A Flexible Method for Information System Security Policy Specification

Rodolphe Ortalo

ortalo@laas.fr
LAAS-CNRS
7, avenue du Colonel Roche
31077 Toulouse cedex 4 France

Abstract. This paper presents a method for the specification of the security of information systems. The proposed approach provides a flexible and expressive specification method, corresponding to the specific needs of organizations. First, we outline the overall guidelines of the security policy definition process, and the different consistency issues associated to the description of the security requirements of an organization information system. The specification language used is based on a convenient extension of deontic logic. The formalism and its extensions are defined briefly. To illustrate the use of this formalism, the paper presents how the method applies to the description of the security requirements of a real organization : a medium-size bank agency.
Keywords: security policy specification, information systems, deontic logic.

1 Introduction

This paper presents a practical framework for the specification of the security policy of an organization such as an industrial company or a public service. These organizations exhibit security needs, but they also have other requirements that induce several specificities with respect to security management. First, the information system may exist prior to the explicit definition of its security policy. Furthermore, it is important to provide an adaptable and structured methodology corresponding to the need for abstraction and progressive refinement in the security policy specification process. Finally, the information system may be imperfect, in the sense that the desired security properties may differ from the actual observed properties. From the security point of view, these contradictions appear intolerable, but they often result from a trade-off between the main requirements of the organization and the enforcement of security measures. In fact, the study of practical regulations often reveals discrepancies between the actual situation and the desired behavior, and the security policy would probably better cope with them rather than ignore them.

Consequently, in the context of an organization, most of the classical security models are often unusable. The problem is that they impose constraints on the design and the behavior of the system that may not be acceptable in a situation where security is not a primary requirement. For example, the mandatory rules

of a multi-level security policy [1] imply to design the information system accordingly, and assume permanent application of the rules. Moreover, in the case where the system already exists, the security policy could not be applied until it has been re-designed. To face such situations, the specification of a security policy must provide more flexibility and expressiveness.

In this paper, we use an extended deontic logic language to specify security policies. Several authors already outlined the main features of deontic logic for specifying security properties [9, 12, 4], and this formalism seems to be more adequate for the specification of the security policy of an organization than usual security models. The specification method we propose is based on an original syntactical extension of the standard deontic logic language that allows a structured and convenient representation of the elements of the policy.

Given the imperfections of real information system policies, several types of security policy inconsistencies may be encountered, corresponding to the elements of the security policy that conflict with each other. By detailing these cases, we identify some inconsistencies corresponding to a trade-off between functionality and security in the organization, that may be tolerated in the specification. Furthermore, we argue that the formalism also offers the opportunity to complement the security policy with a description of the vulnerabilities of the information system. However, it is highly desirable to completely describe the insecure states corresponding to inconsistencies in order to have a chance to trust the system.

The structure of the paper is the following: Section 2 presents the various elements of an organization security policy and the potential conflicts that it may contain. Section 3 provides the definition of the specification language used in this paper. Section 4 details the application of the method to an information system. Section 5 presents a practical application example, addressing the security needs of a bank agency. Finally, we conclude in Section 6.

2 Presentation of the Approach

2.1 Security Policy Components

The security policy of the system corresponds to the description of its security needs. According to [10, Section 2.9], the **security policy** is "*the set of laws, rules and practices that regulate how sensitive information and other resources are managed, protected and distributed within a specific system*". In this paper, we consider that a security policy defines:

- the **security objectives**, i.e. the confidentiality, integrity and availability properties expected of the system;
- and the **security rules** which are imposed on the mechanisms which can modify the security state of the system, in order to guarantee the security properties.

The components of the security policy refer to specific entities of the system (e.g. specific subjects or objects, available operations, or an organization chart).

First, these **description elements** are added to the policy. In an organization for instance, the names of the various individuals are mentioned. Additional **description rules** may also be added to the specification in order to reflect the functional behavior of the system when this behavior may impose constraints on the security-related aspects of the system. In an organization, such functional rules represent the various information flows existing in the organization and the way they interact with each other. The description of these operational rules brings information that help the security administrator to adapt the security policy to the actual operation of the organization. The description of the system operation is limited to the basic elements needed to formulate the security mechanisms and objectives, and to the operational rules that describe the portions of the information system relevant to security. Therefore, this description should remain minimal, and additional elements or rules should be introduced only when needed.

Second, the security policy contains the security rules to be enforced by the system. These rules reflect the way in which the security state of the system evolves. In an organization, such rules define, for example, who can assign a new position to an individual, or on which condition someone may delegate some of his privileges to another person.

Finally, the security policy defines several security objectives that specify the desired security properties for this system. The set of all these properties defines the states of the system that are acceptable with respect to security. A security state of the system that satisfies all the security objectives of the system security policy is a **secure** state. An insecure state is a state where these properties are not satisfied.

2.2 Refinement

The first version of the security objectives introduced in the security policy will probably be rather generic. Therefore, it is very probable that these objectives will be too restrictive and easily invalidated by the operation of the organization. To resolve this problem, the introduction of new elements may be necessary to limit the scope of an objective. Similarly, it may be necessary to introduce new description rules, or accurately detail the various steps of a given task in order to focus on a desired security property. Therefore, the description of the organization will undergo several refinement steps, enabling to clarify the security objectives and to improve the whole policy. Furthermore, the verification of the policy can benefit from an incremental approach, as we will see in Sect. 4.2.

2.3 Security Policy Consistency

The security policy is **consistent** if, starting from a secure state one cannot reach an insecure state without violating the security rules. Given the different security policy components, we can distinguish several types of potential inconsistency:

- First, the description of the system may be inconsistent due to conflicting functional rules. However, this is not a security-related specification problem.

- Several security objectives may be contradictive. In this case, no secure state exists and it is impossible to enforce the policy, therefore the security objectives should be updated (e.g. by reducing their scope, or adding priorities).
- Some of the security rules of the specification may contradict each other. For example, an individual may be able to reach a situation in which an obligation and an interdiction hold simultaneously with respect to the same action (e.g. due to conflicting roles). The modification of the security mechanisms may prevent such conflicts. However, it is also possible that the behavior of the system eliminates the situations in which these conflicts can occur. If so, description rules could be added to the system description to describe the functioning more accurately.
- The verification of each security objective, starting with the system security rules, may reveal an inconsistency. In this case, security rules fail to enforce the objectives. Such inconsistency means that either security objectives should be revised, or it is needed to introduce new security rules to enforce the expected properties.
- Finally, it is possible that the functional rules conflict with the security objectives. This case identifies weaknesses in the organization operation showing that it is possible to bypass the security rules. To face this problem, two directions are possible: it is possible to modify the organization operation (and therefore decide to adapt the organization to the envisaged security mechanisms), or to consider new security mechanisms which are compatible with the actual operation (i.e. adapt the security policy to the information system).

We can identify the mandatory consistency requirements of a security policy specification. First, it seems necessary that the consistency of the system description be obtained. If the description of the system functioning is incorrect, adding security requirements to the policy is a nonsense. Second, if several security objectives are contradictive, i.e. if the enforcement of one objective implies the failure of another, the information system is always in an insecure state. Therefore, consistency of the set of all security objectives is necessary: if two security objectives contradict, a choice must be made.

Other inconsistency types reflect problems such as incomplete security mechanisms that fail to enforce the security objectives, over restrictive security rules that may lead to opposed permissions and interdictions in specific situations, or dangerous functioning of the system that may allow to bypass the security mechanisms. In practice, some of the security policies defined in organizations exhibit such inconsistencies [5]. Furthermore, a situation in which such problems occur may be acceptable if it corresponds to a legitimate trade-off in the organization. Exhaustive description of the insecure states is necessary to trust the system, but security issues may not need to be totally reconsidered. The purpose of the introduction of a description of the system behavior in our approach to security policy specification is precisely to allow such precise insecure states identification for some classes of inconsistent policies. Of course, improvements may also be proposed, but the corresponding mechanisms may not be possible

to be enforced in the organization if they perturb its normal operation, and we think that the security policy specification should take this fact into account and cope with some inconsistencies.

This point of view can be extended. We can envisage to complement the system security policy by a description of the potential vulnerabilities of the system. Vulnerabilities may be due to intrinsic weaknesses of the used security mechanisms (such as cryptographic keys length), or to flaws identified after the system has started its operational life [10, Para. 3.25-28], [11, Sect. 6.C]. Such vulnerabilities compromise theoretically the security of the system, but may be very difficult or very long to exploit. The identification of the insecure states due to vulnerabilities allows to assess their impact on the security objectives. This assessment should help the information system security administrator to propose relevant changes given the known vulnerabilities of the system.

3 Specification Language

In order to handle such cases, we need a security policy specification formalism flexible enough to separate the functional description of the system and the security needs description, and which furnishes methods for exploiting inconsistent specifications due to the integration of vulnerabilities in the description. Modal logic, or more precisely deontic logic, is a formal framework that seems to be expressive and adaptable enough to specify security policies of organizations [9, 12, 4]. *Deontic logic is a branch of modal logic to reason about normative versus non-normative behavior by means of modal operators such as ought, permitted and forbidden* [15, Preface]. Furthermore, by using these special operators, deontic logic provides a mean to specify that some behaviors of the system are non-normative (illegal) but nevertheless possible. We present briefly in this section the deontic logic language that we selected. A general presentation of modal logic can be found in [3].

3.1 Deontic Logic Language

Like any modal logic, deontic logic is an extension of usual propositional logic by one or several deontic operators. More precisely, if Φ is a set of atomic propositions a, b, c, ..., if $\neg$, $\vee$, $\wedge$, $\Rightarrow$ and $\Leftrightarrow$ are the usual boolean operators, and if $\mathbf{O}$, $\mathbf{P}$ and $\mathbf{F}$ are the three modal operators of deontic logic, the deontic language denoted $\mathrm{L_O}(\Phi)$, or simply $\mathrm{L_O}$ when it raises no ambiguity, is the set of *formulas* (or *sentences*) built with the following rules:

- if $p \in \Phi$, p is a formula;
- if p and q are formulas, $\neg p$, $p \vee q$, $p \wedge q$, $p \Rightarrow q$, and $p \Leftrightarrow q$ are formulas;
- if p is a formula, $\mathbf{O}p$, $\mathbf{P}p$ and $\mathbf{F}p$ are formulas.

A *modal* formula is a formula of $\mathrm{L_O}$ which contains at least one operator $\mathbf{O}$, $\mathbf{P}$ or $\mathbf{F}$. A *non-modal* formula is a formula which contains none.

In natural language, formulas $\mathbf{O}p$, $\mathbf{P}p$ and $\mathbf{F}p$ are read, respectively, "*It is*

obligatory that p", "*It is permitted that p*", and "*It is forbidden that p*". Therefore, operators **O**, **P** and **F** denote respectively the notions of obligation, permission and interdiction. Conforming to the definitions of necessity (operator $\Box$) and possibility (operator $\Diamond$) in classical modal logic, one can state relations between these various operators. First, **F** can be defined from **O** by (1). **P** is the operator *dual* of **O**. Therefore, one can also consider the relation (2) between operators **O** and **P**, similar to the relation between necessity and possibility in classical modal logic. Finally, standard deontic logic [3, Chap. 5], also contains axiom D which relates obligation and permission.

$$\mathbf{F}p = \mathbf{O}\neg p \quad (1)$$

$$\mathbf{P}p = \neg\mathbf{O}\neg p \quad (2)$$

$$\mathbf{O}p \Rightarrow \mathbf{P}p \quad \mathrm{D}$$

$$\mathbf{O}(p \Rightarrow q) \Rightarrow (\mathbf{O}p \Rightarrow \mathbf{O}q) \quad \mathrm{K}$$

$$\frac{p}{\mathbf{O}p} \quad \mathrm{RN}$$

A modal logic system is an extension of propositional logic, therefore it contains all the axioms and inference rules of propositional logic. Thus, $\mathrm{L}_{\mathbf{O}}$ contains all tautologies and is closed under *modus ponens*. We consider in this work a *normal* modal logic system, i.e. a system that contains axiom K and the inference rule RN ("*Rule of Necessitation*").

In this paper, we want to use deontic logic for the practical representation of security properties. Even though it may have a direct impact on our work, studying deontic logic in itself, or as a formal language for reasoning about legal issues and normative systems (or security), is out of the scope of this presentation. Therefore, we use standard deontic logic, i.e. the KD system of classical modal logic. Especially we accept (2) as a suitable definition of permission. However, these simplifications are done in order to clarify the security policy specification method, and they do not prevent further extensions of the logical language used, such as suggested in [3, Chap. 10].

Finally, we recall the semantics associated to a normal modal logic language like $\mathrm{L}_{\mathbf{O}}$. It is a *Kripke's semantics*, or *possible world semantics* [13]. A Kripke model M for a modal logic system is defined by a triple $\langle \mathrm{W}, \mathrm{R}, \mathrm{V} \rangle$ where W is a set of possible worlds w, R is a binary relation over W called the accessibility relation, and $\mathrm{V} : \mathrm{W} \times \Phi \rightarrow \{\mathrm{true}, \mathrm{false}\}$ is a function furnishing in each world $w \in \mathrm{W}$ the truth value $\mathrm{V}(w,p)$ of the atomic proposition p. The fact that proposition p is true in a world w of model M is noted $\models^{\mathrm{M}}_{w} p$. This truth value is defined in the following way[1], which extends usual propositional calculus:

- if $p \in \Phi$, $\models^{\mathrm{M}}_{w} p$ if and only if $\mathrm{V}(w,p) = \mathrm{true}$
- $\models^{\mathrm{M}}_{w} \neg p$ if and only if we do *not* have $\models^{\mathrm{M}}_{w} p$
- $\models^{\mathrm{M}}_{w} (p \wedge q)$ if and only if $\models^{\mathrm{M}}_{w} p$ or $\models^{\mathrm{M}}_{w} q$
- $\models^{\mathrm{M}}_{w} \mathbf{O}p$ if and only if $\forall w' \in \mathrm{M}/w\mathrm{R}w', \models^{\mathrm{M}}_{w'} p$

The first three statements conform to the classical truth value definition of a formula in propositional logic. From the last definition, we see that in the Kripke's semantics the formula "It is obligatory that p" is true in a world w if and only if p is true in every world w' which w is in relation with. When w is

[1] Using $\neg$ and $\vee$ as the fundamental boolean operators.

in relation with w', we say that w' is *directly accessible* from w, or equivalently that w' is a *possible world* for w . The definition of the truth value of $\mathbf{P}p$ is easily deduced from the one of $\mathbf{O}p$ using (2) :

- $\models^{\mathrm{M}}_{w} \mathbf{P}p$ if and only if $\exists w' \in \mathrm{W}/w\mathrm{R}w', \models^{\mathrm{M}}_{w'} p$

3.2 Language Extensions

The language $\mathrm{L}_{\mathbf{O}}$, or $\mathrm{L}_{\mathbf{O}}(\Phi)$, is based on a set of atomic propositions Φ. In order to ease the specification task, it is desirable to introduce constructs allowing a structured and hierarchical description of the various formulas of $\mathrm{L}_{\mathbf{O}}$. Therefore, we propose to use a syntactical extension of this language, denoted $\mathcal{L}_{\mathbf{O}}$.

Let Π be a set of *sets* $A, B, C, \ldots$, ordered by the partial order relation of inclusion $\subseteq$. The language $\mathcal{L}_{\mathbf{O}}(\Pi)$, or simply $\mathcal{L}_{\mathbf{O}}$ when it raises no ambiguity, is syntactically identical to the language $\mathrm{L}_{\mathbf{O}}$ defined in Sect. 3.1.

This extension is purely syntactical. Its interest resides in the possibility of structuring the atomic propositions of the language through a hierarchical description of a set by its subsets. In order to ease the construction of the various sets E elements of Π, we allow the use of the operator $\times$ (set product) to define the elements of Π, with the natural extension of the relation of inclusion.[2]

The deontic language $\mathrm{L}_{\mathbf{O}}(\Phi_{\Pi})$ associated to the extended language $\mathcal{L}_{\mathbf{O}}(\Pi)$ is built considering the set Φ_{Π} defined as $\Phi_{\Pi} = \{E \in \Pi / (\forall E' \in \Pi, E' \not\subset E)\}$. Therefore, Φ_{Π} is the set of all the terminal sets of Π, i.e. the sets that do not contain any other sets and form the basis of the hierarchy defined by relation $\subseteq$.

A sentence f of $\mathcal{L}_{\mathbf{O}}(\Pi)$ is associated to a set Σ_f of formulas of $\mathrm{L}_{\mathbf{O}}(\Phi_{\Pi})$ defined recursively by (3), where $f|_{x \leftarrow y}$ is the formula built by substituting for all the occurrences of symbol x the symbol y in formula f. Therefore, a formula f of $\mathcal{L}_{\mathbf{O}}(\Pi)$ is a short notation for several formulas of $\mathrm{L}_{\mathbf{O}}(\Phi_{\Pi})$. These formulas are those obtained by replacing, in f, every occurrence of a non-terminal set of Π by its content until we obtain formulas mentioning only elements of Φ_{Π}.

$$\Sigma_f = \bigcup_{E' \subseteq E} \Sigma_{f|_{E \leftarrow E'}} \quad (3)$$

Using Set Constructors. Similar notations allowing the direct use of sets within formulas have already been applied to logical languages developed for database systems [14]. These extensions aim at facilitating the specification task. Sets allow for a progressive and hierarchical description of the basic elements of the security policy, and these sets can be used directly in the formulas of the extended language. To illustrate the use of this notation, we consider now several formulas written in $\mathcal{L}_{\mathbf{O}}(\Pi)$ and the equivalent set of formulas in $\mathrm{L}_{\mathbf{O}}(\Phi_{\Pi})$.[3]

A simple equation of the extended language $\mathcal{L}_{\mathbf{O}}$, e.g. $\mathbf{P}(A \vee B)$, should be represented by a whole set of formulas in $\mathrm{L}_{\mathbf{O}}$, here: $\{\forall p \in A, \forall q \in B, \mathbf{P}(p \vee q)\}$.

[2] $E \times E' = \{(e, e') / e \in E, e' \in E'\}$, and $E \times E' \subseteq F \times F' \Leftrightarrow E \subseteq F \wedge E' \subseteq F'$.

[3] In these examples, sets A, B, 'Administrators', 'Software types' and 'Departments' refer to sets containing atomic propositions only. 'Install' is itself an atomic proposition, i.e. an empty set.

The set product operator $\times$ of $\mathcal{L}_{\mathbf{O}}(\Pi)$ allows the definition of structured sets as elements of the language. This eases the description of various items starting with some of their typical features. Hence, formula (4) states directly a set of permissions associated to software installation tasks. These tasks are described by the action symbol 'Install', the various people allowed to perform the installation gathered in the set of the system administrators, and the various target software. Pieces of software are themselves described through two characteristics: their type (word processor, spreadsheet, etc.), and the organization department that owns them. The description of all these elements in the original language $\mathrm{L}_{\mathbf{O}}$ would necessitate the definition of many elements, sharing common typical characteristics. Formula (5) shows such a set. As can be seen, in a conventional notation, set product corresponds to several indices.

$$\mathbf{P}\left(\text{Install} \times \text{Administrators} \times (\text{Software types} \times \text{Departments})\right) \tag{4}$$

$$\left\{\mathbf{P}\left(\text{Install}_{a,\text{Software}_{t,d}}\right)\right\}_{a \in \text{Administrators}, t \in \text{Software types}, d \in \text{Departments}} \tag{5}$$

Graphical Representation. These extensions also make the specification language suitable for a graphical presentation that can be manipulated with classical user interaction techniques (drag and drop, etc.). This graphical representation represents the sets hierarchy leading to the definition of the atoms of the language using hierarchical lists. The operators of the language are also represented using graphical artifacts called slot-boxes. The purpose of this representation is to provide a simple and convenient method to define and manipulate the security policy specification elements. Figure 1 presents an example of this representation for defining the content of the sets mentioned in (4).

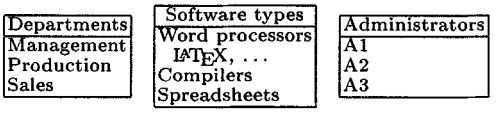

Fig. 1. Example of sets graphical representation

This graphical approach has been implemented within a tool enabling to define and modify easily a security policy. This editor is based on the Amulet graphical library [16] which was very convenient for this realization. In the remaining of this paper, we will adopt this graphical representation for sets, but for space reasons, we will use a conventional linear notation for formulas involving operators.

4 Application to Information Systems

Security needs justifying the elaboration of a security policy appear in many organizations. For example, we can mention banks (transactions integrity), telecommunication companies (system availability), or medical and public-health services (legal confidentiality obligations). However, all these organizations also have other priorities distinct from security. In this section, we present how the

deontic logic language presented previously can be applied to the specification of the security policies of such organizations, and the various possible situations associated to the specification verification.

4.1 Guidelines

In the context of an organization, a security administrator would be in charge of the security policy definition. While this administrator would probably have a good knowledge of the target organization, it is probable that he would not be familiar with a modal logic language. Therefore, we detail several steps of this formalization process.

Reuse of available information. Most of the description elements of the specification appear in documents describing the organization and its operation. These elements are related to physical entities, individuals, or a functional decomposition of the organization. They refer also to specific attributes (such as confidentiality levels). Many of these items appear in description documents already available in the organization, such as an organization chart, some tasks definition documentation, handbooks, norms if applicable, etc. With respect to security issues, other documents help the system administrator. For instance, organization charts also mention the individuals situation in the hierarchy and their privileges. The policy can also be based on existing regulations, documents supporting the evaluation of the organization according to a set of security criteria [10], a previous risk analysis, laws if applicable to the organization, etc. When the security policy specification target is a system to be designed (and not a system in operation), the security designer should look for equivalent information existing in the design documents.

System description. In the deontic logic language presented previously, the description of the organization is performed through the introduction of atomic propositions representing the basic elements, and non-modal formulas (built with the operators of usual propositional logic) representing the aspects of the functioning that are relevant for security. For example, consider the process of internal purchase in an organization: an employee formulates a purchasing demand, which must probably be validated by a manager before going to a specialized department that performs the operation. In order to describe these different steps, first one needs to introduce several atoms such as 'demand', 'manager authorization', 'order' and 'delivery'. Then, one can state the causal dependencies between these steps and the condition for an order to be actually sent by the purchase department. This is represented by logical formulas like: demand $\wedge$ manager authorization $\Rightarrow$ order and order $\Rightarrow$ delivery.

Security rules. The security mechanisms existing in the system are added to the description of its operation. A security mechanism describes the link between the permissions (or interdictions) and the state of the system (or, more precisely, only a subset of the whole state of the system: its security state, made of security attributes). Therefore, it may be represented by a modal formula stating a permission conditioned by some non-modal formula, such as: directory writable $\Rightarrow$ **P** (rename the file).

Integration of roles. The main asset of role-based policies resides in the opportunity that they offer to specify constraints on abstract elements (the roles) instead of enumerating several analogous constraints for all the concerned subjects [19]. Formula (4) presented previously shows that the possibility to use sets directly in a formula in $\mathcal{L}_{\mathbf{O}}$ allows to define and use easily user groups. To support roles, we enumerate not only the various roles, users and permissions, but also the desired pairs (role, user), and use modified authorization rules that constrain users permissions according to user roles. For example, we may modify (4) to: $\mathbf{P}(\text{Install} \times \text{Agent} \times \text{Software}) \Rightarrow \text{Administrator} \times \text{System administrator}$. In this approach, roles appear as intermediate abstract entities between the permissions in the system and the actual users. This extra level improves the flexibility of the specification, as roles are defined independently of the elements they relate. This flexibility is achieved at the cost of an extra specification work to associate users to roles. As shown in [17], several variants of this method can be used to introduce roles in a deontic logic specification.

Security objectives. Finally, we use the basic elements defined previously to describe the security objectives of the information system. If a regulation exists in the organization, it will be transposed in the security policy specification. The deontic logic language allows an easy definition of the security objectives, through the use of the obligation, permission and interdiction operators that allows a natural wording of the desired security properties. Therefore, a security objective is usually represented by one modal formula, such as $\mathbf{O}p$ or $\mathbf{F}(p \wedge \neg q)$.

4.2 Exploiting the Specification

An asset of a formal language resides of course in the availability of formal calculus methods, useful to reason about the security policy. Several directions may be followed to address this issue. See [8, 6] for a general presentation of modal logic proof theory, and [4] for an application to the verification of security properties. Provided that this calculus exists, several verifications can be performed on the specification corresponding to the various inconsistency types identified in Sect. 2.3. With respect to validation problems, the incremental process presented in Sect. 2.2 exhibits new advantages.

Non-modal formulas, such as p, $p \vee q$, $p \wedge q \Rightarrow r$, corresponds to the description of the system components and functioning. Such formulas must form a consistent basis for the security policy. Therefore, validation of all the non-modal formulas should be done separately, prior to further verification of the security-related formulas. As these formulas only form a partial description of the system and are propositional logic formulas, their number is limited and we do not expect their validation to be prohibitive.

When inconsistency comes from the combination of two security objectives (this can be checked directly by verifying the conjunction of the two corresponding modal formulas in the chosen modal system), a choice must be made. This situation may be difficult to manage if it is attempted to verify the whole security policy directly in one step. However, if the specification process follows the

incremental method suggested, the verification of the security objectives compatibility should be done separately[4]. We can also check that the security policy specification exhibit some properties, such as the fact that there are no conflicting obligation and interdiction. This is represented by $\neg(\mathbf{O}p \wedge \mathbf{F}p)$, which must be true in the specification.

Inconsistency types potentially related to an organization trade-off appears mainly when considering the Kripke's semantics of the deontic logic security policy specification. An inconsistency appearing between formulas describing security rules, such as $p \wedge q \Rightarrow \mathbf{F}r$ or $p \Rightarrow \mathbf{P}q$, means that the semantic model M of the specification contains a world w (corresponding to a system state) in which several permissions or obligations conflict (e.g. both $\models_w^{\mathrm{M}} \mathbf{P}p$ and $\models_w^{\mathrm{M}} \mathbf{F}p$ hold for some p and w). Similarly, an inconsistency appearing after the addition of a security objective to the specification shows that there exists worlds in the model where the modal property corresponding to the objective is not true. In each case, a model M invalidating the specification identifies the various intermediary states leading to an insecure state as well as the possible transitions between them, as it provides the truth values $\mathrm{V}(w, p)$ of the atomic propositions in each world, as well as the relation R between the worlds.

The verification process may exhibit a counter-model M invalidating the specification. In this case, the counter-model may be used to identify the various rules (either functional rules or security rules) involved, as well as the conditions on the state of the system that allow this failure. With this information, modifications may be proposed. If the functioning of the system never allows such situations (e.g. one person simultaneously associated to all the roles in the system), additional formulas may be incorporated to the security policy to rule out such states. If a succession of applications of security rules allows to defeat the proposed objective, either the rules may be updated or the objective revised. Finally, if the behavior of the system allows to bypass several security rules and defeat the security objective, modifications of the operation of the system may be proposed in order to improve its security. In each case, the trade off between security issues and operational issues may be difficult to manage, but the information extracted from the counter-model M helps the system administrator to determine the appropriate solution. The interest of the information that can be extracted from such a counter-model may favor the use of a semantics-based deduction method for the verification of a deontic logic security policy specification, such as those presented in [7, 2].

We see that the specification, even when inconsistent, can be exploited usefully. In fact, we argue that the ability to analyze precisely and exhaustively the various possible evolutions of the state of the system that contradict a security

[4]It should be noted that, with the language presented in Sect. 3, it is also possible that global inconsistencies, similar to those corresponding to conflicting objectives, result from a combination of propositional formulas (describing the functioning) and formulas describing security rules. This specific case corresponds to the fact that in a specific system state some security rules lead to a contradiction, and, simultaneously, the functioning of the system always leads to this state. This case should be treated like conflicting objectives and be eliminated.

objective is very important in the context of an organization information system. In this context, the verification of the whole specification is very likely to fail due to contradictory objectives, or incomplete or inadequate security rules. Furthermore, it may be acceptable to tolerate inconsistencies, if all the evolutions of the information system that may lead to an insecure state are very improbable or very difficult to exploit. This assessment can rely, for example, on an evaluation method [18]. However, such situations cannot be understood if a precise description of the security needs does not provide the information needed to qualify the insecure states.

4.3 Description of Vulnerabilities

The approach finally offers the opportunity to include in the security specification additional components potentially leading to inconsistencies, representing the vulnerabilities of the information system. In (6), we define a right d_p with respect to a proposition p as the permission of p. We propose to define a vulnerability as the fact that, given a right and a specific state of the system, it is possible to obtain a new right (possibly not legitimate). We describe this as the property that "if a non-modal (propositional) property p and a right d_q are true in a world w, then there exist another world w' in relation with w such that another right d_r is true in w'."

$$d_p = \mathbf{P}p \quad (6)$$

$$p \wedge d_q \Rightarrow \Diamond d_r \quad (7)$$

$$p \wedge \mathbf{P}q \Rightarrow \Diamond \mathbf{P}r \quad (8)$$

Therefore, we represent a vulnerability by a rule like (7). Given the definition of a right, a vulnerability is of the form of (8). This definition allows the description of general vulnerabilities and, on the contrary of [18], makes a clear distinction between the normative aspects of the specification (captured by the deontic operator $\mathbf{P}$) and the opportunities offered to a potential attacker (captured by the ontic operator $\Diamond$).

5 Example

To illustrate the security policy specification method, we present an example of its application. The target organization is a bank agency, counting 35 employees, and located in a rural area. The security policy presented in this section was built on the basis of several internal documents describing the organization. The analysis of these documents has been completed by a study in the field which lasted several days. The bank help manual identifies more than a hundred different operations potentially performed in the organization, from foreign currency deposit to account freezing following a legal injunction. Among all these operations, only a limited number are integrated in the following specification. We chose to consider operations involving loan agreement, and bearer bonds. These tasks are interesting as the first relies on a delegation mechanism, and the other is considered security-critical due to anonymity.

5.1 Formalization of the Organization Description Documents

The initial elements of the specification are presented in Fig. 2. We note the various individuals mentioned in the policy, several actions, and also a set of roles. The set 'Individuals' is further split into two subsets: 'Customers' is left unspecified and designates any potential customer of the bank, 'Agents' enumerates all the employees of the organization (they are identified by numbers from 1 to 35 in this paper). The roles used in this specification are related to the functions existing in the organization. All the official functions mentioned in the organigram are gathered in the set 'Employment' (defined later) which is therefore a subset of all roles. However, our practical study allowed us to identify a separate unofficial role, 'Cashier', necessary to formulate the security rules of the organization. Astonishingly, it is not explicitly associated to a function in the organigram. Finally, the set 'Actions' contains the various tasks included in the specification. Only three general types of action are considered in this example: common operation performance, loan agreement, and physical access.

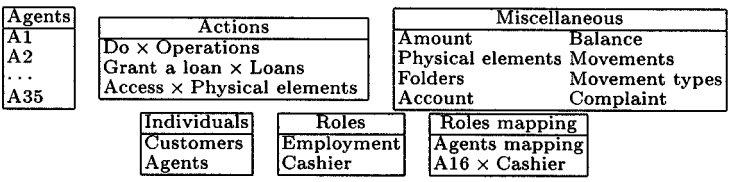

Fig. 2. Security policy specification root

The various organization description elements are given in Fig. 3. The set 'Employment' describes the employee functions (corresponding to roles). It corresponds to a position description document available in the bank. This document identifies several abstract positions that gather different positions actually held by the employees (e.g. 'Leader' groups positions such as agency leader, office manager, etc.). The roles introduced in Fig. 2 allow to define generic description and security rules independently of the actual agents of the organization. To obtain the actual description we need to specify also a mapping between individuals and roles. The mapping of agents to roles, 'Agents mapping', obtained using an organigram of the organization, is defined in Fig. 3. We note in this figure that one agent may be associated to several roles, e.g. A7 is a manager as well as a service agent. A16, the agent playing the role of the cashier is identified directly in Fig. 2. The various operations performed in the organization, presented in the set 'Operations', are also organized in several subsets. The description is limited to the higher levels and the detailed list of all operations has not been included. In this figure, we also introduce several secondary elements. Finally, we include in these description elements a simple representation of the data associated to a bank account.

In order to describe the functioning of the organization, we also need to include description rules. As an example, we show in Fig. 3 how accounts and money movements can be represented. These rules state for example that an usual operation necessitates physical access to one of the bank terminals, and

that such an operation results either in a debit or credit money movement on an account. Of course, this is a simplification of the real functioning. A specific description rule represents the fact that a debit operation, when performed on a negative balance account, follows an exceptional path. The proposition 'Freezing' is a label identifying the exception. Here, this label does not mean that the operation cannot be performed.[5]

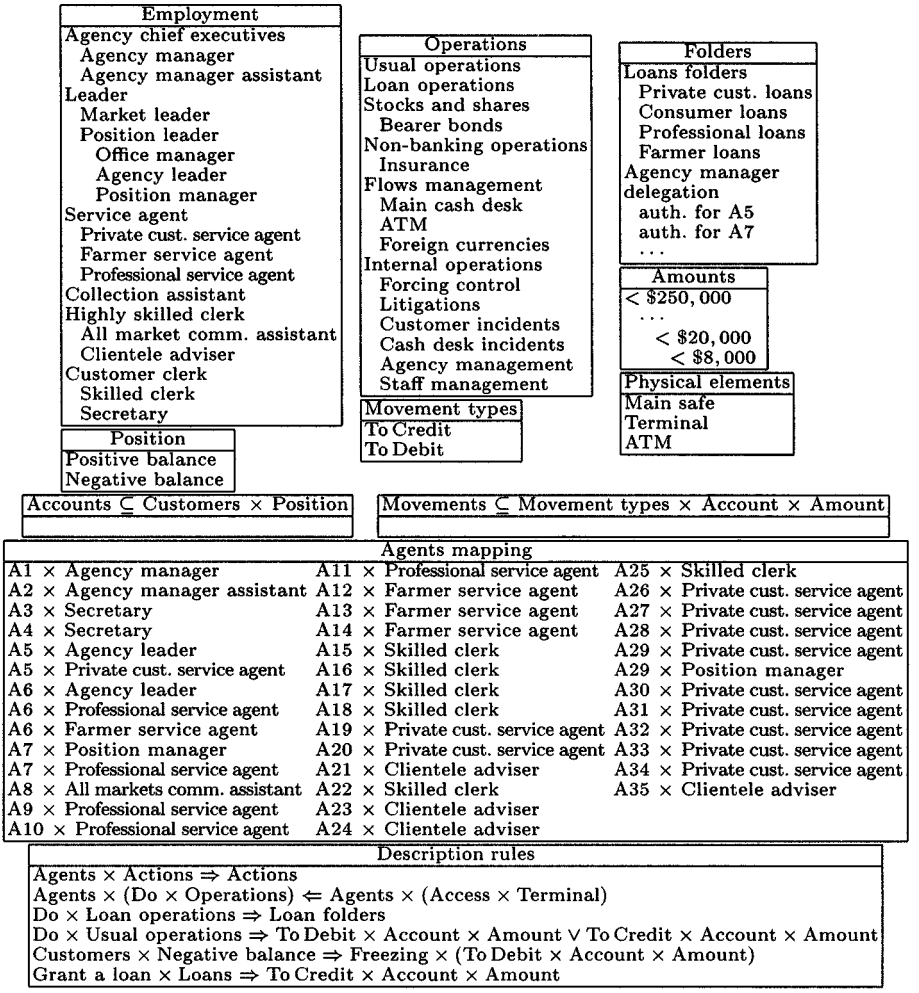

Fig. 3. Security policy description elements

[5]Debit operations on a negative balance account can be eliminated by a formula such as: Customers × Negative balance ⇒ ¬ (To Debit × Account × Amount). We do not include such a rule in our specification because it does not correspond to the actual functioning of the bank. Agents treat such money movements with specific care, but they can perform them.

5.2 Description of the Security Needs

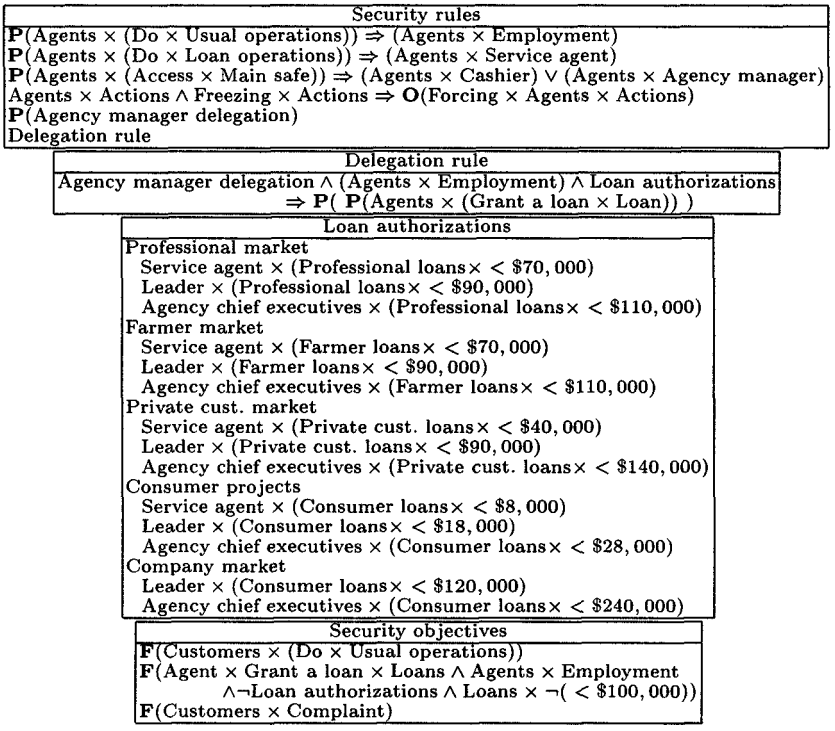

Fig. 4. Security requirements description

The rules describing the security needs of the organization are shown in Fig. 4. Among these rules, we focus on a specific aspect involving delegation of privileges: loan agreement delegation. Chief executives are allowed, in this organization, to delegate some of their privileges to other employees, in the limit fixed by several conditions. The rule itself is shown in the table entitled 'Delegation rule', and the second table 'Loan authorizations' enumerates the loans categories and maximal amounts that may restrict the delegation rights of the agency manager. Note that the abstract positions identified formerly in Fig. 3 (e.g. 'Leader') are used to define the various authorizations, illustrating the interest to describe hierarchically the set 'Employment'. The mechanism of loan agreement delegation that exists in this bank is a typical example of a situation in which a deontic logic specification language may be most useful. The general bank rules state that an agency manager can delegate some of his privileges (with respect to loan decisions) to other employees. These rules define several limits that the agency manager must respect in his delegation decisions, but nothing more. Therefore, the manager defines the actual permissions of the agents. We represent this functioning in the specification using two nested **P** operators meaning that permission is granted (to the agency manager) to grant permission to one of his agents to accept a loan. However, even if it is authorized, the manager may not

delegate full permissions to an agent, e.g. if the latter does not need them as he works on a specific market. Finally, Fig. 4 presents three simple security objectives. The first one is related to customers, but the other two apply to the agents of the organization. The second one aims at enforcing the delegation authorizations with respect to loan agreement, and the last one at protecting customers from abuses perpetrated by employees.

5.3 Integration of Vulnerabilities

As an example, we consider now two vulnerabilities of the bank agency. These vulnerabilities are presented in Fig. 5. The first one is related to trust relationships between the various agents of the organization. We also consider that a customer may trust the agent in charge of his business. Such trust may allow a malicious agent of the organization to misappropriate funds of this customer. However, in our case, we think that this vulnerability arise only for specific operations, more precisely anonymous stocks and shares operations.

Vulnerabilities
Agents × Trusts × Other agent ∧ **P** (Agents × Actions) ⇒ ◇**P** (Other agent × Actions)
Customers × Trusts × Agents ∧ (Agents × Bearer bonds) ⇒ ◇ (¬ (Customers × Complaint))

Fig. 5. Vulnerabilities

The first vulnerability shown in Fig. 5 conflicts with the second security objective shown in Fig. 4. A study in the field allowed us to check in the real organization the various conditions under which a given vulnerability may appear (i.e. to obtain all the true propositions of the form Agents × Trusts × Other agent). Given this information, we identified all the possible scenarios allowing an agent of the organization to defeat the second or third security objective. We observed that trust relationships were numerous in this organization between the employees (174 trust relationships are very strong), and several agents indeed could defeat the loan agreement authorization limits. As it is surely impossible (and probably ill-fated) to regulate the everyday life and the trust relationships existing among the agents, these results do not allow one to influence directly the operation of the organization. However, comparison of these results with the situation of each agent (with respect to his seniority, position or skills) could reveal anomalies in privileges distribution in the organization. In our case, no abnormal tendency arose from the results analysis.

In the specification considered in this example, the second vulnerability conflicts directly with the third security objective. We studied an alternate functioning of the target organization involving validation of all bearer bonds operations by one specific employee before completion. In this case, only a malicious agent who abuses one of his customer's trust *and* the agent in charge of the validation is able to infringe the objective. This means that, given the revised functioning, both vulnerabilities shown in Fig. 5 may conflict with the third security objective of the organization. However, detailed analysis of the possible insecure states shows that they are much less numerous given the revised functioning than with the original one. Therefore, such functioning appears acceptable.

6 Conclusion

The information systems addressed in our study are commonly found in organizations such as banks, industrial companies, etc. Our approach focuses on organizations where security is not the only requirement even though security issues are important. For these organizations, the definition of a security policy necessitates a pragmatic approach, taking into account the functioning of the target information system. Most of the classical security policies define security properties expected of the system independently of a description of the system itself. But such policies also impose strong constraints on the operation of the target system or organization, which may not be acceptable in the context of a conventional information system. In our case, on the contrary, the security policy definition integrates a partial description of the information system. This allows to reuse available information, and to adapt the definition of the security objectives and rules to the target organization structure and behavior. In practice, security rules and objectives may be more detailed and fit better the actual operation of the organization.

Based on a formal language, deontic logic, the method offers the opportunity to perform the verification of the policy. But we identified several types of inconsistency, and some of them correspond to admissible trade-off between security and other requirements of the organization, such as flexibility or efficiency. In these cases, the policy fails to be consistent. However, the verification of the security policy should provide a mean to identify exhaustively all the potential insecure states of the information system and could be a valuable help to the security administrator who manages such trade-off.

Furthermore, a suitable description of the information system vulnerabilities can be proposed using the proposed specification language. This description fruitfully complements the security policy as the formalism offers the opportunity to identify precisely the impact of the vulnerabilities considered on the security of the organization, using verification mechanisms similar to those associated to consistency checking. This information will facilitate the task of the administrator to evaluate the gravity of the organization vulnerabilities, and to propose modifications of the functioning or of security rules in order to improve security.

Acknowledgments

The author is grateful to the Crédit Agricole, and more especially to the CRCAM Quercy-Rouergue for their collaboration, as well as to all of the staff of the agency of Villefranche de Rouergue for their pleasant welcome. This work has been partially supported by UAP Assurances, and European ESPRIT Project 20072 "Design for Validation" (DeVa). Finally, I would like to thank Claudia Almeida, Marie Borrel, Frédéric Cuppens, Yves Deswarte and the anonymous referees for their help on this work.

References

1. D. E. Bell, L. J. LaPadula, *Secure Computer Systems: Unified Exposition and Multics Interpretation*, The MITRE Corporation, Report ESD-TR-73-306, 1975.
2. L. Catach, "TABLEAUX: A General Theorem Prover for Modal Logics", *Journal of Automated Reasoning*, vol. 7, pp. 489-510, 1991.
3. B. F. Chellas, *Modal Logic: An Introduction*, 295 p., ISBN 0-521-29515-7, Cambridge University Press, 1980.
4. L. Cholvy, F. Cuppens, "Analyzing Consistency of Security Policies", in *IEEE Symposium on Security and Privacy*, Oakland, California, May 4-7, pp. 103-112, ISBN 0-8186-7828-3, IEEE Computer Society Press, 1997.
5. F. Cuppens, C. Saurel, "Specifying a Security Policy: A Case Study", in *9th IEEE Computer Security Foundations Workshop*, Kenmare, Ireland, June 10-12, pp. 123-134, ISBN 0-8186-7522-5, IEEE Computer Society Press, 1996.
6. L. Fariñas del Cerro, A. Herzig, "Modal Deduction with Applications in Epistemic and Temporal Logic", in *Handbook of Logic in Artificial Intelligence and Logic Programming, Epistemic and Temporal Reasoning* (D. M. Gabbay, C. J. Hogger, J. A. Robinson, Eds.), vol. 4/5, pp. 499-594, ISBN 0-19-853791-3, Oxford Science Publications, 1995.
7. M. Fitting, "First-Order Modal Tableaux", *Journal of Automated Reasoning*, vol. 4, no. 2, pp. 191-213, 1988.
8. M. Fitting, "Basic Modal Logic", in *Handbook of Logic in Artificial Intelligence and Logic Programming*, Logical Foundations (D. M. Gabbay, C. J. Hogger, J. A. Robinson, Eds.), vol. 1/5, pp. 365-448, ISBN 0-19-853745-X, Oxford Science Publications, 1993.
9. J. Glasgow, G. McEwen, P. Panangaden, "A Logic for Reasoning About Security", in *Computer Security Foundations Workshop*, Franconia, pp. 2-13, IEEE Computer Society Press, 1990.
10. ITSEC, *Information Technology Security Evaluation Criteria*, v1.2, 163 p., ISBN 92-826-3004-8, Office for Official Publications of the European Communities, Luxembourg, 1991.
11. ITSEM, *Information Technology Security Evaluation Manual*, v1.0, 262 p., ISBN 92-826-7087-2, Office for Official Publications of the European Communities, Luxembourg, 1993.
12. A. J. I. Jones, M. Sergot, "Formal Specification of Security Requirements using the Theory of Normative Positions", in *Second European Symposium On Research In Computer Security (ESORICS 92)*, (Y. Deswarte, G. Eizenberg, J.-J. Quisquater, Eds.), Toulouse, France, November 23-25, LNCS, 648, pp. 103-121, ISBN 3-540-56246-X & 0-387-56246-X, Springer-Verlag, 1992.
13. S. A. Kripke, "Semantical Considerations in Modal Logic", *Acta Philosophica Fennica*, vol. 16, pp. 83-94, 1963.
14. G. Kuper, "Logic Programming with Sets", in *6th ACM Conference on Principles of Database Systems (PODS)*, San Diego, California, USA, March 23-25, pp. 11-20, ISBN 0-89791-223-3, ACM Press, 1987.
15. J.-J. C. Meyer, R. J. Wieringa (Eds.), *Deontic Logic in Computer Science*, 317 p., ISBN 0-471-93743-6, Jon Wiley & Sons, 1993.
16. B. A. Myers, R. G. McDaniel, R. C. Miller, A. S. Ferrency, A. Faulring, B. D. Kyle, A. Mickish, A. Klimovitski, P. Doane, "The Amulet Environment: New Models for Effective User Interface Software Development", *IEEE Transations on Software Engineering*, vol. 23, no. 6, pp. 347-365, June, 1997.
17. R. Ortalo, *Using Role-Based Abstractions for Security Policy Specification with Deontic Logic*, 20 p., LAAS-CNRS, Report 97216, June, 1997.
18. R. Ortalo, Y. Deswarte, "Quantitative Evaluation of Information System Security", in *14th IFIP International Information Security Conference (IFIP/SEC'98)*, August 31-September 4, Vienna-Budapest, Austria-Hungary, Chapman & Hall, 1998. *(to appear)*
19. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role-Based Access Control Models", *IEEE Computer*, vol. 29, no. 2, pp. 38-47, February, 1996.