

Response: bridging the link between intrusion detection alerts and security policies

Hervé Debar¹, Yohann Thomas^{1,2}

{herve.debar,yohann.thomas}@orange-ftgroup.com

¹France Télécom R&D - 42 rue des Coutures - 14066 Caen, France

Frédéric Cuppens², Nora Cuppens-Boulahia²

{frederic.cuppens,nora.cuppens}@enst-bretagne.fr

²GET/ENST Bretagne - 2 rue de la Châtaigneraie - 35512 Cesson-Sévigné, France

Keywords: Intrusion Detection, Threat Response, Reaction, Access Control

Abstract

With the deployment of intrusion detection systems has come the question of alert usage. The current trend of intrusion prevention systems provides mechanisms for isolated response, suffering from two important drawbacks. First, the response is applied on a single point of the information system. Second, its application is repeated every time an alert condition is raised. Both drawbacks result in a suboptimal response system, where security is improved at these particular network or host access control points, but where service dependancies are not taken into account. In this paper, we 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

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 also to be strongly consid-

ered. In particular, being able to serve large numbers of users concurrently or to maintain acceptable response times, while lightening the hardware budget, is a major issue, and sometimes results in conflicting choices with respect to security. Moreover, ease of use and automation are frequent requirements to provide better service to users.

Nowadays, this compromise between 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, and similarly for 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 better balance between the different requirements, as time goes by.

Consequently, the compromise between the considered system adjustment variables needs to change, and in particular to respond to threats. This paper describes a mechanism for threat management at the security policy level. The security policy is dynamically updated with respect to current threats. This update is performed in a global manner, ensuring that the whole security policy remains coherent and that threats are handled by order of importance, even when threats have conflicting impacts and may require conflicting countermeasures. Our policy update mechanism also enables countermeasures with safeguards, ensuring that the security officer has control over the most adverse operating conditions, and prevent self-inflicted denial of service. We also provide an architecture to deploy such policies, mostly by reusing already existing security and system management components and protocols.

2 Problem statement

2.1 Domain terminology

Previous chapters of this book have described alerts generated by intrusion-detection systems. In this chapter, we need to broaden the terminology used to introduce additional concepts, as shown by the question marks in figure 1.

We reuse definitions introduced by the MAFTIA project for dependability and security [1]. The central concept is the one of *fault*, that is defined as a breach of the security policy. A synonym in the literature is *threat*, used for example by J. Anderson [2], E. Jonsson [15] or in RFC 2828 [23]. We are particularly interested in the notion of *malicious fault*, when an *attacker* can exercise the fault to carry out an *attack*. When the attacker succeeds, such an attack becomes an *intrusion*; the advantage gained may be reused by the attacker to carry out further attacks. In this paper, we will use the terms *fault* and *threat* interchangeably to describe this same concept.

To prevent these faults from occurring, the MAFTIA project defines four categories of actions ([1], p 9) that can be undertaken in parallel. *Fault prevention* aims at preventing the existence of faults. *Fault tolerance* aims at continuing operations in the presence of faults. *Fault removal* aims at reducing the number

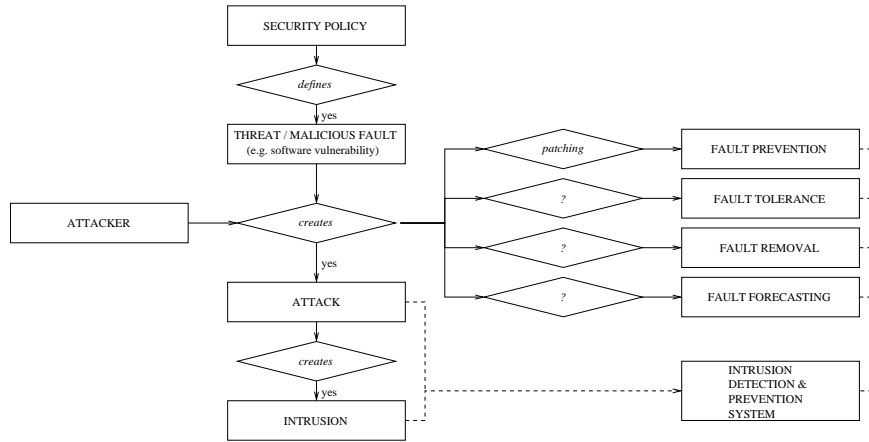


Figure 1: Domain terminology

or severity of faults. Finally, *fault forecasting* aims at estimating the number and severity of faults. Current intrusion detection and prevention systems are pertinent in the three later cases, aiming at alerting the security officer when attacks or intrusions manifest the existence of faults.

Fault prevention is outside the scope of our study, as we consider that patch management is an appropriate answer to the problem. If needed, we could incorporate patch deployment into the framework as a definitive and irreversible countermeasure, but that would be contrary to the philosophy of our approach, which promotes dynamicity (we do not want to “unpatch” systems automatically). We assume that some threats will not be removed, as their removal would have an adverse effect on the global system (for example adversely affecting cost), or because the security policy specifies contradicting objectives (e.g. the duality availability/confidentiality). Thus, system operation in the presence of these threats is a requirement. We are specifically interested in providing solutions for fault tolerance and fault removal, and handle fault forecasting as a possible side effect of our system, to be studied in future work. We further reduce the problem by observing that fault tolerance and removal assume the capability to first identify the fault and then act on it. We assume that fault detection will be handled by intrusion detection and security information management systems, and focus on actions to be carried out to the effect of tolerance or removal. We call these actions *response*.

Figure 1 also shows that response could be understood as *threat, attack or intrusion response*. With respect to intrusion detection, the most obvious term to define is attack response, i.e. responding to alerts that indicate an attack, a possible breach of security policy. Focusing on *intrusion response* would indicate that our system would only respond to successful attack. Threat response indicates that our system is specified from all latent faults, even though

they may not materialize as attacks.

2.2 Intrusion Prevention and 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. In particular, correlating alerts with the inventory of the hosts [24] 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 [3].

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 [25]. Host-based intrusion prevention software has the capability to terminate a process that is trespassing or abusing its privileges, as shown by [21], 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 [13] 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. The objective of the paper is to propose a more comprehensive approach to threat response.

2.3 Comprehensive Approach to 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 [8] for example. As shown in [24], 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.

3 Security Policy Formalism

In this section, we provide background on the security policy formalism and describe a use case.

3.1 Choice of a Security Policy Formalism

Most of current security models such as DAC [14] or RBAC [22] 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 [17]; 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 [6].

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 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 [18]. 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.

3.2 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:

- $security_rule(prohibition, corp, pop_user, read_pop, mail_server, pop_threat)$.

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 threat.

All these concepts, organization, role, activity, view and context, may be structured hierarchically. Permissions and prohibitions are both inherited through these hierarchies (see [5] 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 [6]. 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 [18] for further details). Since the Or-BAC model abides to the Datalog restrictions [26], 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. For 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.

3.3 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. This is modeled by the derivation rule shown in 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 [7] for further details in the case of network security policies).

3.4 Presentation of a use case

To illustrate the response mechanism, we present the following use case, access to mail. Users have access to their mail located on remote exchange servers. They can use three different mail clients, outlook, thunderbird and firefox, over four different transport mechanisms, the outlook mail client accessing the exchange server through native microsoft protocols, thunderbird accessing the POP and IMAP extensions of the same exchange servers, and Firefox accessing the OWA ¹ extension of the same exchange servers. In normal operation, all these four modes are active and allow parallel access to the same information, the consistency being preserved by the backend exchange server.

3.4.1 Use case illustration

Figure 2 provides a description of the various entities involved in this use case. This description is layered to ease understanding of these entities, but a request for access will contain information belonging to all three layers. At the transport layer, a network packet contains information about IP addresses and ports in the headers, and commands and data for the programs in the packet payload according to the application-layer protocol specification. This is also true at the service layer, where commands and data are presented to the processes that obey and manipulate them.

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, *empower*, *consider* and *use*.

3.4.2 Horizontal layer segmentation

The information layer at the top models interactions between the humans and the information they wish to access. In our use case, users wish to access their mail messages. The middle layer represent the system intermediaries, typically programs, that make this mail reading possible. In our use case, mail clients such as Microsoft OutlookTM, Mozilla Thunderbird or Mozilla Firefox² interact with mail servers such as Microsoft ExchangeTM. The bottom layer represents the communication channels, enabling communication between machines; in our case, this enables the exchange of TCP/IP packets between user workstations and servers.

3.4.3 Vertical segmentation

In addition to the layer segmentation, figure 2 also introduces a vertical separation. In addition to the classic Subject/Object duality, there are a large number of infrastructure functions that enable communication at one of the layers. Without these infrastructure functions, access is at best impaired and

¹Outlook Web Access

²used as support for webmail access

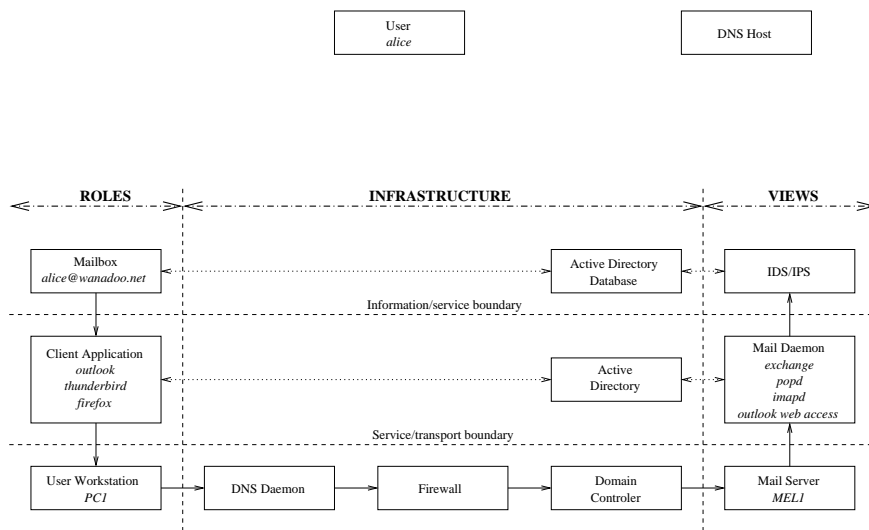


Figure 2: Presentation of the use case

at worst impossible. Hence, they represent an attractive target for attackers and a possibility of countermeasure for the defender, and we wish to extend the classic policy model of subjects and objects by representing these components.

In our use case, user workstations rely on DNS to identify the target machine. User logins and access to information rely on ActiveDirectory to identify and authenticate users, and associate user logins with mail boxes. They also need to traverse firewalls and intrusion detection/prevention systems. Note that it is not necessary to create new concepts to model these infrastructure entities; subjects and objects apply to them as well. Their presence in the model improves the understanding of the security policy and widens opportunities for threat response, since any combination of these elements can be leveraged.

3.4.4 Impact on policy enforcement

The vertical segmentation requires that our model for policy translation incorporate the ability to model reliably the infrastructure components and their interactions. We must incorporate contextual information, such as routing, that is not directly included in alerts (and rarely included in high-level security policies), that enables our system to infer the appropriate policy enforcement points where the policy is applicable and effective.

Such a policy system also implies that there is a capability to collect and correlate state and contextual information (as is currently done in intrusion detection systems). With respect to state, the establishment of connectivity at the transport layer such as the three way TCP handshake will be required before users can present credentials such as user names, passwords and mailboxes.

Higher up, we need to recognize service states (login in, logged in, etc.) and associations between sessions and users.

With respect to information inference, we need to make use of configuration information, as in alert correlation. For example, while firefox will present its identity to OWA, Thunderbird and Exchange may not indicate themselves to the server. Furthermore, this identification information can be spoofed (it is for example very easy to fake the user agent of a web browser to another one). Our system will not be able to enforce the usage of firefox instead of opera or the usage of outlook instead of evolution if it only relies on transport level or view-side policy enforcement points. As long as the protocol exchanges are correct, we will not be able to model this at the abstract level or recognize it at the concrete level; this can only be enforced by a role-side, service-level policy enforcement point.

We are currently working on the representation of such models and are confident that the basic technologies, particularly state and correlation, can be provided by the intrusion detection and security information components available today.

3.5 Modelling of the use case

We now model the use case, define the appropriate abstract entities and describe their relationships with concrete entities. To do this, we will adopt whenever needed a simple tree representation, where properties defined at one node propagate to the nodes below. Abstract entities are represented as ovals and concrete entities are represented as square boxes.

Note that we do not claim that the model is exhaustive; we will limit our description to the needed components.

3.5.1 Roles and views

Since roles and views have symmetric behaviour with respect to the policy, we describe them together in figure 3. Both are segmented according to the three layers of 2; while this is not absolutely necessary and we could attach the various components that are in the layers to the *MailRole* or *MailViews* nodes, it provides additional segmentation that will prove useful for defining and analyzing countermeasures. Accordingly, the various concrete objects are attached to the appropriate abstract nodes.

This model presents a simplified view of the system. We model services by their startup/shutdown script names, mailboxes by the email address, mail clients by their names, and machines by their names. In a complete model, we would need more details on the various processes that intervene in the execution of a particular program, or relationships between machine names and IP addresses. These details could either be stated as additional facts or inserted as knowledge-gathering activities (e.g. using DNS to resolve host names into IP addresses).

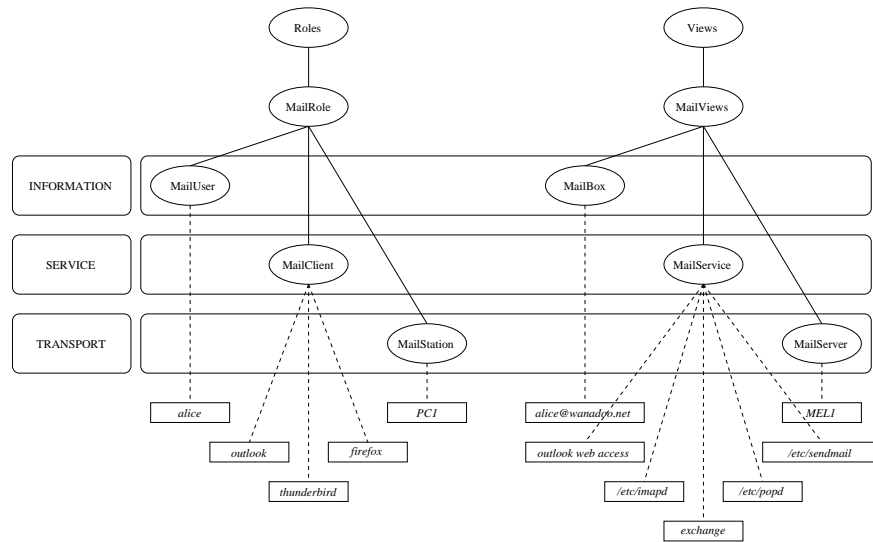


Figure 3: Model of roles and views

We therefore obtain the appropriate role hierarchy, empower and use, as expressed in listing 2 (using the prolog formalism that we reuse in this chapter; the information is expressed as prolog facts).

Listing 2: Definition of roles and views

```

role(org, 'MailRole').
role(org, 'MailUser').
role(org, 'MailClient').
role(org, 'MailStation').
subrole(org, 'MailUser', 'MailRole').
subrole(org, 'MailClient', 'MailRole').
subrole(org, 'MailStation', 'MailRole').

empower(org, 'alice', 'MailUser').
empower(org, 'Thunderbird', 'MailClient').
empower(org, 'Outlook', 'MailClient').
empower(org, 'Firefox', 'MailClient').
empower(org, 'PC-Alice', 'MailStation').

view(org, 'MailView').
view(org, 'Mailbox').
view(org, 'MailService').
view(org, 'MailServer').
subview(org, 'Mailbox', 'MailView').
subview(org, 'MailService', 'MailView').
subview(org, 'Mailserver', 'MailView').

use(org, 'alice@wanadoo.net', 'Mailbox').
use(org, 'OWA', 'MailService').
use(org, '/etc/imapd', 'MailService').
use(org, '/etc/popd', 'MailService').
use(org, '/etc/sendmail', 'MailService').
use(org, 'Exchange', 'MailService').
use(org, 'MEL1', 'MailServer').

```

The role and subrole (resp. view and subview) predicates construct the role (resp. view) hierarchy. Note that this listing is biased; in any realistic deployment, there should be many more concrete entities than abstract entities. The ratio of one to one in the listing is not representative of a realistic setting.

3.5.2 Activities

Activities are described in figure 4. The segmentation across layers appears in this figure in the content of the square box, representing either ports or commands to the service. We have further segmented mail under three activities, connecting to the mailbox, reading mail and sending mail. This segmentation is introduced with respect to the response system, where options such as preventing new sessions but letting existing sessions continue, or letting users read but not send mail, are opening up additional opportunities for the response system to focus and limit the response on the area under attack.

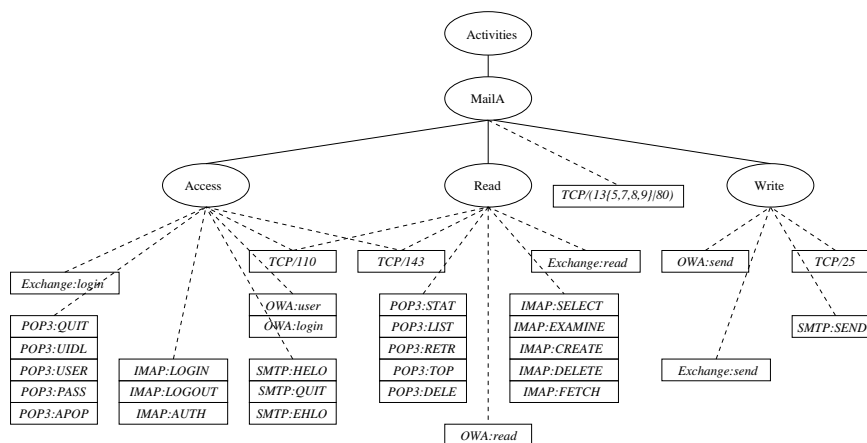


Figure 4: Model of activities

Activities are modeled at the information level by retaining the various keywords used by the protocols to open the connection between the mail client and the mail server, and at the service and transport layer by the protocols and ports involved. Since protocols and ports are both found in IP packets and on machine (as bound ports), it is sufficient to model them with a single concrete object. For convenience, we are using a regular-expression-like notation for the microsoft protocols, since they need to be configured together to enable the activity to succeed. Protocol keywords for POP, IMAP and SMTP are taken from Request for Comments (RFC) 1939, 2060 and 2821 respectively, although we have only introduced here a subset of these keywords. Keywords for OWA are inspired from the interface button names but should ideally associate URLs and AJAX commands. Since the Exchange/Outlook dialog is not a public standard,

we have introduced meta-keywords that correspond to the various mail-related activities of Exchange.

In figure 4, we also have to acknowledge that this granularity does not apply to all possible activities. For example, the direct connection between the Microsoft Outlook mail reader and the Microsoft Exchange mail server limits our capability to separate get and send at the network layer, because all three sub-activities use the same set of ports. This is the same for the separation between login and read at the network layer. The login activity, however, is separable at the service layer, for example by manipulating the active directory server to prevent exchange login requests to succeed, or by filtering out IMAP and POP3 login requests.

Activities are described in listing 3. We have not listed all the consider facts, but the reader can easily construct them from figure 4.

Listing 3: Definition of activities

```

activity(org, 'MailA').
activity(org, 'Access').
activity(org, 'Read').
activity(org, 'Write').
subactivity(org, 'Access', 'MailA').
subactivity(org, 'Read', 'MailA').
subactivity(org, 'Write', 'MailA').

consider(org, 'TCP/(80|13[5,7,8,9])', 'MailA')
consider(org, 'TCP/110', 'Access').
consider(org, 'TCP/143', 'Access').
consider(org, 'TCP/110', 'Read').
consider(org, 'TCP/143', 'Read').
consider(org, 'TCP/25', 'Write').
consider(org, 'Exchange:login', 'Access').
consider(org, 'POP3:APOP', 'Access').
consider(org, 'IMAP:LOGIN', 'Access').
consider(org, 'SMTP:HELO', 'Access').
consider(org, 'OWA:user', 'Access').
consider(org, 'POP3:STAT', 'Read').
consider(org, 'IMAP:SELECT', 'Read').
consider(org, 'SMTP:MAIL', 'Write').
consider(org, 'Exchange:send', 'Write').

```

Note that the normal behaviour of certain protocols puts demands on the model. The rattachement of *TCP/(80|13[5,7,8,9])* to 'MailA' is due to the fact that at the network level, all activities related to the Outlook-Exchange or Firefox-OWA connections cannot be differentiated easily. However, the service level does differentiate access, read and write activities.

3.5.3 Contexts

For the normal operation of our system, we only need one context, the 'My-Mailbox' context, whose objective is to ensure that the user can only access his mailbox. This is necessary because access to the mailbox relies on access to the information system, and there is a need to link the authentication data (login,password) to the mailbox. The 'MyMailbox' context inherits from the 'Normal' context as shown in listing 4. We also define a partial order between

contexts by indicating that the 'MyMailbox' context has priority over the 'Normal' context.

Listing 4: Definition of contexts

```

context(org, 'Normal').
context(org, 'MyMailbox').
subcontext(org, 'MyMailbox', 'Normal').
context_priority_high_low(org, 'MyMailbox', 'Normal').

hold(org, Subject, Action, Object, 'MyMailbox') :-
    empower(org, Subject, 'MailUser'),
    consider(org, Action, 'MailA'),
    use(org, Object, 'Mailbox'),
    arelinkedinactivedirectory(Subject, Object).

hold(org, Subject, Action, Object, 'MyMailbox') :-
    empower(org, Subject, 'MailClient'),
    consider(org, Action, 'MailA'),
    use(org, Object, 'MailService').

hold(org, Subject, Action, Object, 'MyMailbox') :-
    empower(org, Subject, 'MailStation'),
    consider(org, Action, 'MailA'),
    use(org, Object, 'MailServer').

```

The hold predicates are written in listing 4 using primitives to interrogate the Active Directory server. Since information about subjects and objects is separated in three layers, only the higher layer is pertinent for the assertion of the MyMailbox context, as described in the first hold statement. This hold statement verifies that the request is presented with a user name and a mailbox, and queries the active directory server for the connection between the two. However, if we only allow this, we will not succeed in deploying the security policy on policy enforcement points located below the information layer : the hold statement will never be satisfied, because this association only occurs at the information layer. Since servers and processes are shared by all users, the second and third hold statements satisfy the policy for lower-level policy enforcement points.

3.5.4 Security rules

Within the above model, the security policy is expressed by the two simple rules of listing 5.

Listing 5: Definition of contexts

```

security_rule(permission, org, MailRole, MailA, MailView, MyMailbox).
security_rule(prohibition, org, MailRole, MailA, MailView, Normal).

```

According to this policy, users are prevented from reading mailboxes in general. However, a potential conflict occurs when the MyMailbox context holds, and the higher priority of the MyMailbox context with respect to the Normal context enables access.

4 Applying Or-BAC for threat response

Or-BAC contexts provide a natural way to include threat response into Or-BAC policy rules, to create specific rules that apply during an attack. 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* and *pop_attack* are two examples of threat contexts.

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 Examples of threat contexts

We propose here two examples of threats and explain how response is managed depending on the active *hold* predicates and the security rules describing the policy to apply in such cases. Listings follow the *prolog* syntax (SWI-Prolog is our implementation language), although with a simplified representation of the alerts, as the IDMEF XML representation results in deeply imbricated lists that we have simplified to ease comprehension – alerts are abstracted as position dependant 4-tuples (timestamp, attack source, attack destination, attack identification), each of these 4-tuples being represented by a list of (IDMEF token, value) pairs.

4.1.1 Syn-flooding attack

Let us imagine a Syn-flooding attack towards a webserver. We use IDMEF messages (as explained in Section 4.4.1) 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 6.

Listing 6: syn_flooding context definition

```
% Simplification of alert information extraction
content(K, [[K,E] | _], E).
content(K, [[_, -] | L], E) :-
    content(K, L, E).

contentlist([], _, []) :- !.
contentlist([K|LK], X, LER) :-
    findall(Es, content(K, X, Es), E),
    contentlist(LK, X, LE),
    flatten([E, LE], LER1),
    sort(LER1, LER).
```

```

% synflood context description
hold(corp, -, Action, Object, 'syn_flooding') :-
    alert(CreateTime, Source, Target, Classification),
    content('reference', Classification, 'CVE-1999-0116'),
    contentlist(['service'], Target, Action),
    contentlist(['hostname'], Target, Object).

% Arrival of an alert
alert('2001/01/01_01:01:01',
      [['hostaddress', '1.2.3.4']],
      [['service', 'http'], ['hostname', 'ws']],
      [['reference', 'CVE-1999-0116']]).

% The following now holds
?- hold(Org, Subject, Action, Object, Context).

Org = corp,
Action = [http],
Object = [ws],
Context = syn_flooding ;

?-

```

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*.

4.1.2 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 definition of the hold facts is more complex here, due to the need to maintain the coherence between the information, service and transport layers. The corresponding translation in Prolog can be found in Listing 7.

Listing 7: pop_attack context definition

```
% Simplification of alert information extraction
content(K,[[K,E]|_],E).
content(K,[[_,-]|L],E) :-
    content(K,L,E).

contentlist([],_ ,[]) :- !.
contentlist([K|LK],X,LER) :-
    findall(Es,content(K,X,Es),E),
    contentlist(LK,X,LE),
    flatten([E,LE],LER1),
    sort(LER1,LER).

% pop attack context description
hold(corp, Subject, Action, Object, 'pop_attack') :-
    alert(CreateTime, Source, Target, Classification),
    content('reference', Classification, 'CVE-2005-1133'),
    contentlist(['username'], Source, Subject),
    contentlist(['mailaddr', 'file'], Target, Object).

hold(corp, Subject, Action, Object, 'pop_attack') :-
    alert(CreateTime, Source, Target, Classification),
    content('reference', Classification, 'CVE-2005-1133'),
    contentlist(['hostaddr', 'hostname'], Source, Subject),
    contentlist(['service'], Target, Action),
    contentlist(['hostaddr', 'hostname', 'process'], Target, Object).

% Arrival of an alert
alert('2002/02/02_02:02:02',
      [['hostaddr', '1.2.3.4'], ['hostname', 'charlie's'],
       ['username', 'charlie']],
      [['service', 'tcp/110'], ['process', '/etc/initd/pop'],
       ['file', '/var/spool/mail/charlie'],
       ['hostname', 'ms'], ['mailaddr', 'charlie@net.net']],
      [['reference', 'CVE-2005-1133']]).

% The following now holds
?- hold(Org, Subject, Action, Object, Context).

Org = corp,
```

```

Subject = [ charlie ],
Object = [ '/var/spool/mail/charlie', 'charlie@net.net' ],
Context = pop_attack ;

Org = corp,
Subject = [ '1.2.3.4', charlie ],
Action = [ 'tcp/110' ],
Object = [ '/etc/initd/pop', ms ],
Context = pop_attack ;

?—

```

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

- *hold(org, charlie, ['charlie@net.net', '/var/spool/mail/charlie'], pop_attack)*: means that mailbox *charlie@net.net* (and its sibling representation *'/var/spool/mail/charlie'* mail spool file) is now in the threat context *pop_attack*, the attacker being user *charlie*.
- *hold(org, ['1.2.3.4', charlie], ['tcp/110'], ['/etc/initd/pop', ms], pop_attack)* means processes */etc/initd/pop* and server *ms* are now in the threat context *pop_attack*, the attack coming from workstation *charlie* with IP address 1.2.3.4

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 6, it is possible to replace the constant *ws* by a variable, and similarly for constants *charlie* and *ms* in listing 7.

4.2 Atomic contexts

We recon that this approach is likely to greatly increase the number of contexts. To facilitate context management, 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 [9]. 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 rules associated with contexts higher in the hierarchy will override the ones 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 Section 5.3).

If c is a *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.

4.3 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. We therefore define a context algebra to provide a way to combine atomic contexts through a boolean algebra. The algebra provides the following basic functions to manipulate composed contexts:

Negation : $n(c) \leftrightarrow$ context c is **not** active

Conjunction : $\&(c1, c2) \leftrightarrow$ context $c1$ **and** context $c2$ are active

Disjunction : $v(c1, c2) \leftrightarrow$ context $c1$ is active **or** context $c2$ is active

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. Contexts entering in the composition of *composed* contexts are simply named *composing* contexts.

Managing security rules with composed contexts requires the ability to associate a property to the composed context, in relation with the priorities of the composing contexts. We now analyze the possible combinations, giving examples for a better understanding of composed context priorities.

Definition 1 *Since it is possible that there is no attack or intrusion in a negative context, the negation of a context, whatever its category, is an operational 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. $n(\text{working_hours})$, like working_hours , is an operational context; $n(\text{pop_attack})$, negation of pop_attack , is an operational context. Thus, $n(\text{working_hours})$ and $n(\text{pop_attack})$ 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. $\&(\text{working_hours}, \text{in_dmz})$ is the conjunction of a temporal (thus, operational) and a spatial (thus, operational) context. Consequently, $\&(\text{working_hours}, \text{in_dmz})$ 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. $\&(\text{pop_attack}, \text{syn_flooding})$ is the conjunction of two threat contexts. Consequently, $\&(\text{pop_attack}, \text{syn_flooding})$ 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 a 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.

4.4 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_threat*), but also which subject, action and object it deals with, and within which organization.

4.4.1 Information about threat

IDMEF (Intrusion Detection Message Exchange Format [12]) 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 framework³. 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 Section 4.4.2.

4.4.2 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:

³<http://www.prelude-ids.org/>

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.

		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

Table 1: Mapping IDMEF classes on Or-BAC parameters

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. While table 1 does not take into account all IDMEF attributes, we are using the most important ones with respect to the description of alert conditions. We are investigating the alerts produced by different systems to ensure that we are not leaving out important parameters, particularly with respect to information that is stored in the additional data blob.

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 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.

4.5 Context deactivation

Deactivating threat contexts is used to revoke countermeasures once threats are no longer present. We currently manage static context lifetimes, which are computed thanks to IDMEF alerts assessment attributes. Indeed, IDMEF alerts provide an *IDMEF.Assessment.Impact* attribute 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. This is a basic example that relies purely on risk analysis done by the alert providers, but better analysis can be adapted using finer threat analysis, as shown in section 7.1.

Impact severity Impact type	info	low	medium	high	Comment
admin	1	2	4	8	This is the most severe case. We are not currently handling DoS attacks.
dos	0	0	0	0	
file	0	1	2	3	We are not currently handling scans, as they do not result in compromise.
recon	0	0	0	0	
user	0	1	2	4	
other	0	0	1	2	

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

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.

4.6 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. This 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 figure 5. 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 3.

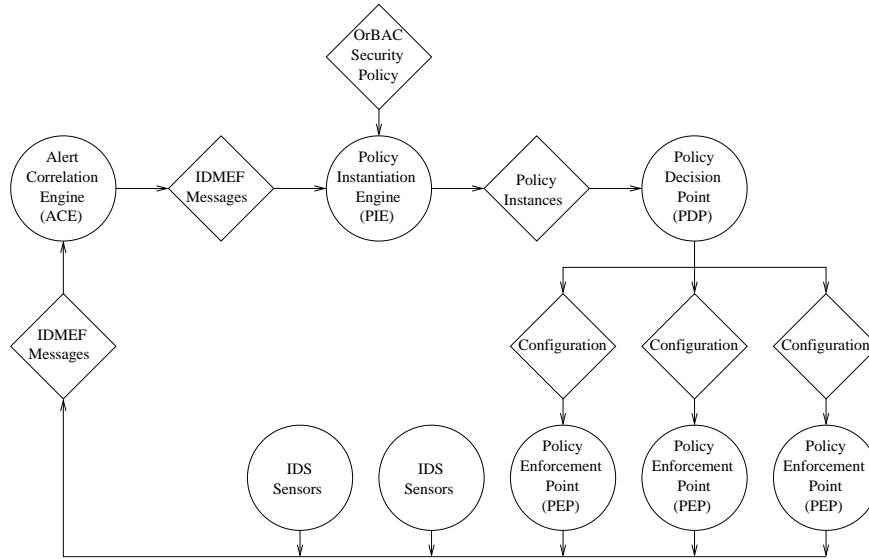


Figure 5: Threat response system architecture

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.

Table 3: Function of the software modules

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 [4]. 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 the scope of this paper, since we consider the existence of valuable works on the subject [4, 11, 19, 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.

5.3 Policy Instantiation Engine (PIE)

The security policy description corresponds to 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 Section 4.4. For operational contexts, such as temporal ones, one can refer to [9] for further information. Note that the PIE also deals with context deactivation, according to Section 4.5. Generic policy rules triggering is explained in Section 3, and examples are given in Section 4.1.

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 managed at the abstract level, by 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 [10].

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 be performed 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) [7]. 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 [7]. 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 possesses specific functionalities of policy enforcement. This characteristic can provide information allowing validation of new policies effective application.

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 6 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 6 also presents in details the components of the Threat Characterization Engine. Threat characterization does not actually consist in a trivial and static mapping, since (1) 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, to ensure that all subjects, actions and objects known to be participating in the attack process are included in the response. The obtained

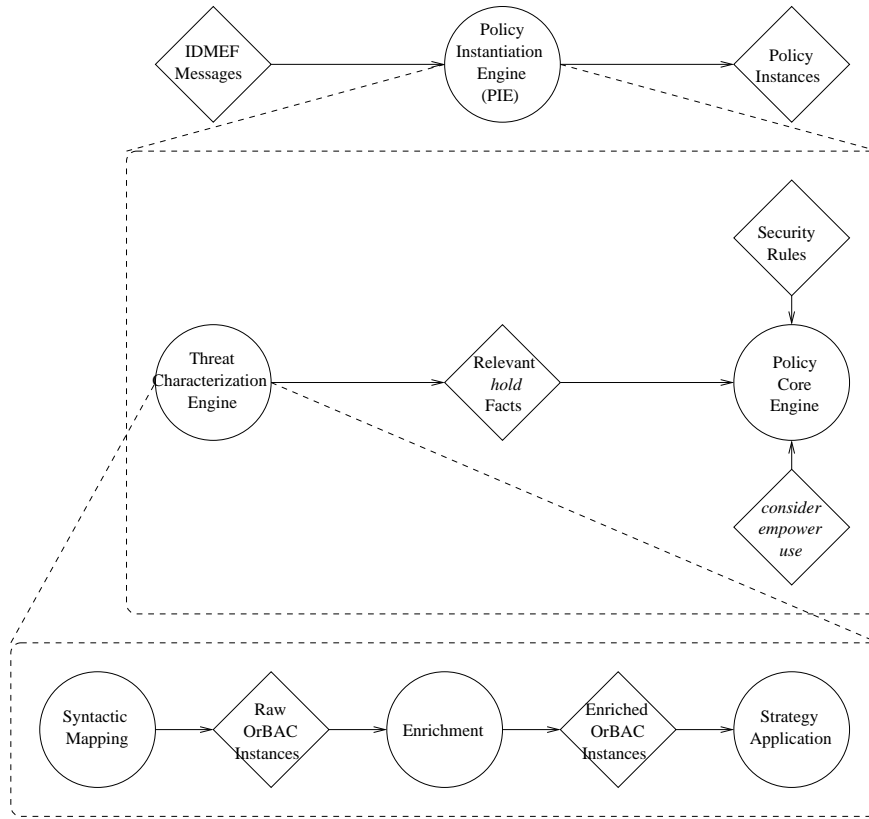


Figure 6: Information workflow, from alerts to new policy instances

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 related to 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 Section 5.4).

1. Subjects, actions and objects are in fact data structures sometimes re-

grouping 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. Although it does not appear on the figure, this process needs IDMEF information related to context triggering (for instance, *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.* extend it 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. Concrete examples of these processes are given in the case study of section 7.

7 Case Study: e-mail Server

We now come back to the case study. As shown in section 3.4, users are able to access their mailboxes through several communication protocols. However, each protocol is vulnerable to attacks, either because of software vulnerabilities or inherent design. Our response system aims at protecting them by disabling usage of the particular component under attack. IDS sensors and other logging systems will detect malicious attempts. When a malicious attempt is detected, we

will react by blocking access to the mechanisms (servers, services or mailboxes) under attack.

We use SWI-Prolog to implement first-order logic based reasoning required by Or-BAC.

7.1 Threats related to the use case

As shown in section 4, and particularly 4.1, the first requirement in our threat response system is to understand the threats that are relevant to the environment of the use case, and derive the appropriate contexts. This means that the security officer needs to carry out a risk analysis on this environment, as already mentioned in section 2.3

We have carried out a software vulnerability search using the National Vulnerability Database⁴ (NVD) using the keywords “exchange” and “pop3”. A partial result of this search is illustrated in table 4. In the table, column 1 gives the CVE reference of the vulnerability. Column 2 evaluates the relevance of this vulnerability to the use case ; the vulnerability is highly relevant if the vulnerable software is clearly present in the use case, low if the vulnerability is clearly absent from the use case, and medium if we could not determine it clearly. Column 3 contains the Common Vulnerability Scoring System⁵ (CVSS) score for the vulnerability, computed from the CVSS vector. The CVSS vector itself is partially explicated in the impact column, where the (*C*) indicates complete impact and the (*P*) indicates partial impact of the vulnerability on either availability, confidentiality or integrity. Finally, columns 5, 6 and 7 analyze the model abstract or concrete entities that are impacted by the vulnerability.

The table contains only cursory information about each vulnerability. The reader is referred to the NVD for a detailed description of the vulnerabilities associated with each CVE reference, and in particular a textual description and the complete CVSS vector. Table 4 only shows partial results from our search, and the search term themselves are not exhaustive ; a thorough examination should include all installed software (we only included 2). It should take into account the detection capability, i.e. the fact that an attempt to use the vulnerability will result in an alert provided by one of the sensors ; if an attack using one of these vulnerabilities cannot be detected, then no response can be included in the threat response system. Also, in the specific case of software vulnerabilities, patching applications will change the relevance of the vulnerability for the threat response system, and thus may change the associated response. We do consider though that there will be some time between vulnerability discovery and patching where such response will be important, and that non-software vulnerabilities (e.g. password guessing) are relevant to the threat response system. The proposed analysis merely illustrates the technology, but must of course be updated when new vulnerabilities are discovered.

We derive the hold facts of the case study from table 4. At least one hold fact is created for each CVE reference. The activity, view and role columns are

⁴nvd.nist.gov

⁵<http://www.first.org/cvss/cvss-guide.html>

CVE Reference	Relevance	CVSS Severity	Impact	Activity (Action)	View (Object)	Role (Subject)
Passwd guessing CVE-2006-7040	High Low	- 3,3	User access DoS	Access TOP	<i>any</i> POP3	<i>any</i> <i>any</i>
CVE-2006-6940	Low	10	Admin access, Confidentiality (C), Integrity (C), Availability (C), DoS	Write, Read, Transfer	OWA, POP3	<i>any</i>
CVE-2006-1193	High	1,9	Unauthorized modification	Read	OWA	Mail user
CVE-2006-0027	High	7	User access, Confidentiality (P), Integrity (P), Availability (P), DoS	Write, Transfer	Exchange	<i>any</i>
CVE-2006-0002	High	7	User access, Confidentiality (P), Integrity (P), Availability (P), DoS	Write, Transfer Read	Exchange <i>any</i>	<i>any</i> Outlook
CVE-2005-1987	High	7	User access, Confidentiality (P), Integrity (P), Availability (P), DoS	Write, Transfer	Exchange	<i>any</i>
CVE-2005-1133	Low	3,3	Allows unauthorized disclosure of information	USER	POP3	<i>any</i>
CVE-2005-0738	High	3,3	DoS	Read	Exchange	Mail user
CVE-2005-0563	High	3,3	Allows unauthorized modification	Read	OWA	<i>any</i>
CVE-2005-0560	High	7	Unauthorized access, Confidentiality (P), Integrity (P), Availability (P), DoS	Write Transfer	Exchange	<i>any</i>
CVE-2005-0420	High	7	Unauthorized access, Confidentiality (P), Integrity (P), Availability (P), DoS	Access	OWA	<i>any</i>
CVE-2005-0044	High	7	User access, Confidentiality (P), Integrity (P), Availability (P), DoS	Write, Transfer Read	Exchange <i>any</i>	<i>any</i> <i>any</i>
CVE-2004-0840	High	10	Admin access, Confidentiality (C), Integrity (C), Availability (C), DoS	Write, Transfer	Exchange	<i>any</i>
CVE-2004-0203	High	10	Unauthorized access, Confidentiality (P), Integrity (P), Availability (P), DoS	Read	OWA	<i>any</i>
CVE-2003-0904	High	5,6	Unauthorized access, Confidentiality (P), Integrity (P), Availability (P), DoS	Access	OWA	<i>any</i>
CVE-2003-0714	High	8	Unauthorized access, Confidentiality (P), Integrity (P), Availability (P), DoS	Write Transfer	Exchange	<i>any</i>
CVE-2003-0712	High	7	User access, Confidentiality (P), Integrity (P), Availability (P), DoS	Read	OWA	<i>any</i>
CVE-2003-0007	High	3,3	Allows unauthorized disclosure of information	Write	<i>any</i>	Outlook
CVE-2002-1876	High	2,3	DoS	Read	Exchange	<i>any</i>
CVE-1999-0116	Medium	3,3	DoS	<i>any</i>	<i>any</i>	<i>any</i>

Table 4: Example of threats considered relevant for the use case

analyzed to determine where this information is present in the IDMEF alert, under which keyword and form. The reader can refer to the example of CVE-1133 in section 4.1.2 for an example of a definition of a hold fact. Finally, the impact, relevance and CVSS score are used to adapt the context lifetime values from the defaults shown in table 2.

7.2 Threat analysis

Table 4 lists the *(role, activity, view)* or *(subject, action, object)* triples potentially impacted by the vulnerability. This definition corresponds to our understanding of the information available in the description of each vulnerability. We use the *any* keyword to denote that any entity can participate in the vulnerability (the value of this field is not important in exploiting the vulnerability directly), italics to denote concrete entities, and plaintext to denote abstract entities. In the table, we observe the following:

Prevalence of any for roles and subjects There are only three exceptions to this in table 4. There are two explanations to this phenomenon:

1. the query term were oriented towards server vulnerabilities. A search for client-side vulnerabilities would likely yield many results where the role would be a specific server side software. Searching the NVD for “Microsoft Outlook” yields 102 responses, 244 for “Mozilla Firefox”, and 124 for “Mozilla Thunderbird”. While these are the immediate query terms, we could also imagine searching for Microsoft Office vulnerabilities (as Outlook is part of the Office suite) or for specific components such as images, OLE or COM. Limiting ourselves to server-side vulnerabilities helps in ensuring that we will indeed have signature and thus alerts that represent usage of these vulnerabilities. Furthermore, e-mail servers are also a gateway to the outside world (as we will see later in this section), and thus carry more risk.
2. the faulty component has been identified, but the table does not take into account natural dependencies. For example, all the table entries listing *OWA* as the object should list *firefox* as the subject, since it is the email client (web browser) used to connect to *OWA*. Taking these dependencies into account in the table is complicated because we then need to determine whether the client software is impacted by the flaw and how ; for all the cross-site scripting vulnerabilities, it is unclear to us whether *firefox* would effectively be vulnerable and whether the user making the final decision would interact in a dangerous way with the server. We consider that this information should not be part of the decision process, because it is not reliable.

Prevalence of abstract entities as activities Many vulnerability descriptions do not refer to a specific action from the user. We have only two exceptions in the table, *TOP* and *USER*, which refer to specific commands of the *POP3* protocol. In the other cases, we can identify a general activity that the user is performing, but not the exact action ; this would probably be possible if we analyzed attack code, but it is unlikely that a security administrator will have the time to do this from a security report highlighting security risks. He will have to base his decision on the vulnerability description, and we have done so as well.

Prevalence of concrete entities as objects As already noted, our search terms were server-oriented, hence it is quite natural that we obtain concrete software vulnerabilities in this column, attached to concrete software objects. As views and roles are symmetric, the same queries we mentioned for roles would yield *any* for the relevant view. The exceptions appear in only two cases, *CVE-2006-0002* and *CVE-2005-0044*. *CVE-2006-0002* impacts separately both exchange and outlook, as noted in the vulnerability description ; however, in the use case, outlook will be the client that connects to exchange, so the two lines are really identical.

Apparition of a “Transfer” activity Many vulnerability descriptions indicate that the vulnerability can be exploited by sending a message from a remote location. The *Write* activity does not fully reflect this, as mail servers also exchange mail with other outside servers. In our use case, we had not originally taken this dimension into account, and considered a closed email system. A better analysis of table 4 has led to the creation of the *Transfer* activity. With respect to server-side threats, *Write* and *Transfer* activities are very close ; client-side threats (as shown for example in *CVE-2003-0007* do imply that the dialog is between a registered user and an internal server, and do not implicate the *Transfer* activity. This “Transfer” activity would enable us to move from a closed mail system to a more realistic open one, although this is left for future work.

Specific handling of low level threats It seems likely that transport-level threats have a much broader impact than the ones at the higher layers. The only example of low level threat in the table, synflood (*CVE-1999-0116*), can be carried out regardless of the role, activity or view, provided that all roles can inject traffic at the transport layer. We believe that all threats related to traffic injection (land, ping-of-death, ...) are likely to be difficult to include in the model unless specific network-level access-control policies are in place (such as authentication of DHCP requests or network level policy enforcement such as DHCP switching as practiced by ungoliant⁶.

We have also included a line about information level threats with the password guessing activity. This attack can occur against the *Access* activity, but can occur through any of the mail clients against any of the mail servers. This is likely due to the synchronization mechanism that replicate the same user/-password information throughout all channels.

7.3 Revised description of the Policy Components

As explained in section 3, we use contexts to formulate additional policies for threat response. We will derive our threat contexts in a tree fashion, according to the information provided in table 4. The key issue is to create the appropriate contexts and the associated hold facts. As shown in figure 7, we re-use the

⁶<http://ungoliant.sourceforge.net/>

nominal and the *my_mailbox* contexts introduced in section 3.5.3. We introduce the *minimal* and *minimal_mail* contexts to support the minimal security requirements introduced in section 3. We also introduce the threat contexts, that correspond to the threats identified in figure 7.

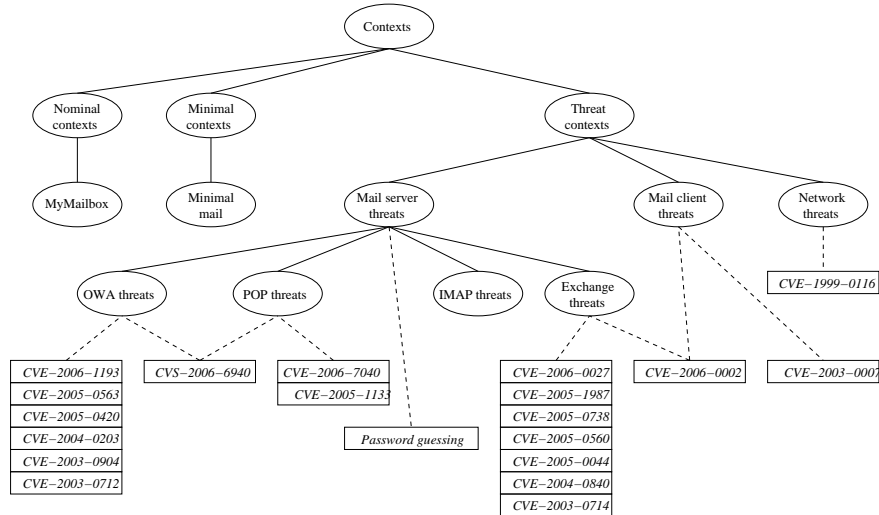


Figure 7: Model of threat contexts

In this case, we separate between network-level threats (where enforcement will occur through filtering devices on the network only), and application (mail)-level threats, where we have more options for enforcement. We further refine mail threats into server-side threats and client-side threats. As a result of our analysis, some of the identified threats are relevant to several contexts.

7.4 Definition of the Security Policy

Listing 8: 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 Section 3, we define the security policy as shown in listing 8. 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 8.

Listing 9: 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) :-
  global_clock (DayClock, TimeClock),
  TimeClock >= '07:00:00',
  TimeClock < '20:00:00',
  DayClock != 'saturday',
  DayClock != 'sunday'.

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

```

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

7.5 The Mapping Predicates

The core of the *hold* predicate related to threats (the first one in listing 9) 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 Section 4.1. 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 Section 6. An example of mapping in this case study is given by listing 10.

Listing 10: 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}
```

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 Section 4.1. 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 Section 4.1, 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.

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] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J.-C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, R. J. Stroud, P. Verssimo, M. Waidner, and A. Wespi. Conceptual model and architecture of MAFTIA. MAFTIA deliverable d21, Malicious- and Accidental-Fault Tolerance for Internet Applications, Project IST-1999-11583, january 2003. <http://www.maftia.org>, last accessed 2007-03-01.
- [2] J.P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, Fort Washington - Technical Report Contract 79F26400, 1980.
- [3] Richard Brackney. Cyber-intrusion response. In *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, pages 413–415, West Lafayette, IN, USA, October 1998. IEEE Computer Society Press.
- [4] F. Cuppens and A. Miège. Alert Correlation in a Cooperative Intrusion Detection Framework. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.
- [5] Frédéric Cuppens, Cuppens-Boulahia, and Alexandre Miège. Inheritance hierarchies in the or-bac model and application in a network environment. In A. Sabelfeld, editor, *Proceedings of the 2004 Foundations of Computer Security Workshop (FCS04)*, pages 41–60, Turku, Finland, July 2004. Turku Center for Computer Science. Report G-31.
- [6] Frédéric Cuppens, Nora Cuppens-Boulahia, and Meriam Ben Ghorbel. High-level conflict management strategies in advanced access control models. In *Workshop on Information and Computer Security (ICS)*, Timisoara, Roumania, November 2006.

- [7] Frédéric Cuppens, Nora Cuppens-Boulahia, Thierry Sans, and Alexandre Miège. A formal approach to specify and deploy a network security policy. In *Proceedings of the Second Workshop on Formal Aspects of Security and Trust (FAST'04)*, Toulouse, France, August 2004. IFIP WCC 2004.
- [8] Frédéric Cuppens, Sylvain Gombault, and Thierry Sans. Selecting appropriate counter-measures in an intrusion detection framework. In *17th IEEE Computer Security Foundations Workshop (CSFW'04)*, page 78, Pacific Grove, CA, USA, June 2004. IEEE Computer Society Press.
- [9] Frédéric Cuppens and Alexandre Miège. Modelling contexts in the or-bac model. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003)*, pages 416–427, Miami Beach, Florida, USA, December 2003. IEEE Computer Society Press.
- [10] Frédéric Cuppens and Alexandre Miège. Administration Model for Or-BAC. *Computer Systems Science and Engineering (CSSE'04)*, 19(3), May 2004.
- [11] O. Dain and R. Cunningham. Fusing a Heterogeneous Alert Stream into Scenarios. In *Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications*, pages 1–13, November 2001.
- [12] Hervé Debar, David Curry, and Ben Feinstein. The intrusion detection message exchange format. RFC 4765, November 2006. <http://www.ietf.org/rfc/rfc4765.txt>.
- [13] S. Floyd. Inappropriate tcp resets considered harmful. RFC 3360, August 2002. <http://www.ietf.org/rfc/rfc3360.txt>.
- [14] M.A. Harrison, W.L. Ruzzo, and J.D. Ullman. Protection in operating systems. *cacm*, 19(8):461–471, August 1976.
- [15] Erland Jonsson. Towards an integrated conceptual model of security and dependability. In *Proceedings of the First International Conference on Availability, Reliability and Security (ARES 2006)*, pages 646–653, Vienna, Austria, April 2006. IEEE Computer Society Press.
- [16] Anas Abou El Kalam, Salem Benferhat, Alexandre Miego and Rania El Baida, Frédéric Cuppens, Claire Saurel, Philippe Balbiani, Yves Deswarte, and Gilles Trouessin. Organization based access control. In *Proceedings of the Fourth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, pages 120–134, Lake Como, Italy, June 2003. IEEE Computer Society Press.
- [17] Michiharu Kudo and Satoshi Hada. Xml document security based on provisional authorization. In *Proceedings of the 7th ACM conference on Computer and communications security (CCS '00)*, pages 87–96, Athens, Greece, November 2000. ACM Press.

- [18] Alexandre Miège. *Definition of a formal framework for specifying security policies. The Or-BAC model and extensions*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, 2005.
- [19] Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducassé. M2D2 : A Formal Data Model for IDS Alert Correlation. In *Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID)*, October 2002.
- [20] P. Ning, Y. Cui, and D. S. Reeves. Constructing Attack Scenarios Through Correlation of Intrusion Alerts. In *Proceedings of the 9th Conference on Computer and Communication Security*, 2002.
- [21] M. Petkac and L. Badger. Security agility in response to intrusion detection. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, page 11, New Orleans, Louisiana, USA, December 2000. IEEE Computer Society Press.
- [22] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *ieeec*, 29(2):38–47, 1996.
- [23] R. Shirey. Internet Security Glossary. RFC 2828, 2000.
- [24] Yohann Thomas, Hervé Debar, and Benjamin Morin. Improving security management through passive network observation. In *Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06)*, pages 382–389, Vienna, Austria, April 2006. IEEE Computer Society Press.
- [25] Thomas Toth and Christopher Kruegel. Evaluating the impact of automated intrusion response mechanisms. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC'02)*, pages 301–310, Las Vegas, Nevada, USA, December 2002. IEEE Computer Society Press.
- [26] Jeffrey D. Ullman. *Principles of database and knowledge-base systems, Vol. I*. Computer Science Press, Inc., 1988.

Index

- access control, 5
- CVE
 - CVE reference, 16, 18, 32, 33
- CVSS
 - CVSS score, 32, 34
 - CVSS vector, 32
- IDMEF, 16, 31
 - alert message, 16
 - IDMEF alert, 33
 - IDMEF alerts, 25, 28
 - IDMEF message, 28, 30, 31
 - IDMEF messages, 16, 23, 24, 29, 31
 - IDMEF message attributes, 23–25
- intrusion prevention, *see also* intrusion prevention systems, 4
- intrusion prevention systems, 1, 4, 5
- National Vulnerability Database (NVD), 32
- Or-BAC, 6, 7, 28, 32, 39
 - Or-BAC contexts, 16
 - Or-BAC framework, 40
 - Or-BAC hold facts, 26, 28, 29
 - Or-BAC instances, 30, 31
 - Or-BAC model, 6–8
 - Or-BAC policy rules, 16, 23
 - Or-BAC rules, 28
 - Or-BAC security policies, 8
- response, 1, 3, 4, 25, 29, 32
 - attack response, 4
 - intrusion response, 4
 - threat response, 16
 - threat response, 4, 5, 10, 16, 36, 39, 40
- response strategy, 26
 - attacker-centric response, 26
 - network-oriented response, 26
 - user-oriented response, 26
 - victim-centric response, 26
- security policies, 6, 10
 - contextual security policies, 6, 28
 - network security policies, 8
- security policy, 1–7, 10, 15, 16, 26, 28, 37, 39
 - security policy description, 28
 - security policy formalism, 5, 39
- threat response
 - threat response mechanisms, 4, 5, 9, 24
 - network-based threat response mechanisms, 4
 - threat response system, 26, 32
- vulnerabilities, 2, 4, 34
 - client-side vulnerabilities, 34
 - cross-site scripting vulnerabilities, 34
 - server-side vulnerabilities, 34
 - software vulnerabilities, 32, 35
 - vulnerability databases, 5
 - vulnerability descriptions, 34, 35
- vulnerability, 26, 32, 34
 - vulnerability description, 35