Secure Dependencies with Dynamic Level Assignments

Pierre Bieber  
Frédéric Cuppens  

ONERA-CERT  
2 Av. E. Belin  
31055, Toulouse Cedex  
France  
email: {bieber,cuppens}@tls-cs.cert.fr

Abstract

Most security models explicitly (or implicitly) include the tranquillity principle which prohibits changing the security level of a given piece of information. Yet in practical systems, classification of objects may evolve due to declassification and subject current level may evolve according to subject requests.

In [2], we proposed a modal logic definition of security whose counterpart is a constraint on the system traces that we called causality. In this paper, we give a generalization of causality which avoids the tranquillity principle.

We give an interpretation of our model in the case of a multilevel security policy when the levels can be assigned dynamically. Then we provide efficient conditions to control the dynamic assignment of both the object classification and the subject current level. We propose a comparison of our approach with the non-deducibility generalization of [15]. Finally, we give several examples of systems where security levels are dynamically assigned.

Introduction

Most of security models explicitly (or implicitly) include the tranquillity principle which prohibits changing the security level of a given piece of information. Many reasons show that it could be useful to avoid this principle. In particular, dynamic level assignment makes it easier to model systems in which:

1. A secret user could decide to perform an unclassified job. In this case, the inputs performed by subjects are not always classified on the basis of their fixed clearance level.

2. There are resources shared by subjects that have different clearances and the classification of the resources depends on the level of the subject using them. In this case the classification level of a resource may change.

3. A given piece of data needs to be declassified. In this case the classification level of an object may decrease.

In the Bell-LaPadula model [1], it is possible to model a system in which levels are dynamically assigned. Each object is associated with a classification level that may be modified by an action performed by a subject. Each subject is associated with a constant clearance level and a current level that may be changed by the subject. According to Bell and LaPadula, a system is secure if the "no read up" and "no write down" requirements are both satisfied. However, these requirements are not adequate to control level modification. As an example, J. McLean [12] proposed the system Z that has only one type of action: when a subject requests any type of access, every subject and object is downgraded to the lowest possible level. It is easy to see that system Z is secure according to the Bell-LaPadula model. But, as this system will give all subjects access to all objects, it cannot be generally considered as secure. This example shows that the Bell-LaPadula model is not adequate to control dynamic level assignment. In [12], J. McLean reports that it has been argued that this model implicitly includes the tranquillity principle in order to avoid level modification problems.

Other security models such as the non-interference security model of [9], the non-deducibility model of [14] or the restrictiveness model of [11] also include this principle. Actually, there exists only few attempts
to avoid it. A first one is proposed by J. McLean in [13]. It is an extension of the Bell-LaPadula model based on the access-matrix approach. Let us denote \( \mathcal{A} \) the set of subjects and \( \mathcal{O} \) the set of objects. Each subject \( A \in \mathcal{A} \) is associated with a set \( \mathcal{c}_A(A) \) of subjects who can change the security level of \( A \) and each object \( o \in \mathcal{O} \) is associated with a set \( \mathcal{c}_O(o) \) of subjects who can change the security level of \( o \). The classical Bell-LaPadula model corresponds to the case \( \forall A \in \mathcal{A}, \forall o \in \mathcal{O}, \mathcal{c}_A(A) = \mathcal{c}_O(o) = \mathcal{A} \) and Bell-LaPadula supplemented by tranquillity corresponds to the case \( \forall A \in \mathcal{A}, \forall o \in \mathcal{O}, \mathcal{c}_A(A) = \mathcal{c}_O(o) = \emptyset \). It is also possible to specify the intermediate position held by the SMMS model [10] where a subject \( sso \in \mathcal{A} \) is designated system security officer, and \( \forall A \in \mathcal{A}, \forall o \in \mathcal{O}, \mathcal{c}_A(A) = \mathcal{c}_O(o) = \{sso\} \). However, it seems that, in this approach, only trusted subjects can be authorized to change a security level. To authorize a non-trusted subject to change a security level, we need a model based on information flow controls as non-interference or non-deducibility. Such an approach is proposed by I. Sutherland, S. Perlo and R. Varadarajan in [15]. They give a generalization of non-deducibility which allows security levels to be assigned dynamically. In this case, every change of a security level must satisfy the non-deducibility requirement. In this paper, we draw our inspiration from another approach proposed by G. Eizenberg in [6].

In [2], we proposed to define security by the logical formula \( K_A \varphi \rightarrow B_A \varphi \) that could be read "If \( A \) knows \( \varphi \) then \( A \) should have the permission to know \( \varphi \)." The counterpart of this logical definition is a constraint on the system traces that we called causality. We model a security policy by associating each subject with an authorized role. In [2], we assumed that the authorized role is fixed, this assumption is similar to the tranquillity principle.

In this paper, we first give in section 1 a generalization of our security model which avoids this principle. This leads to a generalization of the causality constraint when the subjects authorized roles are not fixed. In section 2, we give an interpretation of our model in the case of a multilevel security policy when the levels can be assigned dynamically. As in the Bell-LaPadula model, we consider that each object is associated with a classification level and each subject is associated with a clearance level and a current level. In section 3 and 4, we provide efficient conditions to control the dynamic assignment of both the object classification and the subject current level. In section 5, we propose a comparison of our approach with the non-deducibility generalization of [15]. Finally, in section 6, we give several examples of systems where security levels are dynamically assigned including a level reservations system, a secure shared resource and several declassification methods.

## 1 A Definition of Security

Here, we propose a security model that can be used as a possible-worlds model for the security logic (see [3]). A security model has two parts, the first one is a general model of computer systems, the second one provides a definition of security:

**Definition 1** The model is a five-tuple \( \mathcal{S} =< \mathcal{S}, \mathcal{O}, \mathcal{D}, \mathcal{A}, \mathcal{T} > \) where:

- \( \mathcal{O} \) is the set of objects, it is partitioned into three subsets:
  - \( \text{In} \): the set of input objects
  - \( \text{Out} \): the set of output objects
  - \( \text{Intern} \): the set of internal objects
- \( \mathcal{D} \) is the domain of value of the objects (this set should contain the undetermined value \( \text{Null} \)).
- \( \mathcal{T} \) is the set of time points, we assume that \( \mathcal{T} \) is the set of integers.
- \( \mathcal{S} \) is a subset of \( \mathcal{E} \), where \( \mathcal{E} \) is the set of total traces of system \( \mathcal{S} \).
- \( \mathcal{A} \) is the set of subjects of system \( \mathcal{S} \).

Every function \( X : \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{P}(\mathcal{O} \times \mathcal{T}) \) that associates each trace \( s \in \mathcal{S} \) at each time \( \tau \in \mathcal{T} \) with a set of pairs \( \text{object, time} \) is called a role. The set of roles corresponds to the set of every observation that virtual agents could perform. We identify a subject (a real agent) with the role it plays in system \( \mathcal{S} \), i.e., for each trace \( s \in \mathcal{S} \) and time \( \tau \in \mathcal{T} \), the set of pairs \( \text{object, time} \) such that this subject can observe its values. As some roles could not be played in system \( \mathcal{S} \), the set \( \mathcal{A} \) is a subset of the set of roles.

**Definition 2** We note \( s|\alpha \), where \( \alpha \) is a subset of \( \mathcal{O} \times \mathcal{T} \) and \( s \) is a trace of \( \mathcal{S} \), the restriction of function \( s \) to \( \alpha \).

If \( A \) is a subject, then, \( s|\mathcal{A}(s, \tau) \) models the observation of subject \( A \), at time \( \tau \) when the system ran according to the trace \( s \).
Definition 3 Two traces s and s' are equivalent with respect to A observation at time r if:

\[ s[A(s, r) = s'[A(s', r)] \]

The second part of a security model contains everything that actually deals with security. This part should enable computer designers to decide whether a specification enforces a security property. In this paper, we are only concerned with confidentiality.

Confidentiality may be defined (see [2],[8]) by the following formula of the modal logic of security:

\[ K_{A,r}\varphi \rightarrow R_{A,s}\varphi. \]

This formula could be read "If, at time r, A knows that \( \varphi \) then, at time r, A has the permission to know \( \varphi \)." In ([2, 3]) we showed how the model of systems we just described could provide a semantics for the \( K_{A,r} \) and \( R_{A,s} \) operators:

- The semantics of the epistemic operator \( K_{A,r} \) is defined thanks to the accessibility relation: equivalence of observation with respect to A. In trace s, subject A, at time r, knows \( \varphi \) if \( \varphi \) is true in every trace s' equivalent to s with respect to A's observation at time r.

- The semantics of the deontic operator \( R_{A,s} \) depends on the security policy to be enforced. We consider that a policy associates each subject A in each trace s at each time point r with a set \( R_A(s, r) \) of authorized roles. In trace s, subject A, at time r, is authorized to know a formula \( \varphi \) if A can know \( \varphi \) by playing a role X that belongs to \( R_A(s, r) \).

In [2] we proposed a constraint on traces that is a counterpart of the modal logic definition of confidentiality and called it causality.

Definition 4 System S is secure (with respect to a subject A) if for every traces s and s' and every time \( r \), there is an authorized role of A, X, such that if s and s' are equivalent with respect to X's observation at time \( r \) then s and s' are equivalent with respect to A's observation at time \( r \).

\[ \forall r \in T, s \in S, s' \in S, \exists X \in R_A(s, r), s[A(s, r) = s'[A(s', r)] \]

2 Interpretation of a multilevel security policy

In a multilevel context, each object \( o \) is associated with a classification level and each subject \( A \) is associated with a clearance level. For the reasons we exposed in the introduction, it could be convenient to let levels evolve. Consequently, as in the Bell-LaPadula model, we will associate to every subject a current level which represents the sensitivity of actions performed by this subject at a given time. So the definition of a multilevel security policy is the following:

Definition 5 A multilevel security policy associated to a system \( \langle S, O, D, A, T \rangle \) is a tuple \( \langle L, I, L, c \rangle \) where:

- \( L \) is a set of levels. We assume that \( L \) is a lattice. We denote \( \leq \) the ordering relation in this lattice and we denote \( U \) the lowest possible level in the lattice.
- \( I \) is a function from \( O \times T \times S \) into \( L \) called classification.
- \( L \) is a function from \( A \) into \( L \) called clearance.
- \( c \) is a function from \( A \times T \times S \) into \( L \) called current level.

As we want to model systems in which the level of data could be changed, we consider that the classification of an object depends on both trace and time. Similarly, each subject is associated with a current level which also depends on trace and time. But, for the sake of simplicity, we assume that the clearance level \( L \) associated with each subject is fixed. We could also consider that this security level could be changed, but, in this case, the operation which computes the clearance level must be completely certified. Moreover, this operation should only be performed by the security administrator.

We want to associate each subject with its authorized roles. In [5], we have studied the circumstances in which a subject can have several authorized roles. This is useful when the security policy contains an aggregation exception [7] as in the Brewer-Nash model [4]. In the following, we will not be concerned with aggregation. So we can consider that each subject is associated with only one authorized role and we will identify this singleton with the set of authorized roles \( R_A(s, r) \). Before defining this role in the case of a multilevel security policy, we have to make some remarks on the meaning of the classification, clearance and current level functions:

- If \( o \) is an input object, then \( l(o, r, s) = l_1 \) means that, in trace \( s \), at time \( r \), the system expects that object \( o \) receives the input performed by a subject whose current level is \( l_1 \). As in [15], we
consider that it is up to measures external to
the machine to ensure that if the machine ex-
pects an input of a given level to occur (or not),
then this input will really occur (or not) on the
basis of information of level \( l_1 \). These
external measures include safeguards such as
authentication. Moreover, we must assume that if a secret
subject decides to perform an unclassified ses-
sion, then it only performs inputs on the basis
of its unclassified knowledge. This last rule im-
plies that a subject must be aware of its current
level.

Now, consider a subject \( A \) whose clearance is \( l_2 \)
such that \( l_2 \geq l_1 \). This subject has the permis-
sion to observe every piece of information whose
sensitivity is less than \( l_2 \).

- If \( o \) is an internal object, then we assume that
initially (i.e. at \( \tau = 0 \) ) the security level of every
internal object is correctly assigned by the secu-

ity administrator. So, \( l(o, 0, s) = l_1 \) means that
initially the object \( o \) contains a piece of information
whose sensitivity is equal to \( l_1 \). Hence, a
subject \( A \) of clearance \( l_2 \) such that \( l_2 \geq l_1 \) has
initially the permission to observe object \( o \) at time \( \tau \) in
trace \( s \).

When the system runs, this piece of information
as well as the classification of object \( o \) may be
modified. It is the role of information flow con-
trol to ensure that, if \( l(o, \tau, s) = l'_1 \), then the
information stored in the object \( o \) is always at a
lower level than \( l'_1 \). In particular, if the system
is not secure it could be dangerous to give a sub-
ject \( A \) whose clearance is \( l_2 \) such that \( l_2 \geq l_1 \)
the permission to observe the object \( o \) at time
\( \tau > 0 \).

Consequently, \( A \) has, a priori, the permission to
observe \( o \) in trace \( s \) only at time \( \tau = 0 \). However,
in some cases, a subject may explicitly have the
permission to observe an internal object at time
\( \tau > 0 \). This case can occur when the value of
object \( o \) at time \( \tau \) is computed by a certified
process (for example, a process which performs
encryption or declassification). For the moment,
we do not consider this problem, but, we will
come back on it in section 5.

- If \( o \) is an output object then \( l(o, \tau, s) = l_1 \) means
that the system expects the object \( o \) to be ob-
served by a subject whose clearance is \( l_1 \). As for
inputs, this system's assumption must be en-
sured by external measures. Moreover, as for
internal object, it is the role of information flows
control to ensure that the output object \( o \) does
not contain any information with a sensitivity
greater than \( l_1 \).

After these remarks, we can now define for every
\( s \in S \) and \( \tau \in T \), the authorized role of \( A \) by the
following. First, we define the \( \text{Init}_A \) function by:

\[
\forall s \in S, \text{Init}_A(s) = \{ (o, 0) \mid o \in \text{Intern} \\
\land l(o, 0, s) \leq L(A) \}
\]

\( \text{Init}_A(s) \) corresponds to the set of internal objects \( A \)
has initially the permission to observe; those that have
a classification dominated by \( A \)'s clearance.

Then, we define the \( \text{In}_A \) function by:

\[
\forall s \in S, \text{In}_A(s) = \{ (o, \tau) \mid o \in \text{In} \\
\land l(o, \tau, s) \leq L(A) \}
\]

\( \text{In}_A(s) \) corresponds to the set of inputs performed
by subjects whose current levels are dominated by \( A \)'s
clearance.

The authorized role of \( A \) in trace \( s \) at time \( \tau \) is
defined by:

\[
\forall A(s, \tau) = \{ (o, \tau') \mid (o, \tau') \in (\text{Init}_A(s) \cup \text{In}_A(s)) \\
\land \tau' \leq \tau \}
\]

\( \forall A(s, \tau) \) corresponds to a subject who observes:

1. The initial value of every internal object whose
classification is less or equal to \( A \)'s clearance.

2. The inputs performed before \( \tau \) by subjects
whose current levels are less or equal to \( A \)'s
clearance.

3 Security conditions with dynamic
assignment of object classification

Let \( < L, l, c > \) be a multilevel security policy.
We first define, for every subject \( A \in A \), the \( O_A \) func-
tion by:

\[
\forall s \in S, \forall \tau \in T, \\
O_A(s, \tau) = \{ (o, \tau') \mid l(o, \tau', s) \leq L(A) \land \tau' \leq \tau \}
\]

\( O_A(s, \tau) \) represents the set of pairs (object,time) that
the subject \( A \) can observe when it works at the con-
stant level \( L(A) \). This set \( O_A(s, \tau) \) actually represents
a superset of subject $A$'s real observation in trace $s$ at time $\tau$. In particular, we assume that $A$ cannot directly observe internal objects. In this section, we want to find sufficient conditions to ensure causality with respect to $O_A$, i.e.:

$$\forall s \in S, \forall s' \in S, \forall \tau \in T, s[R_A(s, \tau)] = s'[R_A(s', \tau)]$$

$$\rightarrow s[O_A(s, \tau)] = s'[O_A(s', \tau)]$$

Definition 6 The classification function $I$ is securely defined with respect to $A$ if and only if the following conditions are satisfied:

1. $\forall s \in S, \forall s' \in S, \forall o \in Out,$
   $$s(o, 0) = s'(o, 0) \land l(o, 0, s) = l(o, 0, s')$$

2. $\forall s \in S, \forall s' \in S, \forall \tau > 0,$
   $$\forall (o, \tau) \in (O_A(s, \tau) - (\text{In} \times T)),$$
   $$s[O_A(s, \tau - 1)] = s'[O_A(s', \tau - 1)]$$
   $$\rightarrow s(o, \tau) = s'(o, \tau) \land l(o, \tau, s) = l(o, \tau, s')$$

3. $\forall s \in S, \forall s' \in S, \forall (o, \tau) \in R_A(s, \tau),$   $$s[R_A(s, \tau)] = s'[R_A(s', \tau)]$$
   $$\rightarrow l(o, \tau, s) = l(o, \tau, s')$$

The first condition is the initial condition. It says that, at $\tau = 0$, the value and classification of each output object are identical in every trace.

The second condition is the inductive condition. It says that, if subject $A$ can observe a non-input object in trace $s$ at a positive time $\tau$, and if the traces $s$ and $s'$ are equivalent with respect to $O_A$ at time $\tau - 1$, then the value and classification of this object are identical in both traces $s$ and $s'$. For instance, let us consider an internal object $o$ whose classification is secret at time $\tau - 1$ and unclassified at time $\tau$. Since, an unclassified subject can observe the object $o$ at time $\tau$, this second condition guarantees that the value and classification of object $o$ at time $\tau$ must be determined by unclassified information.

The third condition says that if $A$ has the permission to observe the object $o$, it must also have the permission to observe the classification of this object. In a concrete system, we can implement this requirement by storing the classification of every object $o \in O$ in another object, denoted $lev_o$:

$$\forall o \in O, \exists lev_o \in O, \forall \tau \in T, s(lev_o, \tau) = l(o, \tau, s)$$

If $l(lev_o, \tau, s) \leq l(o, \tau, s)$, then this third condition is satisfied.

We will illustrate these conditions in more details in section 6.

Fact 1 If the classification function $I$ is securely defined with respect to $A$, then causality with respect to $O_A$ is satisfied.

This result provides efficient conditions to prove the security of a system if we assume that a subject always works at its maximal level $L(A)$. However, this assumption is too restrictive to be practical in a real system. If we want to model a system in which a secret user could decide to perform an unclassified job, we must also provide conditions to control the dynamic assignment of the subject current level. These conditions are given in the following section.

4 Security conditions with dynamic assignment of subject current level

Let $<L, I, L, c>$ be a multilevel security policy. In this section, we define, for every subject $A \in A$, the $CA$ function by:

$$\forall s \in S, \forall \tau \in T,$$

$$CA_A(s, \tau) = \{ (o, \tau') \ | \ l(o, \tau', s) \leq c(A, \tau', s)$$

$$\land \tau' \leq \tau \}$$

$CA_A(s, \tau)$ represents the set of pairs (object, time) that subject $A$ can observe when it works at its current level. This set is also a superset of subject $A$'s real observation in trace $s$ at time $\tau$.

We want to find sufficient conditions to ensure causality with respect to $CA_A$, i.e.:

$$\forall s \in S, \forall \tau \in T,$$

$$s[R_A(s, \tau)] = s'[R_A(s', \tau)]$$

$$\rightarrow s[CA_A(s, \tau)] = s'[CA_A(s', \tau)]$$

Definition 7 The current level function $c$ is securely defined with respect to $A$ if and only if the following conditions are satisfied:

1. $\forall s \in S, \forall \tau \in T,$
   $$c(A, \tau, s) \leq L(A)$$

2. $\forall s \in S, \forall s' \in S,$
   $$c(A, 0, s) = c(A, 0, s')$$

3. $\forall s \in S, \forall \tau > 0$
   $$\rightarrow s[O_A(s, \tau - 1)] = s'[O_A(s', \tau - 1)]$$
   $$c(A, \tau, s) = c(A, \tau, s')$$

The first condition says that the subject current level must always be dominated by its clearance level.

The second condition says that, initially, the current level of a subject is the same in every trace.

67
The third condition is the induction condition for the current level. It says that if the traces \( s \) and \( s' \) are equivalent with respect to \( O_A \) at time \( \tau - 1 \), then the current level of \( A \) must be the same in both traces \( s \) and \( s' \). This means that the current level of \( A \) at time \( \tau \) must be determined by information that \( A \) can previously observe. We will also illustrate this condition in more details in section 6.

Fact 2 If the classification function \( l \) and the current level function \( c \) are both securely defined with respect to \( A \) then causality with respect to \( C_A \) is satisfied.

This result provides efficient conditions to prove the security of a real system. Indeed, to enforce causality with respect to \( A \), we simply have to show that:

\[
\forall s \in S, \forall \tau \in T, A(s, \tau) = C_A(s, \tau) \cap W_A
\]

where \( A(s, \tau) \) is subject \( A \)'s real observation and \( W_A \) is a "window" used by subject \( A \) to observe the system. This last condition must be ensured by external measures. In particular, the outputs must be observed by subjects whose clearances are correct with respect to the output classification.

5 Comparison

In this section, we propose a comparison of our approach with the "Deducibility Security with Dynamic Level Assignment" of I. Sutherland, S. Perlo and R. Varadarajan [15]. The comparison is made through an example. We consider a system with only two subjects \( A \) and \( B \). The clearance levels of \( A \) and \( B \) are respectively unclassified (\( U \)) and confidential (\( C \)). The current level of \( B \) is initially unclassified and \( B \) can perform an action which raises its current level at confidential for one unit of time. \( A \) can ask the system about the current level of \( B \).

We first give the system \( S_1 = \langle S_1, O_1, D_1, A_1, T \rangle \) which corresponds to this informal description. Then, we associate with this system a multilevel security policy.

- \( A_1 = \{ A, B \} \)
- \( O_1 \) contains:
  - Two input objects \( \text{Req}_A \) and \( \text{Req}_B \) which respectively receive the inputs performed by \( A \) and \( B \).
  - One output object \( \text{Ans}_A \) which is observed by \( A \). For the sake of simplicity, we assume that \( B \) does not observe any output.
  - One internal object \( \text{CurLevel} \) which is used to store \( B \)'s current level.

\( D_1 = \{ \text{Level}, \text{RaiseLevel}, U, C, \text{Null} \} \)

\( S_1 \) is the set of total functions \( s \) from \( O \times T \) into \( D \) which satisfy the following conditions for every \( \tau > 0 \):

1. \( s(\text{Ans}_A, 0) = \text{Null} \land s(\text{CurLevel}, 0) = U \)
2. \( (s(\text{Req}_A^?, \tau - 1) = \text{Level}\land s(\text{CurLevel}, \tau - 1) = I) \rightarrow s(\text{Ans}_A, \tau) = I \)
3. \( s(\text{Req}_A^?, \tau - 1) \neq \text{Level} \rightarrow s(\text{Ans}_A, \tau) = \text{Null} \)
4. \( (s(\text{Req}_B^?, \tau - 1) = \text{RaiseLevel}\land s(\text{CurLevel}, \tau - 1) = U) \rightarrow s(\text{CurLevel}, \tau) = C \)
5. \( (s(\text{Req}_B^?, \tau - 1) \neq \text{RaiseLevel}\land s(\text{CurLevel}, \tau - 1) = C) \rightarrow s(\text{CurLevel}, \tau) = U \)

The first condition describes the initial state: \( A \) observes a null output and the current level of \( B \) is unclassified.

The conditions 2 through 5 describe the system transitions. We assume that the time of computation for a given action is always equal to 1.

The second condition says that if \( A \) performs the action \( \text{Level} \) then the system will provide \( A \) with \( B \)'s current level. The third condition says that if \( A \) does not perform this action then it will receive a null output.

The fourth condition says that if \( B \) performs the action \( \text{RaiseLevel} \) when its current level is unclassified then its current level will be raised at confidential. The fifth condition says that if \( B \) does not perform this action or if its current level is already confidential then, in both cases, its current level will become unclassified. This last condition implies that \( B \) cannot remain at the confidential level during more than one unit of time.

We associate this system with a multilevel security policy \( < L, I, L, c > \) defined by the following conditions:

- \( L = \{ U, C \} \) and \( U \preceq C \)
The current level function $c$ are both securely defined. In the proof is the following. If we will not give here the formal proof of this result is performed on the basis of unclassified information. Moreover, this input also determined that why the subject depends on unclassified information. This is the reason why the subject $A$ can observe $B$'s current level. The problem will be completely different if we slightly modify the transitions 4 and 5 of system $S_1$:

4. $s(ReqB?, \tau - 1) = Raise\_level$
   \[ \rightarrow \] $s(Cur\_level, \tau) = C$

5. $s(ReqB?, \tau - 1) \neq Raise\_level$
   \[ \rightarrow \] $s(Cur\_level, \tau) = U$

In this case, $B$ can decide to remain at the $C$ level. This implies that the change in $B$'s current level could depend on an input performed by $B$ when its current level is $C$. This input can be performed on the basis of confidential information, so this last system is not secure with respect to our security conditions.

Now, let us return to our initial system $S_1$ and let us analyze the security of this system with the non-deducibility condition proposed in [15]. Following their approach, we need to define a function of $S$ which takes a trace $s \in S$ and returns the view of trace $s$ seen by $A$, denoted $View_A(s)$. In our model, we can set $View_A(s) = s([Low(s)]$ with:

$$\begin{align*}
Low(s) &= \{(o, \tau) \mid l(o, \tau, s) = U\} \\
&= \{(ReqB?, \tau) \mid l(ReqB?, \tau, s) = U\} \\
&\cup \{ReqB?, AnsA!, Cur\_level\} \times T
\end{align*}$$

We need also to define a function of $S$ which takes a trace $s \in S$ and returns the part of trace $s$ which is supposed to be hidden from $A$, denoted $Hidden\_from_A(s)$. In our model, we can set $Hidden\_from_A(s) = s([High(s)]$ with:

$$\begin{align*}
High(s) &= \{(o, \tau) \mid l(o, \tau, s) = C\} \\
&= \{(ReqB?, \tau) \mid l(ReqB?, \tau, s) = C\}
\end{align*}$$

In [15], a system is considered secure if it satisfies the non-deducibility condition: for every pair of traces $s$ and $s'$ there exists a trace $s''$ such that:

$$View_A(s'') = View_A(s) \land Hidden\_from_A(s'') = Hidden\_from_A(s')$$

Clearly, this last condition cannot be satisfied by our example. Indeed, we can find a first trace $s$ such that $s(AnsA!, 1) = U$ and a second trace $s'$ such that $s'(ReqB?, 0) = Raise\_level$. But, we cannot find a third trace $s''$ such that $s''(AnsA!, 1) = U \land s''(ReqB?, 0) = Raise\_level$. So, according to non-deducibility, our example is not secure.

However, in our example, $A$ does not learn anything about $B$'s confidential inputs. In particular, $B$ may perform only null inputs when it works at confidential level. $A$ only learns that $B$ is working at confidential level. According to non-deducibility, this would be a security violation. We do not agree with this point of view and consider that this is an important restriction of the approach proposed in [15]. Moreover, from our point of view, this restriction seems to be inherent to the non-deducibility condition.

Indeed, if the object classification depends on the trace of the system, then it is generally possible to find a trace $s$ in which an object $o$ has a given classification $l_1$ and a given value $v_1$ at time $\tau$ and another trace $s'$ in which the same object $o$ has a different classification $l_2'$ and a different value $v_2'$ at the same time $\tau$. Then, let us consider a subject $A$ whose clearance $l_2$ is such that $l_2 \geq l_1$ and $l_2 \neq l_2'$. As $(o, \tau) \in View_A(s)$ and $(o, \tau) \in Hidden\_from_A(s')$, it would not be possible to find a trace $s''$ such that: $(s''(o, \tau) = s(o, \tau)) \land (s''(o, \tau) = s'(o, \tau))$. Hence, it would not be possible to satisfy the non-deducibility condition.
6 Examples

In this section we give several examples of systems where security levels are dynamically assigned. The facts presented in sections 3 and 4 are used in order to prove that these systems are correct with respect to the causality property.

6.1 Current Level Reservation

As indicated in the introduction, one reason for dynamically assigning security levels is that subjects may need to work at lower level than their clearance. The first example we consider models a system that changes the current level of subject according to level reservations issued from subjects. Once the new current level is set, the system also forwards to another system requests issued from subjects working at the new current level.

In order to enforce causality this system performs several controls. First of all, the system controls that the level at which a subject wants to work is dominated by its clearance. Moreover, the system must also control how current levels change. In the following we explain why.

As we want subject \( X \) to be aware of its current level, the system has an output object whose value is the current level of \( X \). The classification level of this object should be unclassified to authorize every subject to observe it. In order to let subject \( X \) feed the system with information whose sensitivity corresponds to its current level, the classification level of the input object that receives the request issued by the subject must be equal to the current level of this subject. The reason why the system should control how the current level changes is that the value of the unclassified object containing the current level of \( X \) may depend on the inputs performed by \( X \) at its current level. Hence, the system will forbid \( X \) to issue level reservation when the current level is not \( U \). Otherwise, the value of the unclassified object containing the current level would depend on the value of a more classified object.

As we want a subject to have the ability to change its current level several times in a run, the system automatically resets the current level to \( U \) from time to time. Once the current level is set to \( U \), subjects are authorized to issue new level reservations.

The system \( S^C_0 = S^C, O^C, D^C, A^C, T \) we consider is such that:

- \( O^C \) contains:
  - For each subject \( X \in A^C \), an input object, \( Req_x \), whose value is the request issued by subject \( X \). There is another input object \( Signal \) whose value corresponds to the current level reset signal.
  - For each subject \( X \in A^C \), an internal object, \( Clearance_x \), whose value is the clearance of subject \( X \).
  - For each subject \( X \in A^C \), two output objects, \( Req_x \), whose value is the forwarded requests and \( Lev_x \), whose value is the current level of \( X \).

- \( D^C = \{ Null, Reset \} \cup \{ (Req) \times Requests \} \cup \{ (Rsv) \times L \} \) where \( Requests \) is a set of request values and \( L \) is a set of levels.

- \( T = \mathbb{N} \)

- \( S^C \) is the set of total functions \( s : O^C \times T^C \rightarrow D^C \) such that, for each subject \( X \in A^C \):

  1. Initially no request issued by \( X \) is forwarded and its current level is \( U \).
     \[ s(Req_x, 0) = \text{Null} \land s(Lev_x, 0) = U \]
  2. The value of the internal object \( Clearance_x \) is equal to the clearance of subject \( X \).
     \[ \forall r, s(Clearance_x, r) = L(x) \]
  3. If, at time \( r - 1 \), a current level reset signal is received then, at time \( r \), the current level is set to \( U \).
     \[ \forall r > 0, \quad s(\text{Signal}, r - 1) = \text{Reset} \quad \text{then} \quad s(\text{Lev}_x, r) = U \]
  4. If, at time \( r - 1 \), the subject issues a reservation for level \( L \) and \( L \) is dominated by the subject clearance and no current level reset signal is received then, at time \( r \), the current level is set to \( L \).
     \[ \forall r > 0, \forall L \in \mathbb{L}, \quad s(Req_x, r - 1) = (Rsv, L) \land \]
     \[ s(Lev_x, r - 1) = U \land \]
     \[ s(Clearance_x, r - 1) < L \land \]
     \[ s(\text{Signal}, r - 1) = \text{Null} \quad \text{then} \quad s(\text{Lev}_x, r) = L \]
  5. If the subject issues a request, at time \( r - 1 \), and no current level reset signal is received then, at time \( r \), the request is forwarded and the current level is not modified.
     \[ \forall r > 0, \forall R \in requests, \forall L \in \mathbb{L} \]
We define the classification and current level functions \( l \) and \( c \):

- the current level of subject \( X \) is equal to the value of \( Lev_X^1 \).
  \[ c(X, \tau, s) = s(Lev_X^1, \tau) \]
- the security level of the input and output of subject \( X \) are equal to the current level of \( X \).
  \[ l(Reqx?, \tau, s) = c(X, \tau, s) \wedge l(Reqx!, \tau, s) = c(X, \tau, s) \]
- the classification level of the other objects is \( U \).
  \[ l(Signal?, \tau, s) = U \wedge l(Levx!, \tau, s) = U \wedge l(Clearance_X, \tau, s) = U \]

In order to prove that the previous system is secure it is sufficient to prove that the functions \( l \) and \( c \) are securely defined with respect to \( A \) and \( B \). We list the argument used in order to prove this statement.

- condition 1 of definition 6 and 2 of definition 7 are satisfied because initially, there is only one value for objects \( ReqX \) and \( LevX \). Hence, there is only one classification level for these objects and only one current level for subjects.
- condition 1 of definition 7 is satisfied (i.e. the current level of \( X \) is dominated by its clearance). This is implied by constraints 2 and 4 on traces of system \( S^C \).
- condition 2 of definition 6 is satisfied because:
  - the value of \( Clearance_X \) at time \( \tau \) is determined by the values of the unclassified object \( Clearance_X \) at time \( \tau - 1 \).
  - the value of \( LevX \) at time \( \tau \) is determined by the value, at time \( \tau - 1 \), of unclassified objects \( Signal? \), \( LevX \) and by the value, at time \( \tau - 1 \), of \( ReqX \) when the classification level of this object is \( U \).
  - the value of \( ReqX \) at time \( \tau \) is determined by the value, at \( \tau - 1 \), of the unclassified object \( Signal? \) and by the value, at time \( \tau - 1 \), of object \( ReqX \) when \( ReqX \) and \( ReqX \) have the same classification level.
- condition 3 of definition 6 is satisfied because the classification of objects is unclassified or stored in unclassified objects.

- condition 3 of definition 7 is satisfied because the value of \( LevX \) is determined by the value of unclassified objects.

The level reservation mechanism we provide is secure but is not very user friendly. Indeed, the amount of time subject \( X \)'s current level will remain constant does not depend on \( X \)'s requests. Another version of level reservation system could take into account requests containing a level and an amount of time the subject wishes to work at this level. When a new current level is set, a counter containing the time amount asked by the subject is also set, and when the amount of time is finished the system will issue a current level reset signal for subject \( X \). This new system is secure provided that the counter is an unclassified object.

The other limitation of the system we provided is that only level reservations issued at unclassified level are authorized. We could modify another time the system in order to work in high water mark mode (see [7]). At every current level, it should be possible for a subject to issue level reservations that increase its current level. We now consider that the classification level of \( LevX \) is equal to \( X \)'s current level. This new system is secure because the fact that the value of \( LevX \) at time \( \tau \) depends on the value of \( ReqX \) at time \( \tau - 1 \), is not a problem because the level of \( LevX \) at \( \tau \) dominates the level of \( ReqX \) at \( \tau - 1 \). Notice that in this mode it is not possible for a subject to decrease its current level.

### 6.2 Shared Resource

A potential source of covert channels in a system is the use of resources shared by several subjects that have different clearances. A high level subject could transmit information to a low level subject just by using or not a resource shared by these two subjects. Examples of shared resources can be found everywhere from a microprocessor, where registers are shared by several subjects that want to perform memory transfers, to medium access protocols as CSMA-CD or Token ring, where the communication medium is shared by several subjects that want to transmit messages.

In the following we study a very abstract (and incomplete) description of this problem. We show that it is possible to provide a secure shared resource by assigning to it a classification level that depends on the current level of the subject using it. We want to model a system that receives two requests issued by two subjects and that can forward only one request at a time. The system \( S^M = \langle S^M, O^M, D^M, A^M, T \rangle \) we consider is such that:
Thanks to the We associate each object with a classification level and the level of Subject X and the value of Level X is the current level of Subject X.

- One internal object Choice that is used to solve the conflict between the two subjects and to decide which request will be forwarded.
- Two output objects, Req! and Lev! corresponding to the values of the input objects ReqX? and LevX? of the chosen subject.

- \( D^M = \{A, B, \text{Null}\} \cup \text{Request} \cup \mathcal{L} \)
- \( T = \mathbb{N} \)
- \( S^M \) is the set of total functions \( s : O^M \times T \rightarrow D^M \) such that, for all \( X \in A^M \):
  1. Initially no request is forwarded, the current level of the output is \( U \) and the chosen subject is \( A \).
     \[ s(Req!, 0) = \text{Null} \land s(Lev!, 0) = U \land s(Choice, 0) = A \]
  2. If, at time \( r-1 \), the chosen subject is \( X \), the request of Subject X is \( R \) and its current level is \( L \) then, at time \( r \), the value of Req! is \( R \) and the value of Lev! is \( L \).
     \[
     \forall \tau > 0, \forall \text{Req} \in \text{Requests}, \forall \text{Lv} \in \mathcal{L}, \\
     s(Choice, \tau - 1) = X \land s(Req?, \tau - 1) = R \land s(Lev?, \tau - 1) = L \\
     \rightarrow s(Req!, \tau) = R \land s(Lev!, \tau) = L
     \]
  3. If at time \( r-1 \), the chosen subject is \( A \) then, at time \( r \), the chosen subject will be \( B \).
     \[
     \forall \tau > 0, \\
     s(Choice, \tau - 1) = A \\
     \rightarrow s(Choice, \tau) = B
     \]
  4. If at time \( r-1 \), the chosen subject is \( B \) then, at time \( r \), the chosen subject will be \( A \).
     \[
     \forall \tau > 0, \\
     s(Choice, \tau - 1) = B \\
     \rightarrow s(Choice, \tau) = A
     \]

We associate each object with a classification level thanks to the \( l \) function:

1. the level of ReqX? is equal to the value of LevX? and the level of LevX? is \( U \):
   \[
   \forall X \in \{A, B\}, \\
   l(ReqX?, \tau, s) = s(LevX?, \tau) \land l(LevX?, \tau, s) = U
   \]
2. the level of Req! is equal to the value of Lev! and the level of Lev! is \( U \):
   \[
   l(Req!, \tau, s) = s(Lev!, \tau) \land l(Lev!, \tau, s) = U
   \]
3. the level of the choice indicator is \( U \):
   \[
   l(Choice, \tau, s) = U
   \]

We could prove that the function \( l \) is securely defined with respect to \( A \) and \( B \). We list the arguments used in order to prove this statement.

- condition 1 of definition 6 is satisfied because initially, there is only one value for objects Req! and Lev!. Hence, there is only one classification level for these objects.
- condition 2 of definition 6 is satisfied because:
  - the value of Choice, at time \( r \) is determined by the value of the unclassified object Choice at time \( r - 1 \).
  - the value of Lev!, at time \( r \) is determined by the value, at time \( r - 1 \), of unclassified objects Choice, LevA?, LevB?.
  - the value of Req!, at time \( r \), is determined by the value, at \( r - 1 \), of object ReqX? when ReqX? and Req! have the same classification level.

The system we provide is secure but very constraining. The conflict resolution mechanism neither depends on the value of the request issued by the subjects nor depends on the current levels of the subjects. The problem we face is that output objects, in particular Lev!, depend on the value of Choice. If Lev! is an unclassified object then the value of Choice should depend only on unclassified objects. Hence Choice cannot depend on the value of ReqA? and ReqB? whenever \( A \) or \( B \) does not work at unclassified level.

Notice that in the particular case where both subjects work at the same current level, the value of Lev! depends only on the values of LevA? and LevB? and does not depend on the value of Choice. In this case it should be possible to handle conflict resolution depending on the value of the requests. In the general case, one way to provide this service could be to hook-up this system with a system that would guarantee that all incoming requests have the same current level. This solution like level multiplexing of the resource. The interest of such a method is that we could use in a secure fashion existing conflict resolution schemes that were developed without security awareness.
6.3 Declassification

Dynamic level assignment is also useful to model a system in which the level of a given piece of data may need to be changed. The security conditions we impose to the classification function \( l \) in the definition 6 allow object declassification. For instance, an object \( o \) which is secret at time \( \tau \) can become unclassified at time \( \tau + 1 \). However, if the classification function \( l \) is securely defined, the condition 2 of definition 6 guarantees that the value (and also the classification) of object \( o \) at time \( \tau + 1 \) must only depend on unclassified information. This condition rules out J. McLean's system of definition 6. We propose two approaches to solve this problem.

The first one is called external declassification. The declassification is done by removing the data from the system and reintroducing it as if it were a new input at the new level (see figure 1). The reclassification can be done:

1. Manually. In this case the declassifier is a human being.
2. Automatically. In this case, the declassifier must be a trusted process and its use must be controlled by the security administrator.

As the input performed at \( \tau + \delta \) is explicitly unclassified, then every subject whose clearance is unclassified can observe this input. So, we can continue to apply our security requirements to the system. But, notice that these security requirements do not apply to the declassifier.

The second approach is called internal declassification. In this case, the declassification process is described when we specify the system security policy. For instance, let us call \( Ins \) the secret input we must declassify and let \( A \) be a subject whose clearance is unclassified. Then, we will specify \( A \)'s authorized role by the following:

\[
\begin{align*}
\text{Input } U & \text{ at } \tau + \delta \\
\text{System} & \\
\text{Declassifier} & \\
\text{Output } S & \text{ at } \tau
\end{align*}
\]

Figure 1: External declassification

Notice that, when applied to the example proposed by I. Sutherland, S. Perlo and R. Varadarajan in [15], our security requirements for the classification functions are too strong. They consider a system in which the secret inputs performed at time \( \tau \) are always declassified after a fixed interval \( \delta \). Let us consider that the secret input performed at time \( \tau \) is stored in an internal object \( o \) which is secret until \( \tau + \delta \) and becomes unclassified after \( \tau + \delta \). Then, after \( \tau + \delta \), the value of the unclassified object \( o \) depends on previously secret information. This represents a violation of requirement 2 in definition 6. We propose two approaches to solve this problem.
This means that A has the permission to observe what was the Ins value at time $\tau$ only after $\tau + \delta$. Then, we can correctly apply causality for this security policy. In particular, if the Ins value at time $\tau$ is stored in an internal object $o$ which is secret until $\tau + \delta$ and becomes unclassified after $\tau + \delta$ and if this internal object is not modified by another secret user between $\tau$ and $\tau + \delta$, then according to causality, A can observe object $o$ value at time $\tau + \delta$. We do not need an auxiliary trusted process in this case.

7 Conclusion

In this paper, we have extended the notion of security, proposed in [2], in order to take into account dynamic level assignments. The extension is straightforward because the definition of security we use is rather general. It was sufficient to consider that the set of observed objects and the set of authorized roles should depend on the current classification of objects and on the current level of the subjects.

Furthermore, we proposed conditions that allow to prove that a system is secure in a more convenient fashion. These conditions are quite similar in spirit to the unwinding theorem developed for non-interference (see [9]), because it is sufficient to study the system step by step in order to prove that the system is secure. Notice that these conditions control in a similar fashion the value of observed objects and the level of these objects. In the previous examples, the security proof was simplified because for every object in the system whose classification is not fixed there is an unclassified object which stores the level of the object. Hence it was sufficient to prove that the system enforces the conditions on the values of observed objects in order to prove that the conditions on their levels are enforced. The various examples we provided showed that evolving levels do not complicate security proofs.

Moreover, dynamic level assignment could provide simple solutions to problems that might be difficult to solve with fixed levels as the secure shared resource example. The level reservation scheme we studied uses of course evolving current level of subjects but the proposed solutions also depend on evolving classification of objects. We were able to investigate different solutions to the level reservation problem, thanks to the separation between the specification of the system under consideration and the definition of the level functions. This modular description of systems is interesting if one wants to change the security policy without changing the system specifications. Finally we studied several ways to perform data declassification. We investigated both internal and external declassification. Internal declassification is more elegant because we can completely analyze the security of the system including the declassification process. We have shown that causality can be used to perform this analysis. But, in this case, we need to adapt our step by step conditions. This is an area for further work.

References


