cover multiple attacks, or may not be modeled in our system. We need to specify what happens when an alert is incomplete.
3. We also need to specify complex responses mechanisms, that take into account environmental information, expressing complex reaction scenarios. For example, a complete response system may require moving from HTTP to HTTPS, and hence opening and closing multiple network accesses, and starting and stopping multiple services.

Table 1 lists the elements which should be taken into account to provide relevant mappings. 'X' means that the considered information is very likely to be found in the IDMEF messages and thus to be used in the mappings. 'x' means that the information is less likely to be found, or that it is not yet used in the mappings. Note that not all IDMEF parameters are taken into account in this table, but we are currently investigating a complete and systematic mapping of every IDMEF parameter.

The table reveals for instance that not only *Classification.Reference* can be considered to instantiate contexts, but also *Target.Process* and *Target.Service*. In fact, some information may be redundant. For instance, both the reference CVE-2005-1133 and the target service $tcp/110$ can be used to diagnose a pop_attack context. This kind of redundancy can help detecting conflicting attack references, and is also used to determine contexts even in case of missing information. For example, an alert with a missing reference but a target port could be managed considering only the target port. However, one has to note that such information are not necessarily rigorously equivalent, since one may look for more precise evidences. For example, CVE-2005-1133 not only inform that we face a pop threat, but also that it is a reconnaissance attack, which can be of interest in the mapping

Table 1 Mapping IDMEF classes on Or-BAC parameters

	Subject	Action	Object	Context	Lifetime
Createtime					
ntpstamp				X	
Source					
Node.name	X				
Node.Address.address	X				
Node.Address.netmask	x				
User.Userid.name	X				
Process.name	x			x	
Service.name	x			x	
Service.port	x			x	
Target					
Node.name			X		
Node.Address.address			X		
Node.Address.netmask			x		
User.Userid.name			x		
Process.name		X	x	x	
Service.name		X	x	x	
Service.port		X	x	x	
Classification					
Reference.name		x		X	
Assessment					
Impact.severity					X
Impact.type					X

process. On the opposite, target port $tcp/110$ only provides means to derive that we are coping with pop threat.

This mapping also takes into account organization-related policies for response. For example, mappings may always ignore *IDMEF.Source* information, concentrating on blocking traffic that reaches *IDMEF.Target*. They may prefer system-related information (host names or network addresses) to user names, to ensure a global response to the threat, or prefer user names to deliver extremely targeted responses at the user account level.

4.4 Context deactivation

Deactivating threat contexts is used to revocate countermeasures once threats are no longer present. On this purpose, we currently manage static context lifetimes, which are computed thanks to IDMEF alerts assessment attributes. Indeed, IDMEF alerts provide an *IDMEF.Assessment.Impact* attribute (denoted in dotted notation to follow the IDMEF class hierarchy) with three sub-attributes, severity, completion and type. If completion is set to failed, no context will be activated. Otherwise, based on the impact severity, and type, we derive the duration of the context activity, according to the matrix defined in Table 2.

When an alert occurs, it is asserted for a certain duration. Thus, the corresponding context is activated with the expiration date set according to the table. While this alert remains stored in the system, the context remains active. When the lifetime expires, the alert is removed from the database, and the context is deactivated, unless another instance of the alert has been received in the meantime. Both asserts and retracts trigger a re-evaluation of the security policy.

The values of Table 2 have been defined through expert knowledge of the risks incurred by each protocol. We currently use the same matrix for evaluating the risk incurred by each access mechanism; the variation in risk associated with each individual protocol is handled by the proper setting of the impact severity attribute.

4.5 Influence of mapping on the response strategy

The mapping from alerts to contexts (or more generally to Or-BAC *hold* facts) also influences the response strategy. Depending on the information available, one may provide a network-oriented response by retaining only network-based information such as IP addresses and port numbers and discarding user-based information such as user names, or conversely provide a user-oriented response. One may also combine both for a very specific response. In a number of cases, network-oriented response may be the only practical option, as network information is available in the alerts and network security devices such as firewalls are capable of blocking the undesired traffic.

Also, mapping influences the response to be either victim-centric or attacker-centric. A victim-centric response aims at blocking traffic towards the attack target, assuming that other attackers may attempt to exploit the same attack mechanisms. An attacker-centric response aims at blocking traffic from the attack source, ensuring that the attacker is prevented from accessing other servers that may offer the same service or vulnerability, as is often the case in large environments—indeed, our own case study shows three mail servers with identical characteristics; an attack on one of them is equally dangerous for the two others, even though the attacker may not have yet stricken.

Finally, one may degrade the mapping, for example by authorizing a mapping from IP addresses to subnet masks only. Hence, the response would apply to all machines in the subnet, instead of the single victim machine.

5 The threat response system

5.1 System architecture

The architecture of the threat response system is presented in Fig. 1. Software or hardware modules are depicted by circles and messages and configuration information associated with our components by diamonds. We assume that any organization will deploy sensors and a security information management framework, from which we will collect alert information. This is depicted by the *sensor* block. The policy changes will be applied to *PEPs*, for example mail servers,

Table 2 Intrusive context lifetime according to IDMEF impact severity and type, in minutes

Impact severity Impact type	Info	Low	Medium	High	Comment
Admin	1	2	4	8	This is the most severe case.
Dos	0	0	0	0	We are not currently handling DoS attacks.
File	0	1	2	3	
Recon	0	0	0	0	We are not currently handling scans, as they do not result in compromise.
User	0	1	2	4	
Other	0	0	1	2	

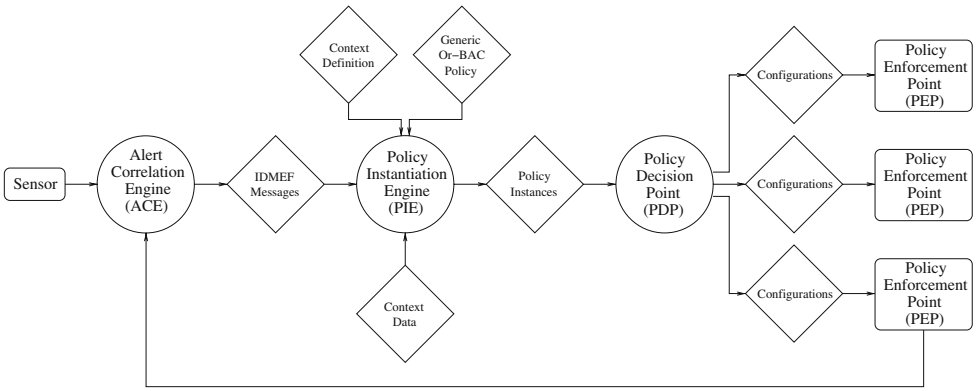

Fig. 1 Threat response system architecture

firewalls or intrusion detection systems. It is therefore likely that some PEPs will also act as sensors. The function of our software modules is described further in Table 3.

5.2 Alert correlation engine (ACE)

Generally, information produced by sensors cannot be considered on their own. Indeed, this information actually comes from many sources (sensors), and with different formats (ex: a Snort alert, a Netfilter firewall log, etc.). Moreover, there is a strong need for alerts volume reduction and semantics improvement. Alert correlation aims at realizing this task, thus permitting false positives reduction and producing meta-alerts offering a better semantics and severity levels for more efficient analysis. This is mainly done by merging redundant information and similarities in order to obtain global alerts with a fusion process [22]. We define an ACE as an entity receiving as input every possible event produced by sensors and giving as output high-level IDMEF-compliant alerts (meta-alerts).

Note that the exact definition of this module is considered out of scope for this paper, since we consider the existence of valuable works on the subject [22–25] and of a SIM commercial market as a proof of feasibility. Our current ACE prototype only verifies and modifies impact information in the IDMEF message, and validate sources and targets with respect to contexts.

5.3 Policy instantiation engine (PIE)

The security policy description is ensured by a set of Or-BAC rules. The possibility to express contextual policies offered by Or-BAC is used in order to trigger rules considering high-level and fine-grained information. Thus, a policy instantiation engine (PIE) has two major functions: (1) activate contexts (through Or-BAC *hold* facts) which (2) trigger re-evaluation of the security policy (through activation of abstract Or-BAC rules). Intrusive contexts activation is addressed in Sect. 4.3. For operational contexts, such as temporal ones, one can refer to [20] for further information. Note that the PIE also deals with context deactivation, according to Sect. 4.4. Generic policy rules triggering is explained in Sect. 2, and examples are given in Sect. 3.

Note that the PIE manages conflict resolution at the policy evaluation level to produce a coherent set of policy instances (concrete Or-BAC rules) to deploy. Conflict resolution is ensured at the abstract level, that is between Or-BAC security rules. It consists in deciding which rule takes precedence when two or more rules present intersections of roles, activities, views and/or contexts. On this purpose, we consider Or-BAC abstract entities inheritances and priorities depending on contexts categories to define a partial order relationship between conflicting rules, which is sufficient to ensure the proper evaluation of the security policy, as shown in [26].

Table 3 Function of the software 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	Or-BAC concrete rules	Or-BAC policy and context definitions	Activate threat contexts. Extract a new security policy from the active contexts.
PDP	Or-BAC concrete 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.
PEP	Config scripts	IDMEF messages		Apply the configuration script that implements the security policy.

5.4 Policy decision point (PDP)

Policies instantiated in response to threat contexts are transmitted to one or more PDP(s). A PDP is in charge of local policy decisions. Whenever it receives a new policy instance, that is an Or-BAC concrete rule (permission or prohibition), a PDP has to map this information onto concrete actions to produce on PEPs to enforce the new policy. A PDP thus have to be aware of its PEPs abilities, 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, considering the implementation of the PEP (e.g. a "Netfilter" firewall) [19]. Note that part of the decisional capability of the PDP relies on the fact that a given Or-BAC 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. One may also imagine advanced deployment scenarios, taking into account network or application sessions continuity. For example, an advanced scenario could be to first alert users on an imminent service disruption, but let them a definite time to terminate their immediate action.

5.5 Policy enforcement point (PEP)

PEPs receive new policies (or policy elements), which have been translated by the PDP [19]. Expressing a new policy may have implications on multiple PEPs. For example, it can involve both a server (stopping a service) and a firewall (blocking a port). Each PEP dealing with a policy instance is sent a configuration script, considering its type (ex: firewall), but also its implementation (ex: Netfilter). Note that a PEP can also be considered a sensor, which possess specific functionalities of policy enforcement. This characteristic can provide information allowing validation of new policies effective application.

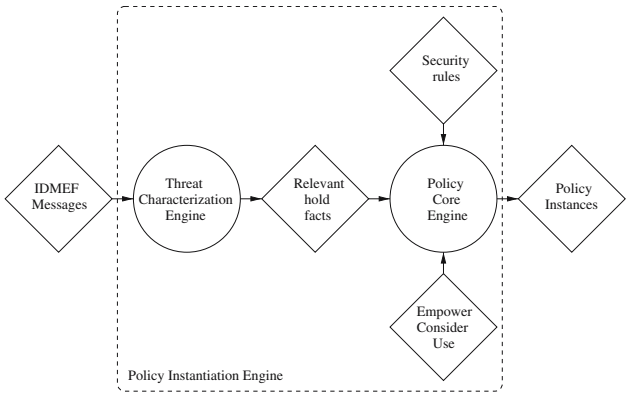

Fig. 2 Information workflow, from alerts to new policy instances

6 From alerts to new policies

We present here the workflow allowing the mapping from alerts to new policy instances. The PIE is divided into two subparts allowing (1) to map alerts reported as IDMEF messages into Or-BAC hold facts characterizing threats and allowing adequate countermeasures, and (2) to derive new policy instances thanks to hold facts and to the abstract policy definition. Figure 2 presents a global view of these two PIE functions. Mapping threats to hold facts is managed through the Threat Characterization Engine (TCE), whereas concrete policy instantiation is realized thanks to the Policy Core Engine (PCE). Note that conflict resolution is managed at PCE level since it is ensured at the policy evaluation step.

Figure 3 presents in details the components of the Threat Characterization Engine. Threat characterization does not actually consists in a trivial and static mapping, since (1)

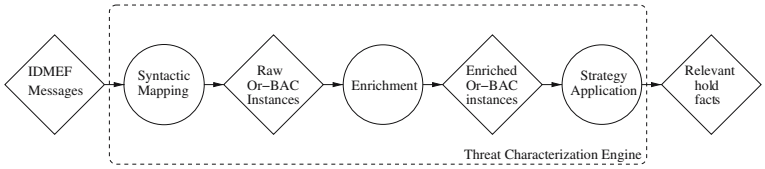

Fig. 3 Threat characterization engine

IDMEF messages may contain various information which can be translated in different Or-BAC triples (subject, action, object), (2) some information may lack in IDMEF messages, and (3) generated hold facts must be relevant to current threat in order to provide the best adequate response. On this purpose, the TCE process is divided into three steps: (1) syntactic mapping, (2) enrichment, and (3) strategy application.

6.1 Syntactic mapping

The first stage consists in realizing a quite trivial syntactic mapping, that is extracting as many triples ($subject$, $action$, $object$) as possible from a given IDMEF message. The obtained information is called *raw Or-BAC instances*, since they are not usable to respond to threat. Such mappings are statically defined, a $subject$ being for instance the IP address of the source host given by the IDMEF message. An example of an $action$ is the target port (service) and an example of an $object$ is the DNS name of the target.

6.2 Enrichment

Once syntactic mapping has been realized, we face two issues linked with the fact that alerts are sometimes incomplete, or that some optional parameters are not necessarily defined. Thus, two enrichment steps are provided: (1) enrich subjects, actions and objects which are only partially instantiated, and (2) find similar actions to instantiated ones, but at different levels (e.g. network and service levels, as aforementioned in Sect. 5.4).

1. Subjects, actions and objects are in fact data structures sometimes regrouping equivalent information. For instance, IP addresses and DNS names are considered equivalent since they qualify the same subject. Another example is the equivalence between a service port (tcp/110) and a service name (pop3). Consequently, and because we may sometimes find in IDMEF messages only part of these information, a first enrichment step consists in finding all equivalences and thus provide exhaustive subjects, actions and objects.
2. Actions may present similar instances, but at different levels. In particular, we distinguish network actions, such as $tcp/110$, from service actions, such as $popd$. These information both aim at responding to the pop3 service, but the former is probably to result in a firewall reconfiguration (network response, for instance blocking tcp/110 port) and the latter may trigger a server reconfiguration or stopping (service response, for instance stopping /etc/popd daemon). Since we may find only part of these information in the alerts, enrichment also consists in trying to instantiate all the similar instances.

6.3 Strategy application

At this stage, the system has provided *enriched Or-BAC instances*, that is all possible and exhaustive ($subject$, $action$, $object$) triples characterizing the threat.

Strategy application first consists in triggering the context. On this purpose, and although it does not appear on the figure, this process needs IDMEF information related to context triggering (for instances, *Classification.Reference*).

Then, considering all available information (IDMEF message and enriched Or-BAC instances), and optional information related to user-defined strategy settings, the strategy application process deals with instantiation of relevant $hold(org, s, a, o, c)$ facts in order to respond to the considered threat. This means that subject, action, and object may be altered considering desired strategy, to tune the response scale (e.g. extense to a group of users, a sub-network, etc.).

Finally, given the obtained $hold$ fact(s), the PIE is able to find the associated $security\_rule$(s), which allow(s) then to derive concrete authorizations as explained in Listing 1.

7 Case study: e-mail server

The case study is the email environment of our organization. The objective of the adaptive security policy is to preserve access to email information, but not necessarily via the same protocol. Email is a fairly critical service hosted on three exchange servers, which can be accessed by three different mechanisms, the native outlook to exchange, pop and imap. In normal operation, all these three modes are active and allow parallel access to the same information. Messages read and sent by one mechanism are also altered by the other mechanisms. We use SWI-Prolog to implement first-order logic based reasoning required by Or-BAC.

7.1 Description of the policy components

The description of the case study and the policy components that we need to develop for this case study are presented in Fig. 4. Ellipses represent abstract information in Or-BAC (organizations, roles, activities, views and contexts) and dashed rounded square boxes represent concrete instances linked by the *empower*, *consider* and *use* facts.

This case study is built upon the architecture of our email service, serving over 5000 users in multiple physical locations. The email service is hosted on three exchange servers, protected by a specific firewall, as shown in Fig. 4a. Users have three channels for accessing email, the classic pop and imap protocols with their application of choice and outlook using the proprietary exchange protocol. All three are kept synchronous, and changes in the same account using one of the access mechanisms are immediately seen using the others. While this case study is limited in scope—a number of equipments do not appear on the schema, such as active directory authentication servers and DNS servers—it provides a sound basis for description and development of the technology.

The *organizations* are the corporation and its different branches. The *roles* graph of this case study is quite simple, since it only deals with users, differentiating mail users from the system administrator. Note that roles are here limited to users, but in a larger case study, we would as well have hosts playing specific roles, such as *workstation*, *web-proxy*, etc. The *activities* graph is limited to reading mail activity, declined along the three possible protocols. Note that all possible activity instances are not represented in the figure. The *views* graph is trivial, since the only necessary view in the use case deals with mailserver description. Finally, the *contexts* graph defines the different contexts that can be activated and are used in defining policy rules. Note that according to Sect. 4, contexts are divided into three categories: *operational*, *threat* and *minimal*. Among operational contexts, the nominal context is always active and defines the security policy that offers the most convenience to users. Intrusive contexts define contexts that are activated by the alert correlation engine when alert information is received from intrusion detection systems and when this alert information is related to one of the specific protocols used for email access. Finally, temporal contexts are used in policy rules to constrain minimal mail context, to put the light on availability during working hours, and to focus on confidentiality and integrity during non-working hours.

In the case study, the ACE, PIE and PDP are implemented as Prolog predicates in SWI-Prolog, and the PEP as XSLT transformations. The components of the model (graphs of abstractions and instances) are modeled in a straightforward way using Prolog facts, among them *empower*, *consider* and *use*.

7.2 Definition of the security policy

Listing 4 Email access control policy

```
sr(perm,corp,mail_user,read_exchange,mail_server,&(minimal_mail,working_hours)).
sr(prohib,corp,pop_user,read_pop,mail_server,pop_attack).
sr(prohib,corp,imap_user,read_imap,mail_server,imap_attack).
sr(prohib,corp,outlook_user,read_exchange,mail_server,exchange_attack).
sr(perm,corp,mail_user,read_mail,mail_server,nominal).
```

Following the definitions of Sect. 2, we define the security policy as shown in Listing 4. In this policy, we consider that it should always exist a way to read mail during working hours, but not necessarily on non-working hours. Indeed, although availability is of crucial interest during working hours, it may not be so important during non-working hours, and the priority could be higher for confidentiality and integrity. A solution to this availability issue is to define an exception with a rule permitting for example exchange via outlook access with a high level priority (*minimal* context), as shown in the first rule of Listing 4. Thus, we avoid the case for which the system would close all possible paths to mail, which would lead to self-inflicted denial-of-service.

This security policy then specifies that any attack against one of the email access mechanisms invalidates the access mechanism being attacked, and that by default, mail users have access to all mechanisms to read mail. This simple expression is obtained by taking into account that each rule also applies to children in the graphs.

Note that this concise expression is generic and adaptable to multiple physical architectures. If we had multiple mail servers spread per location instead of a centralized mail server farm, we would express the same policy. However, we would change the deployment strategy at the PDP level and have a different list of PEPs.

Once we have modeled the environment and the security policy, we need to express the *hold* predicates as shown in Listing 5. The $working\_hours$ context is modeled in a straightforward way, as is the $nominal$ context. We define the $minimal\_mail$ context as a sub-context of the $minimal$ context. The context $minimal\_mail$ is active when all three email access mechanisms are attacked. Hence, during working hours, when the $\&(minimal\_mail, working\_hours)$ context is active, the policy expresses that the exchange access is re-opened ensuring continued availability of email information.

Note that we do not necessarily consider only availability in such a case. Indeed, confidentiality and integrity guarantees can also be provided by defining additional constraints. For instance, one could define security rules ensuring that resources are accessed only via a secured protocol. For example, one may choose to switch from pop to $pops$ in the case of pop3, or from $imap$ to $imaps$ in the case of imap, etc. Moreover, it is possible to elevate authentication requirements. For example, users could be forced to use certificates or biometric

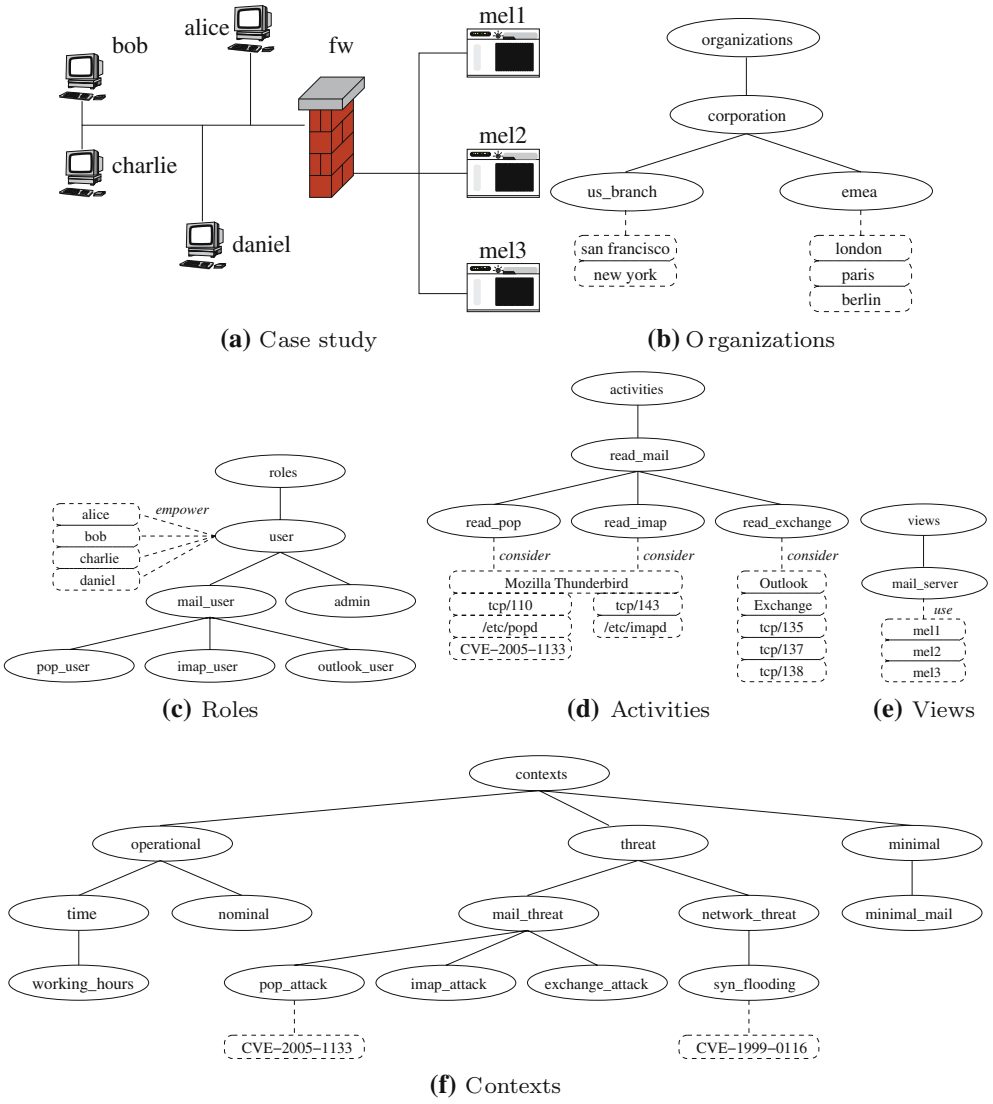

Fig. 4 Description of the policy components

means to authenticate. Thus, availability requirement is still fulfilled, but provided that additional conditions related to confidentiality and integrity are ensured.

7.3 The mapping predicates

The core of the *hold* predicate related to threats (the first one in Listing 5) is represented by the four mapping functions, *trigger*, *map_syntax*, *map_enrichment* and *map_strategy*. The *trigger* function aims at mapping an alert reference on its corresponding context, as explained in Sect. 3. References are thus grouped considering attack classes, which represent threat contexts. The *map_syntax*, *map_enrichment* and *map_strategy* functions are implemented with respect to requirements explained in Sect. 6. An example of mapping in this case study is given by Listing 6.

Listing 5 Email access control policy

```
hold(corp,Subject,Action,Object,Context) :-
    alert(CreateTime,Source,Target,Classification),
    reference(Classification,Reference),
    trigger(Reference,Context),
    map_syntax(Source,Target,RawSubject,RawAction,RawObject),
    map_enrichment(RawSubject,RawAction,RawObject,EnrSubject,EnrAction,EnrObject),
    map_strategy(EnrSubject,EnrAction,EnrObject,Subject,Action,Object).

hold(corp,Subject,_,Object,minimal_mail) :-
    hold(corp,Subject,_,Object,pop_attack),
    hold(corp,Subject,_,Object,imap_attack),
    hold(corp,Subject,_,Object,exchange_attack).

hold(corp,_,_,_,working_hours) :-
    globalclock(DayClock,TimeClock),
    TimeClock >= '07:00:00',
    TimeClock < '20:00:00',
    DayClock != 'saturday',
    DayClock != 'sunday'.

hold(corp,_,_,_,nominal).
```

Note that concerning response strategy, we have chosen here to protect user accounts rather than eliminate attackers. It is thus different from the example given in Sect. 3. For example, if Charlie performs a brute-force attack on Alice's email

password, the *Source.User.Userid.name* will be charlie and the *Target.User.Userid.name* will be alice. According to our mapping, we will block access to Alice's account, not from Charlie's account. This stems from the fact that *Source.User* is rarely instantiated in our alerts, and is often unreliable. Another solution may consist in blocking the source, but at the host level rather than at the user level. However, although this may apply to the case of an internal attack, as explained in Sect. 3, where the actual attacker is reported by the alert, it is not clear whether it would be efficient for an external attack. Indeed, the proxy is seen as the source of the attack from the internal network, and this may lead to the blocking of the proxy, instead of the real source. Such a response would mean that all external hosts are blocked instead of attacker only. Moreover, another issue deals with spoofing, that is react on the source is impossible when the alert reports a spoofed source, since it is not the actual attacker. Such considerations are typical information entering into the process of response strategy. The exact implementation of the mappings predicates is still an area of research; while our case study shows that it is possible to define such mappings, the evaluation of what constitutes the "best" mapping remains to be done.

Listing 6 Possible mapping in the case study

```
subject = IDMEF.Target.User.Userid.name
action  = IDMEF.Classification.Reference.name or IDMEF.Target.Service.{name, port}
object  = IDMEF.Target.Node.{name, Address}
```

8 Issues with the approach

While this approach is still under development, the current work has brought up a number of interesting issues, especially concerning service continuity and dynamicity of policy changes.

Service continuity The first question raised by this approach is service continuity. If connectivity is cut at the network level, clients receive error messages but are not informed automatically about other opportunities to access the information they need. We therefore need to interact with clients to inform them that they should change their access mechanism.

Server-side-only automated redirection is possible only in a limited number of protocols. For example, in a web environment where clients have the opportunity to use both HTTP and HTTPS, we would be able to automatically redirect clients from HTTP to HTTPS by changing the URLs embedded in the web pages returned by the server. When the client clicks on a particular link (assuming that the security policy has not changed in the meantime), he is redirected to the appropriate service. Unfortunately, this opportunity does not seem to exist for email protocols; therefore, we are studying the possibility to configure multiple email accounts on a mail client, and change configurations when needed.

Dynamicity of policy changes System and network administrators are quite conservative when it comes to policy changes. Therefore, we need to discourage rapid changes in policies and oscillations between policies, that would perturb the clients and force them to change their access mechanisms several times during their sessions. Experiments with the matrix shown in Table 2 should clarify this problem and in particular allow us to verify if the proposed timings converge towards the $working\_hours$ policy or leave enough room for multiple simultaneous access methods. Implementing dynamic context deactivation should also prevent from such issues. Indeed, defining static context lifetimes is a first step towards context deactivation, but it requires a strong expertise, and it may not provide the best results, since the threat could be shorter than the resulting countermeasure lifetime, or on the contrary, longer than the resulting countermeasure. Future work shall in part consist in improving the context deactivation process, by making use of information reported by policy enforcement points, acting as sensors, in order to dynamically characterize the state of a considered threat.

9 Conclusion

In this paper, we have proposed a systematic approach to threat response. The approach builds upon Or-BAC, an advanced security policy formalism, to define a contextual security policy that will be applied to the information system. This enables the definition of multiple equilibrium points between security, performance, convenience and compliance objectives. These equilibrium points are expressed as contexts or context combinations of the security policy. The Or-BAC framework includes tools for formally verifying the security policy and for translating the formal security policy into practical configuration scripts that can be applied to policy enforcement points to change the security policy. The expression of the security policy allows the definition of simple responses to each threat, a global and efficient response in the face of multiple threats being computed during the instantiation of the security policy.

The threat contexts vary according to alerts collected by various sensors. These alerts received as IDMEF messages are mapped onto policy subjects, actions and objects and are used to activate specific contexts. The mapping from IDMEF messages to policy entities is complex and has implications on the choice of response that will be available to handle the threat. When a particular context is activated, the new set of policy rules is validated and translated to the enforcement

points. These mechanisms have been implemented and validated on a case study environment. The organization-based approach shows encouraging results and we are confident that deployment at a larger scale will be possible.

Future work includes modeling service continuity, ensuring that clients get continuous access to information seamlessly, defining and evaluating mapping functions to formalize the impact these mapping functions have on threat response choices, and evaluating the performances of the prototype approach with respect to performance and efficiency in threat response.

References

1. Thomas, Y., Debar, H., Morin, B.: Improving security management through passive network observation. In: ARES '06: Proceedings of the 1st International Conference on Availability, Reliability and Security (ARES'06), pp. 382–389. IEEE Computer Society, USA (2006)
2. Brackney, R.: Cyber-intrusion response. In: Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, West Lafayette, IN, p. 413 (1998)
3. Petkac, M., Badger, L.: Security agility in response to intrusion detection. In: Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00), New Orleans, LO, USA, p. 11 (2000)
4. Toth, T., Kruegel, C.: Evaluating the impact of automated intrusion response mechanisms. In: Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC). IEEE Computer Society Press, Las Vegas, NV, USA (2002)
5. Floyd, S.: Inappropriate TCP Resets Considered Harmful. RFC 3360 http://www.ietf.org/rfc/rfc3360.txt. (2002)
6. Cuppens, F., Gombault, S., Sans, T.: Selecting appropriate counter-measures in an intrusion detection framework. In: 17th IEEE Computer Security Foundations Workshop (CSFW), Pacific Grove, CA, USA (2004)
7. Mounji, A., Charlier, B.L.: Continuous assessment of a unix configuration integrating intrusion detection and configuration analysis (1997)
8. Ragsdale, D.J., Carver, C.A., Humphries, J.W., Pooch, U.W.: Adaptation techniques for intrusion detection and intrusion response system. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Nashville, TN, IEEE Computer Society Press, pp. 2344–2349 (2000)
9. Carver, C.A., Hill, J.M., Pooch, U.W.: Limiting uncertainty in intrusion response. In: Proceedings of the 2001 IEEE workshop on Systems, Man, and Cybernetics Information Assurance and Security, United States Military Academy, West Point, NY, pp. 142–147 (2001)
10. Debar, H., Thomas, Y., Cuppens-Boulahia, N., Cuppens, F.: Using contextual security policies for threat response. In: Bueschkes, R., Laskov, P. (eds.) Proceedings of the 3rd Conference on Detection of Intrusions and Malware Vulnerability Assessment (DIMVA 06). Springer, Berlin (2006)
11. Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in operating systems. Commun. ACM **19**(8), 461–471 (1976)
12. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Comput. **29**(2), 38–47 (1996)
13. Kudo, M., Hada, S.: XML document security based on provisional authorization. In: CCS '00: Proceedings of the 7th ACM Conference on Computer and Communications Security, pp. 87–96. ACM Press, New York (2000)
14. Cuppens, F., Cuppens-Boulahia, N., Ghorbel, M.B.: High-level conflict management strategies in advanced access control models. In: Workshop on Information and Computer Security (ICS), Timisoara, Roumania (2006)
15. Miège, A.: Definition of a formal framework for specifying security policies. The Or-BAC model and extensions. PhD Thesis, ENST (2005)
16. Kalam, A.A.E., Benferhat, S., Miège, A., Baida, R.E., Cuppens, F., Saurel, C., Balbiani, P., Deswarte, Y., Trouessin, G.: Organization based access control. In: Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), Lake Como, Italy (2003)
17. Cuppens, F., Cuppens-Boulahia, N., Miège, A.: Inheritance hierarchies in the Or-BAC Model and application in a network environment. In: Sabelfeld, A. (ed.) FCS'04, vol. 31, pp. 41–59 (2004)
18. Ullman, J.D.: Principles of Database and Knowledge Base Systems. Computer Science Press, New York (1989)
19. Cuppens, F., Cuppens-Boulahia, N., Sans, T., Miège, A.: A formal approach to specify and deploy a network security policy. In: 2nd Workshop on Formal Aspects of Security and Trust (FAST), Toulouse, France (2004)
20. Cuppens, F., Miège, A.: Modelling contexts in the Or-BAC model. In: ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference, vol. 416. IEEE Computer Society, (2003)
21. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format. RFC 4765 (2006)
22. Cuppens, F., Miège, A.: Alert correlation in a cooperative intrusion detection framework. In: Proceedings of the IEEE Symposium on Security and Privacy (2002)
23. Dain, O., Cunningham, R.K.: Fusing a heterogeneous alert stream into scenarios. In: Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications, pp. 1–13 (2001)
24. Morin, B., Mé, L., Debar, H., Ducassé, M.: M2D2: A formal data model for IDS alert correlation. In: Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID) (2002)
25. Ning, P., Cui, Y., Reeves, D.S.: Constructing attack scenarios through correlation of intrusion alerts. In: Proceedings of the 9th Conference on Computer and Communication Security (2002)
26. Cuppens, F., Miège, A.: Administration model for Or-BAC. In: On The Move Federated Conferences (OTM'03), Workshop on Metadata for Security, Catania, Sicily, Italy (2003)