

Using contextual security policies for threat response

Hervé Debar¹, Yann Thomas^{1,2}, Nora Boulahia-Cuppens², and Frédéric Cuppens²

¹ France Télécom R&D, 42 rue des Coutures, F-14000 Caen
{herve.debar,yohann.thomas}@francetelecom.com

² GET/ENST Bretagne, 2 rue de la Châtaigneraie, F-35512 Cesson Sévigné
{nora.cuppens,frédéric.cuppens}@enst-bretagne.fr

Abstract. With the apparition of accurate security monitoring tools, the gathered alerts are requiring operators to take action to prevent damage from attackers. Intrusion prevention currently provides isolated response mechanisms that may take a local action upon an attack. While this approach has been taken to enhance the security of particular network access control points, it does not constitute a comprehensive approach to threat response. In this paper, we will examine a new mechanism for adapting the security policy of an information system according to the threat it receives, and hence its behaviour and the services it offers. This mechanism takes into account not only threats, but also legal constraints and other objectives of the organization operating this information system, taking into account multiple security objectives and providing several trade-off options between security objectives, performance objectives, and other operational constraints. The proposed mechanism bridges the gap between preventive security technologies and intrusion detection, and builds upon existing technologies to facilitate formalization on one hand, and deployment on the other hand.

1 Introduction

Information systems are designed to ensure the best compromise between multiple constraints, one of them being security. However, it is frequently the case that security requirements have to be relaxed because of convenience or performance issues. For example, Netcraft recently noted that a number of banks have recently switched from HTTPS to HTTP for their login screen pages. While the login information is still transferred in encrypted form, this move has been prompted by performance issues on busy pages. This is a typical example of finding the equilibrium point between serving more users and maintaining security, which we would like to avoid setting in stone. Therefore, we would like to design security policies that are adaptive in nature, i.e. that in nominal mode ensure that performance or convenience objectives are met, that these objectives are more constrained when threats are detected, and that minimal objectives (typically fulfilling legal requirements, service level agreements (SLAs) or ensuring minimal convenience) are always met.

This paper presents a mechanism for building adaptive security policies, which can then be applied to the information system to ensure that the required security objectives are 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 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. The objective of intrusion prevention is not only to detect threats but also to block them, to prevent the attacker to build upon its advantage and further propagate within the information system, and this has been forecasted for quite some time [1].

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 like classic firewalls, adding the capability to block traffic based on packet content in addition to headers and connection context. Response is statically associated with each alert, which leads to undesirable side effects [2]. 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. 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 [4] 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 operating 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 [5] for example. As shown in [6], 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 attacks 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 [7, 8], 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 1, 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 security policy rules, which are then applied to the information system when the context becomes active. The aforementioned alert management and correlation platform should therefore, in addition to obtaining synthetic alerts, instantiate the appropriate contexts. We will describe the particular security policy followed in our approach in section 2, apply this formalism to threat response in section 3 and present the architecture of the threat response system in section 4. We will present an application of this threat response to a particular system in section 5, and conclude by discussing issues and future work in section 6.

2 Security Policy Formalism

2.1 Choice of a Security Policy Formalism

Most of current security models such as DAC [9] or RBAC [10] can only be used to specify *static* security policies. When an intrusion occurs, the security administrator has to manually update the policy by removing no longer appropriate 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 [11]; 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.

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 section 4.1 below) to manage this global security policy. The PIE will have to clearly separate the global policy from its implementation in the PEP (Policy Enforcement Point). In particular, the conflicts are to be solved at the abstract level before generating PEP's 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 [12]. 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. 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 modelling 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 modelled 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:

- $security_rule(prohibition, corp, user_pop, read_pop, mail, pop_threat)$.

means that, in organization *corp*, a pop user is forbidden to use the pop service to consult his mail in the context of pop threat.

All these concepts, organization, role, activity, view and context, may be structured hierarchically. Permissions and prohibitions are both inherited through these hierarchies (see [13] 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. 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, if a given security rule is inherited by a given entity, this rule will have lower priority than other security rules 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 [12] for further details). Since the Or-BAC model abides to the Datalog restrictions [14], 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 PEP's. 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 modelled 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 $empower(corp, alice, user_pop)$ means that organization $corp$ empowers Alice in role $user_pop$.

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 .

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 $context$.

Contexts can be combined to obtain conjunctive, disjunctive and negative contexts. For this purpose, we introduce the functions $\&$, $|$ and $\bar{}$. If c_1 and c_2 are two contexts, then $\&(c_1, c_2)$ is a conjunctive context, $|(c_1, c_2)$ is a disjunctive context and \bar{c}_1 is a negative context. We shall actually use the infix notations $c_1\&c_2$ and $c_1|c_2$ in place of the prefix notations $\&(c_1, c_2)$ and $|(c_1, c_2)$.

A conjunctive context $c_1\&c_2$ is active if both contexts c_1 and c_2 are active. A disjunctive $c_1|c_2$ is active if context c_1 is active or context c_2 is active. Finally, a negative context \bar{c} is active when context c is not active.

Using the model, one can then derive concrete authorizations that apply to subject, action and object from organizational security rules. This general principle of derivation of concrete authorizations from organizational authorizations is used to automatically generate concrete configurations (see [15] for further details in the case of network security policies).

3 Application of the Or-BAC Formalism to Threat Response

The central idea of our proposal is based on using contexts to model how to dynamically update the security policy when an intrusion is detected. Therefore, the core of our proposal is to manage contexts according to threat information.

3.1 Contexts Expression

Let C be a set of contexts. We assume that $nominal \in C$. The *nominal* context defines the security policy when no intrusion is detected³. We then consider a set $IC \subseteq C$ of *intrusion contexts*. A context $c \in IC$ is activated when a given intrusion is detected. It defines the new security rules that apply to fix the intrusion. There is also a context $minimal \in C$ that defines 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 the *nominal* context is lower than intrusion contexts in IC which are in turn lower than the *minimal* context. We can also define that some intrusion contexts are lower than some other intrusion contexts. Since several intrusion contexts may be active in parallel, this is useful to solve possible conflicts between intrusion contexts.

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 . If c is an intrusion context, then subject s , action a and object o must be respectively mapped onto the threat source, the threat classification and the 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.

For instance, see Listing 1.1 for the *syn_flooding* context definition, using IDMEF messages as explained in Section 3.2. This definition says 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 given service whose name is *Action* (for instance http) and (3) the target corresponds to a network node whose name is *Object*, then the *syn_flooding* context is active for this *Action* and *Object*. 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 $hold(org, s, a, o, c)$ is derived for some intrusion context c . So, c is now active and the security rules associated with this context are triggered to react to the intrusion.

³ For the sake of simplicity, we assume that, in the absence of intrusion, the organizational policy is defined using a single *nominal* context. Of course, in a more realistic setting, this policy may depend on other contexts, for instance temporal contexts.

Listing 1.1. Context definition

```
hold(corp, -, Action, Object, syn_flooding) :-  
    alert(CreateTime, Classification, Target, Source),  
    reference(Classification, 'CVE-1999-0116'),  
    service(Target, Service),  
    name(Service, Action),  
    node(Target, Node),  
    name(Node, Object).
```

Notice that our approach provides *fine-grained* reaction. For instance, let us consider a network where a given host *ws* with IP address 111.222.1.1 is assigned to the role *web_server*. Let us assume that a Syn-flooding attack is detected against this host on port 80. In this case, we shall derive the following fact:

- *hold(org, -, http, ws, syn_flooding)*: means that host *ws* is now in the intrusion 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(prohibition, org, internet, tcp_service, web_server, syn_flooding)*: means that, in the intrusion 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*.

Thus, in our approach, we can associate intrusion 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).

3.2 Contexts and IDMEF Alerts

IDMEF (Intrusion Detection Message Exchange Format [16]) 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 framework⁴. 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.

⁴ <http://www.prelude-ids.org/>

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, to define context parameters, 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, to define context parameters, 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, to define context parameters, and to activate contexts.

We use the two first attributes to compute a context lifetime, as shown in section 3.3. We use mapping functions to translate the last three attributes into contextual information, as shown in section 3.4.

Our approach also requires some additions to the Or-BAC model of a system. They are limited to the *activities* graph, namely we shall add *malicious* activities. By contrast, we consider that the *views* graph and the *use* facts are usable without modifications, because the objects available for the normal activity of the information system are also the objects that are susceptible to attacks. The same stands true for the *roles* graph and the *empower* facts at the moment, because we believe that it is extremely difficult to model an attacker. At the present stage, we could define attacker roles (e.g. script kiddie or skilled attacker), but there would be no *empower* fact associated with these roles. As such, we would not be able to use them in *hold* predicates (they require concrete information), and therefore would not use them to activate contexts.

3.3 Context Lifetime

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 as 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 1.

When an alert occurs, the context is activated with the expiration date set according to the table. If the context is already active, the duration of the context activity is replaced by the current value. When the duration expires, the context is retracted from the contexts database. Both asserts and retracts trigger a re-evaluation of the security policy.

The values of table 1 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.

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	

Table 1. Duration of context activity according to IDMEF impact severity and type, in minutes

3.4 Mapping from Alerts to Contexts

Mapping alert information to contextual information 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 need 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 modelled 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.

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.

3.5 Influence of Mapping on the Response Strategy

The mapping from alerts to contexts 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.

4 The Threat Response System

4.1 System Architecture

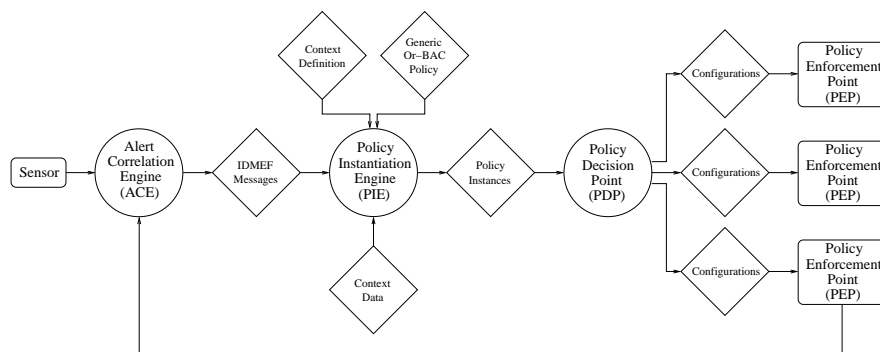


Fig. 1. Threat response system architecture

The architecture of the threat response system is presented in figure 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, 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 2.

Module	Input	Output	Configuration	Function
ACE	IDMEF messages	IDMEF messages	External security reference databases	Verify and update impact information in IDMEF messages for context assessment. Verify target information for views and roles assessment.
PIE	IDMEF messages	Or-BAC rules	Or-BAC policy and context definitions	Extract a new security policy from the active contexts.
PDP	Or-BAC 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.

Table 2. Function of the software modules

4.2 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 semantic improvement. Alert correlation aims at realizing this task, thus permitting false positives reduction and producing meta-alerts offering a better semantic 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 [17]. We define an ACE as an entity receiving as input every possible event produced by sensors and giving as output high-level IDMEF-conformant 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 [18, 19, 17, 20] 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.

4.3 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 a triple function: 1. activate contexts which 2. trigger generic policy rules, and 3. produce a coherent set of rules to deploy while ensuring conflict resolution. The PIE also manages the context lifetime according to the parameters described in Section 3.3.

Context formalization We explained that generic Or-BAC rules are instantiated by the PIE considering active contexts. Thus, there is a need for context formalization, in order to express fine-grained Or-BAC rules allowing fine-grained responses to threats. On the same purpose, all other Or-BAC entities should fulfil this requirement. To achieve that, we propose to manage hierarchies of organizations, roles, activities, views and contexts thanks to graphs definition (see for example fig. 2). Note that graphs definition should be as detailed as possible, in order to express accurate and efficient responses. Indeed, it is essential to characterize contexts as precisely as possible since contexts are used to represent threats. On the same purpose, it is also of great interest to have detailed information concerning organization, role, activity and view.

Context priority Since many contexts are to be activated at the same time, there is a need for a context order property. In particular, it is possible that two rules are activated for similar Or-BAC entities, but corresponding to opposite actions (a permission and a prohibition). For example, a *threat* context must have a higher priority than the *nominal* context. Thus, we say that *threat* contexts override the *nominal* context. Note that a *minimal* context overrides all other contexts, since it has the highest priority.

Context composition Once contexts are activated, they become part of contextual data, that is they enter in the process of context activation. In fact, some contexts may only be active provided other contexts are active. For example, a context may be defined only under specific temporal conditions, characterized by a temporal context (ex: *working_hours*). Moreover, it helps fulfilling the *minimal* requirements.

In our case study (Section 5), let us consider the fact that it should always exist a way to read mail. 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 by the first rule of listing 1.2. 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.

Note that this availability problem is solved here with a static rule, indicating an explicit permission to exchange via outlook access. However, the concept is extensible to more complex strategies, for example taking into account temporal contexts to define the priorities over confidentiality, integrity and availability. 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. Also, while exchange via outlook access offers the most extensive pack of features (mail, but also calendar and address book), it is expensive network-wise, and we could prefer to preserve webmail access in the case of denial of service attacks.

4.4 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 generic rule, it first decides whether or not it has to take it into account considering its PEPs (Policy Enforcement Points) abilities. Then, when a PDP accepts a rule, it splits it into sets of sub-rules expressing actions to produce on PEPs to enforce the new policy. Lastly, these sub-rules are translated according to a local strategy. The same rule does not necessarily results in the same translation within different domains. For example, a prohibition may result in the stopping of a service in a domain and be characterized by a port blocking in another one, or maybe both.

Deployment Deployment is the process of adapting a generic policy rule to a concrete enforcement strategy. For example, a prohibition for a specific service may be split between an action on a firewall (block a port), an action on the service (stop the service), or both. However, above such typical primitive scenarios, it is possible to imagine more advanced ones, 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.

Translation The deployment process returns Or-BAC rules which should be directly translatable by the PDP. The translation process is divided in two sub-processes: the first considers the PEP type (ex: a firewall) and the second takes into account the PEP implementation (ex: a “Netfilter” firewall) [15].

The current PDP implementation generates firewall rules for reconfiguring the iptables firewall acting as PEP, sitting between the email servers and the clients.

4.5 Policy Enforcement Point (PEP)

PEPs receive new policies (or policy elements), which have been translated by the PDP [15]. 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.

5 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 3 exchange servers, which can be accessed by four different mechanisms, the native outlook to exchange, pop, imap, and webmail via Outlook Web Access. In normal operation, all these four 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 the first-order logic required by Or-BAC.

5.1 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 figure 2. 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 and a web server, protected by a specific firewall, as shown in figure 2(a). Users have four channels for accessing email, the classic pop and imap protocols with their application of choice, outlook using the proprietary exchange protocol, and a webmail application. All four 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 *views* of this case study are limited to the email activity, declined along the four possible protocols. The *roles* graph of this case study is a little bit more complex since we need to differentiate domain authentication used to access the internal web and email-access authentication. The *activities* graph is limited to reading mail activity, again specialized around the four different available mechanisms. Note that all possible activity instances are not represented in the figure. Finally, the *contexts* graph defines the different contexts that can be activated and are used in defining policy rules. The three principal contexts are related to nominal activity, to attacks and to time-dependant policy rules. The nominal context is always active and defines the security policy that offers the most convenience to users. The attack context defines contexts that are activated by the alert correlation engine when alert information is received from intrusion detection systems and when this alert information is relative to one of the specific protocols used for email access. Finally, the time context defines policy rules that are used to override rules activated by the attack context and that enable minimal access to email information, to ensure that this information is always available through the less risky protocol.

In the case study, the ACE, PIE and PDP are implemented as prolog predicates in SWI-Prolog, and the PEP as XSLT transformations. For the purpose of the case study, 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. The components of the model

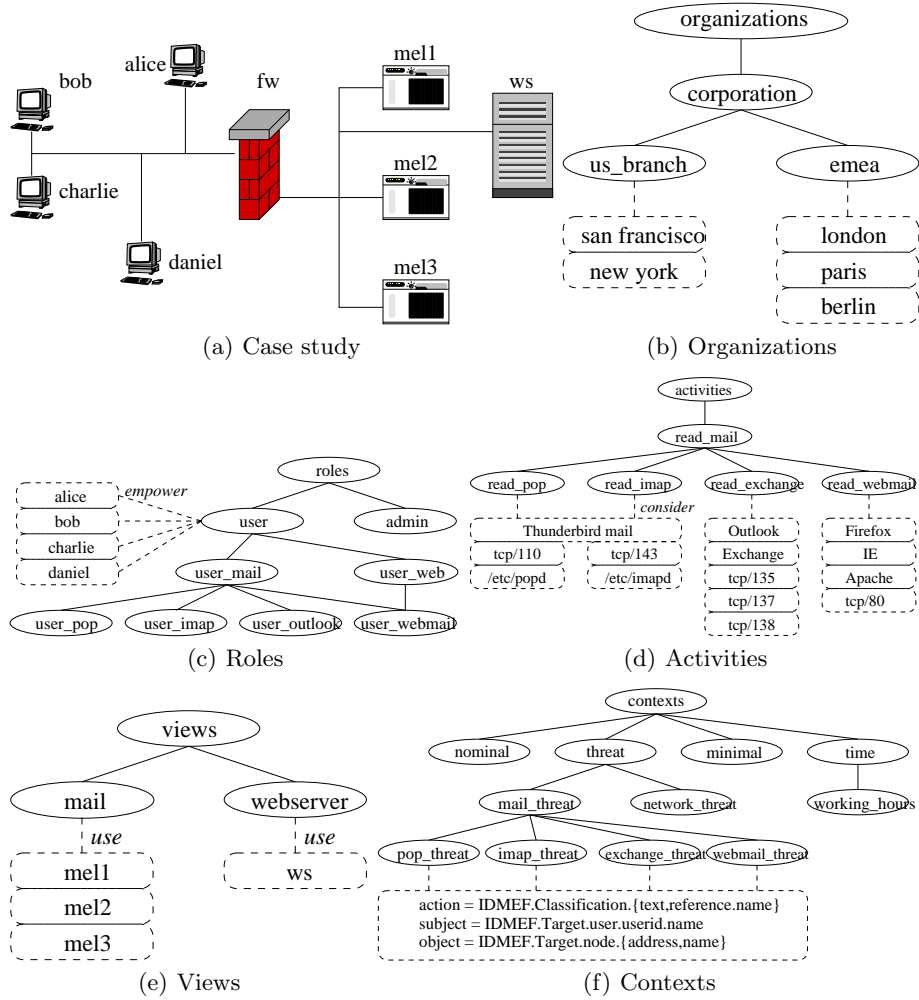


Fig. 2. Description of the policy components

(graphs of abstractions and instances) are modelled in a straightforward way using prolog facts, among them *empower*, *consider* and *use*.

5.2 Definition of the Security Policy

Following the definitions of section 2, we define the security policy as shown in listing 1.2. This security policy specifies, in a few statements, that users must have access to outlook during working hours even in the case of attacks (minimal requirement), that any attack against one of the email access mechanisms invalidates the access mechanism being attacked, and that by default, users have

Listing 1.2. Email access control policy

```
security_rule(permission, corp, user_outlook, read_exchange, mail, minimal & working_hours).
security_rule(prohibition, corp, user_pop, read_pop, mail, pop_threat).
security_rule(prohibition, corp, user_imap, read_imap, mail, imap_threat).
security_rule(prohibition, corp, user_webmail, read_webmail, webserver, webmail_threat).
security_rule(prohibition, corp, user_outlook, read_exchange, mail, exchange_threat).
security_rule(permission, corp, user, read_mail, mail, nominal).
```

Listing 1.3. Hold predicates

```
-- Specific predicate for transforming alerts into contexts
hold_threat(corp, Subject, Action, Object, Context) :-
    alert(CreateTime, Classification, Target, Source),
    map_context(Classification, Target, Source, Context),
    map_subject(Classification, Target, Source, Subject),
    map_action(Classification, Target, Source, Action),
    map_object(Classification, Target, Source, Object).

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

hold(corp, Subject, Action, Object, Intrusive_context) :-
    hold_threat(corp, Subject, Action, Object, Intrusive_context).

hold(corp, Subject, Action, Object, minimal) :-
    hold(corp, Subject, Action, Object, pop_threat),
    hold(corp, Subject, Action, Object, imap_threat),
    hold(corp, Subject, Action, Object, webmail_threat),
    hold(corp, Subject, Action, Object, exchange_threat).

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

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 modelled the environment and the security policy, we need to express the *hold* predicates as shown in listing 1.3. To facilitate the expression of contexts, we have synthesized all threat-related activity into the *hold_threat* predicate; doing so stabilizes the *hold* predicate interface. The *working_hours* context is modelled in a straightforward way, as is the *nominal* context. We define the *minimal* context as a simultaneous concatenation of all attacks against one of the email access mechanisms. Hence, during working hours, when all four access mechanisms are under attack and being suppressed, the context *minimal&working_hours* is active and the policy specifies in this case that the exchange access is re-opened ensuring continued availability of email information. Context priorities are defined to ensure conflict resolution between similar rules activated for different contexts.

The partial order relationship defined by the predicates and the inheritance mechanism is sufficient to ensure the proper evaluation of the security policy, as shown in [21].

5.3 The Mapping Predicates

The core of the *hold_threat* predicate is represented by the four mapping functions, *map_context*, *map_subject*, *map_action* and *map_object*. An example of such mappings is shown as instance of the relevant contexts in figure 2(f). Note that this mapping is not quite naive. It provides multiple choices for mapping the IDMEF classification to an action, either the text that names the alert or any reference that is associated with the IDMEF message. If the mapping fails, the context will not be activated.

It also includes important threat response choices. In this case study, we have chosen to protect user accounts rather than eliminate attackers. 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. We do not attempt to verify that the user is included in our model yet. 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.

6 Issues with the Approach

While this approach is still under development, the current work has brought up a number of interesting issues.

6.1 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.

6.2 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 1 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.

7 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, ease of use 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 to 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 contexts in threat response vary according to alerts collected by various sensors. These alerts received as IDMEF messages are mapped to policy subjects, objects and actions and are used to activate specific contexts. The mapping from IDMEF messages to policy objects 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 modelling 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. Brackney, R.: Cyber-intrusion response. In: Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, West Lafayette, IN (1998) 413
2. Toth, T., Kruegel, C.: Evaluating the impact of automated intrusion response mechanisms. In: Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC), Las Vegas, NV, IEEE Computer Society Press (2002)
3. Petkac, M., Badger, L.: Security agility in response to intrusion detection. In: 16th Annual Computer Security Applications Conference (ACSAC'00), New Orleans, LO (2000) 11

4. rfc3360: Inappropriate tcp resets considered harmful. RFC 3360 (2002) <http://www.ietf.org/rfc/rfc3360.txt>.
5. 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 (2004)
6. Mounji, A., Charlier, B.L.: Continuous assessment of a unix configuration integrating intrusion detection and configuration analysis (1997)
7. Ragsdale, D., Carver, C., Humphries, J., Pooch, U.: 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 (2000) 2344–2349
8. Carver, C., Hill, J., Pooch, U.: Limiting uncertainty in intrusion response. In: Proceedings of the 2001 IEEE workshop on Information Assurance and Security, United States Military Academy, West Point, NY (2001)
9. Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in Operating Systems. *Communication of the ACM* **19**(8) (1976) 461–471
10. Sandhu, R., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. *IEEE Computer* **29**(2) (1996) 38–47
11. 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, ACM Press (2000) 87–96
12. Miège, A.: Definition of a formal framework for specifying security policies. The Or-BAC model and extensions. PhD thesis, ENST (2005)
13. Cuppens, F., Cuppens-Bouahia, N., Miège, A.: Inheritance hierarchies in the Or-BAC Model and application in a network environment. In: Second Foundations of Computer Security Workshop (FCS'04), Turku, Finland (2004)
14. Ullman, J.D.: Principles of Database and Knowledge Base Systems. Computer Science Press (1989)
15. Cuppens, F., Cuppens-Bouahia, N., Sans, T., Miège, A.: A Formal Approach to Specify and Deploy a Network Security Policy. In: Formal Aspects of Security and Trust (FAST), Toulouse, France (2004)
16. Debar, H., Curry, D., Feinstein, B.: The intrusion detection message exchange format. Internet Draft (2005) Work in progress, expires July 31st, 2005.
17. Cuppens, F., Miège, A.: Alert Correlation in a Cooperative Intrusion Detection Framework. In: Proceedings of the IEEE Symposium on Security and Privacy. (2002)
18. Dain, O., Cunningham, R.: Fusing a Heterogeneous Alert Stream into Scenarios. In: Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications. (2001) 1–13
19. Morin, B., Mé, L., Debar, H., Ducassé, M.: M2D2 : A Formal Data Model for IDS Alert Correlation. In: Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID). (2002)
20. 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)
21. Cuppens, F., Miège, A.: Administration Model for Or-BAC. In: International Federated Conferences (OTM'03), Workshop on Metadata for Security, Catania, Sicily, Italy (2003)