Distributed Systems

Implementing Secure Dependencies over a Network by Designing a Distributed Security SubSystem

Bruno d'AUSBOURG

CERT - ONERA
Département d'Etudes et de Recherches en Informatique
2, Avenue E. Belin - B.P. 4025
31055 Toulouse - Cedex- FRANCE
email: ausbourg@tls-cs.cert.fr

It was recently argued that the presence of covert channels should no longer be taken for granted in multilevel secure systems. Until today, multilevel security seems to have been an ideal to approach and not a requirement to meet. The question is: is it possible to design a practical multilevel system offering full security? Based on which architecture? The approach described in this paper reflects some results of a research project which suggests some ideas to answer this question. We have chosen the distributed architecture of a secure LAN as an application framework. In particular we show how controls exerted on dependencies permit to control exhaustively the elementary flows of information. The enforced rules govern both the observation and the handling of data over the whole system. They are achieved by means of some hardware mechanisms that submit the access of hosts to the medium to a secure medium access control protocol. We evaluate how secure dependencies used to ensure confidentiality in such an architecture may also be used to answer some other needs with respect to other attributes of security.

1 Introduction

Many systems were designed in order to protect confidentiality of data and processes. This can be done by building multilevel architectures of machines and networks. These architectures tolerate the existence of covert channels, because standards consider that covert channels are inevitable. Proctor and Neumann in [14] argued that the presence of covert channels should no longer be taken for granted in multilevel secure systems. Indeed, applications should not tolerate any compromise of multilevel security, not even through covert channels of low bandwidths. They argued also that systems with multilevel processors seem to be either impractical or insecure. They suggest to redirect research and development efforts towards developing multilevel disk drives and multilevel network interface units for use with only single level processors in building multilevel distributed systems.

This position may be debated, but the asked question is interesting. Until today, multilevel security seems to have been an ideal to approach and not a requirement to meet. The question is: is it possible to design a practical multilevel system offering full security? Based on which architecture?

The approach described in this paper reflects some results of a research project[1] which suggests some ideas to answer this question. This project aims at building a system architecture (machine and LAN) that offers a high degree of protection both for storage, processing and communication of user data. This protection is based on an exhaustive control of information flows, including timing flows, and ensuring there is no place for covert channels. We have chosen the distributed architecture of a secure LAN as an application framework.

2 Related works

Randell and Rushby described a secure distributed system in [15]. This system was designed to offer multilevel file system services over a network. The security was founded on an interpretation of the Bell and La Padula model [3] and enforced by use of cryptographic techniques to enforce separation between levels and to ensure confidentiality or integrity of files and file servers. The security was located in trusted network interface units (TNIUs), trusted terminal interface units (TTIUs) and a trusted multilevel station running a security kernel. This approach relied on securing application services over a network.

It was developped and extended through the DSS project ([17] and [18]) at DRA Malvern: the architecture of the Distributed Secure System is close to our own architecture. But the mechanisms chosen to enforce separation are not the same.

Other approaches tend to locate security controls inside various protocol layers and to protect connections established between entities over a network. The Trusted Network Architecture [10] is based on secure data channels over which only authorized subjects can send, inspect or modify the data stream. This is achieved through the use of mechanisms including encryption, checksums and nonces. Network communication that bypasses these secure data channel is not possible.

More commercial approaches [6] tried to devise secure networks. The Verdix secure LAN, for example, also founds the multilevel security it enforces on the Bell and La Padula model. A network security centre manages and controls all the operations and connections exerted by trusted network interface units. Protection and separation between connections is logical, and is based on the use of cryptographic techniques. The Boeing Secure LAN also consists of trusted interface units. The secure network server attaches labels to datagrams and provides mandatory access control decisions based on the value of the labels. The Sun MLS OS is an extension of SunOS to provide mandatory access control. It requires source hosts to label packets and destination hosts to check labels on received packets.

All these approaches were developed in order to protect confidentiality in systems. They are founded on the use of labels and of cryptographic methods to separate levels. But they do not prevent some illicit information flows. In particular, they are not involved in managing the allocation of resources among levels. And resource allocation or management is the reason for most of the covert channels in

1. This project was supported by DGA in France.

systems. In this case, cryptographic methods and labelling of packets are inefficient. Of course, if network lines are vulnerable, encryption can help to preserve the confidentiality and integrity of data transmitted by the network.

But if the system does not carefully manage the allocation of resources among levels, user communicating at low levels could detect and perceive the activity at higher levels. And encrypting messages does nothing to eliminate these covert channels. Achieving an efficient control of information flows, able to separate system domains in a quite secure manner, can eliminate them. Our goal is to devise such a "secure system".

3 Causal Dependencies and Security

3.1 Causal dependencies

A system may be described as a set of points *(o,t)*. A point references an object *o* at a time or date *t*. This introduction of time is necessary because time can be observed in the system, for example: durations of operations.

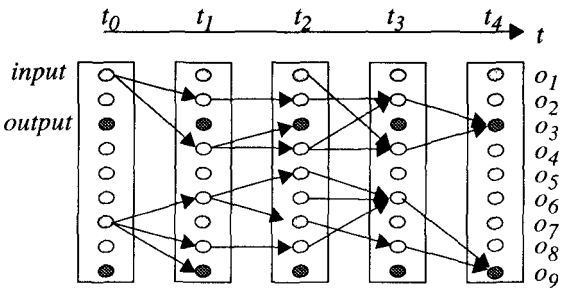

Fig. 1 Causal dependencies inside a system

So, one can act on the value of the object *o*, at the instant *t*, or one can act on the instant *t* at which the object *o* is given a particular value. In the first case, the object *o* can be used to transmit some information if any semantics can be assigned to its value and a storage channel is involved here; in the second case, time is used and therefore, a timing channel is involved if any semantics can be assigned to the observed instant values.

Some of these points are *input* points, others are *output* points, and the last ones are *internal* points. These points evolve with time and this evolving is due to the elementary transitions made by the system. An elementary transition can modify a point: then, at instant *t*, it sets a new value *v* for the object *o* of the point. This instant *t* and the new value *v* functionally depend on previous points.

This *functional dependency* on *previous* points is named *causal* dependency[1]. The causal dependency of *(o,t)* on *(o',t')* with $t' < t$ is denoted by $(o',t') \rightarrow (o,t)$. Informally, by *(o,t)* "causally depends on" *(o't')* we mean that the point *(o't')* is used to generate the point *(o,t)*. An interesting discussion of McLean in [11] illustrates the need to take account of these dependencies when addressing information flows. The

assumption made here may seem too strong with respect to this discussion. But it is useful because it permits to build sufficient conditions for security.

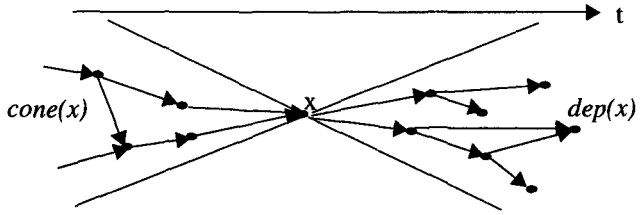

Fig. 2 Cones of causality and of dependencies

The transitive closure of the relation "$\rightarrow$" (denoted "$\rightarrow^*$") at (o,t) defines the *causality cone* of (o,t), in short:

$$cone(o,t) = \{(o',t') / (o',t') \rightarrow^* (o,t)\}.$$

Conversely, we denote the *dependency cone* the set $dep(o,t)$ of points which causally depend on (o,t):

$$dep(o,t) = \{(o',t') / (o,t) \rightarrow^* (o',t')\};$$

A relation between the sets *cone* and *dep* is given by:

$$y \in dep(x) \Leftrightarrow x \in cone(y)$$

where x and y denote two points (o,t) and $(o't')$ of the system.

These causal dependencies make up the structure of information flows inside the system. If a subject s has any knowledge about the internal functioning of the system, then he is able to know the internal scheme of causal dependencies. So, by observing any output point x_o, he is able to infer any information in $cone(x_o)$. In particular $cone(x_o)$ may include input points x_i which contain some input data of the system.

Conversely, by altering an input point x_i, s can alter any point in $dep(x_i)$ and in particular an output point $x_o \in dep(x_i)$.

In particular, if a subject s can observe a set O_s of output points x_o in the system, we denote by Obs_s the set of all points that s can observe in the system:

$$Obs_s = \bigcup_{x_o \in O_s} cone(x_o)$$

Similarly, if a subject s can alter a set A_s of input points x_i in the system, we denote by Alt_s the set of all points that s can alter and

$$Alt_s = \bigcup_{x_i \in A_s} dep(x_i)$$

3.2 Security

The aimed security must control both observation and alteration over the system. In a first part, we address only the observation problem, and its related

property of confidentiality. Informally, the system must ensure that causal dependencies enforce secure internal information flows.

Obs_s contains the points that a subject s in the system is able to observe in the system. The set R_s contains the points that the subject s has the right to observe in accordance with the security policy. So, we say in accordance with [5] that the system is secure if a subject s can observe the only objects he has the right to observe:

$$Obs_s \subseteq R_s \qquad (1)$$

When the security policy which is used to define the rights of subjects is the multilevel security policy, a classification level $l(x)$ is assigned to points x and a clearance level $l(s)$ is assigned to subjects s and the set $R(s)$ may be defined quite naturally by:

$$R(s) = \{x / l(x) \leq l(s)\}$$

3.3 Security conditions

Two conditions are *sufficient* in order to guarantee the security defined by *(1)*. Firstly, an interface rule expresses conditions on the classification level of an output point x_o and on the clearance level of the subject s who can observe this point:

$$\forall s, x_o \in O_s \Rightarrow l(s) \geq l(x_o) \qquad (2)$$

The second condition requires a monotonic increasing of levels over causal dependencies. If values of levels increase with sensitivity of points:

$$\forall x, \forall y, x \rightarrow y \Rightarrow l(x) \leq l(y) \qquad (3)$$

Cone-Lemma. If condition *(3)* is enforced then

$$x \in cone(y) \Rightarrow l(x) \leq l(y) \qquad (4)$$

Proof. We take the depth of the cone into account. We define $cone_n(y)$ the cone of y of depth n: so, $\forall x \in cone_n(y)$ there is a string of n points $x=x_n \rightarrow x_{n-1} \rightarrow ... \rightarrow x_1 \rightarrow y$. We want to prove that $\forall n\ x \in cone_n(y) \Rightarrow l(x) \leq l(y)$. This is done by induction on n. If $n=1$ then $x \rightarrow y$ and by *(3)* we have $l(x) \leq l(y)$.
For a depth of $n+1$, $x=x_{n+1} \rightarrow x_n \rightarrow x_{n-1} \rightarrow ... \rightarrow x_1 \rightarrow y$. And $l(x_n) \leq l(y)$ by induction assumption. Then $x \rightarrow x_n$ and by *(3)* $l(x) \leq l(x_n) \leq l(y)$. ■

Fact 1. If Conditions *(2)* and *(3)* are enforced in a system then the system is secure.

Proof. We must show that *(2)* + *(3)* $\Rightarrow$ *(1)* .
If $x \in Obs_s$ then $x \in O_s$ or $\exists y \in O_s / x \in cone(y)$. If $x \in O_s$, then by *(2)* , $l(x) \leq l(s)$.. In the other case $\exists y \in O_s / x \in cone(y)$, and then $l(x) \leq l(y) \leq l(s)$ by *(2)* and Cone-Lemma. So $x \in R_s$. ■

With respect to confidentiality, the both rules *(2)* and *(4)* ensure that for any subject s who has the right to observe an output point x_o, the observation of x_o will give to s only information he has the right to observe. So the definition of security given by *(1)* is satisfied.

The rule *(3)* defines *secure* dependencies. It gives the semantics of an internal control which can be exerted on each system transition when a relation of causal dependency is involved. This control on levels is sufficient to guarantee the security of

the whole system. It enforces the exhaustive control of information flows. This control of information flows (including its temporal aspects embedded in the definition of points) is achieved by making sure each transition and each elementary transfer of information from input points until system points which can be observed directly by a user. An other equivalent formulation of this last condition may be expressed as

$$\forall x, \forall y, l(x) > l(y) \Rightarrow \neg [x \rightarrow y]$$

When $x = (o,t')$ and $y=(o,t)$ and $t' < t$, this means that the level value of the object o may be downgraded, but both points before and after downgrading must not depend one on the other. In other words, this change on the value of the classification level of the object o may be done, but the value of the object o must also be erased, for example, in order (o,t) after downgrading does not causally depend on (o,t') before downgrading.

4 Interpretation

4.1 Choice of the hardware layer as context of interpretation

These rules must be instantiated in the framework of a real system to be used as a reference for building a secure system. This can be done by making an interpretation of the model in the context of one among the various abstract layers of a real system. The choice has been made to perform this interpretation in the most concrete system layer: the hardware layer. This approach offers two main advantages.

Firstly, the hardware layer manages only elementary objects whose granularity is the smallest in the system. So, by defining exhaustively all the observable objects inside this layer and by defining security controls on elementary operations that can be exerted on these objects, all the operations done on more abstract objects in the system will be submitted to these controls. Indeed, these more abstracts objects are built from elementary concrete objects and are accessed through combinations of elementary operations on these concrete objects. For example, assume that memory cells and disk blocks can be observed and that controls are enforced consequently on read and write operations. Then we can state that the use of files, in upper layers of the system, are constrained by these elementary controls: if write operations are prohibited on memory cells and blocks implementing a file f, neither f nor data structures associated to f (as file descriptor nodes) can be used by open or write operations on f to store any new information.

Secondly, this layer has few types of objects and subjects. So, the expression of controls is simplified and their enforcement may be done exhaustively. Let us detail this fact.

4.2 Security conditions in network interface units

Network interface units U connecting stations to a comunication medium constitute the system architecture. These units access the medium according to the CSMA/CD Medium Access Control protocol, as defined by IEEE 802.3. We denote by M the *Medium* managed by the Physical Layer. In particular, this layer offers two

elementary signals (*Carrier Sense, Collision Detection*) and *B* which contains the bit value carried by M.

The active entities, which are the *subjects* inside this hardware layer, are only the network interface units *U*. These units have one input *delay* value , that is chosen externally as a uniformly distributed random value in a finite range. They can be represented by two data cells: the bit value *b* it has to deposit on *M* or it has sampled from *M* and *d* that contains a delay value to spend before transmitting.

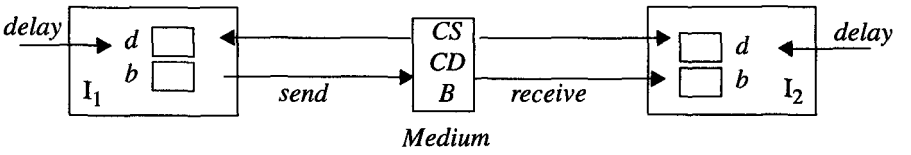

Fig. 3 The system architecture

In the same way, the *objects* are the internal cells *b* and *delay* of *U* and the communication medium *M* (including *CS,CD* and *B*). A level is assigned to all objects and subjects. The cells *b* and *delay* in *U* are doted with the level *l(U)* and all objects in *M* share the same level *l(M)*. The elementary transitions include the elementary *send* and *receive* operations made by *U* between its own cells and *M*.

The *receive* operation, as expressed in the CSMA/CD protocol, consists in permanently listening to signals *CS* and to the bit value *B* carried by *M*. This operation produces a new value for *b* and the following dependencies are involved:

$$\{CS,B\} \rightarrow b$$

Condition-receive. In this case, the rule *(3)* applied to the *receive* operation produces:

$$l(M) \leq l(U)$$

The *send* operation is less simple. Firstly, the decision by U to deposit a bit value upon *M* is taken by listening to *M* and watching at signals *CS* and *CD*. The transmission of the *b* value may be delayed according to the delay value stored in *d* when *CD* indicates that a collision occured. When transmitting the bit *b*, a new value is assigned to the *M* components. So

$$\{CS,CD\} \cup \{delay\} \rightarrow \{d\}$$
$$\{b,d\} \rightarrow \{CS,CD,B\}$$

Condition-send. The rule *(3)* applied to these dependencies produces

$$l(M) \leq l(U) \leq l(M) \Rightarrow l(M) = l(U)$$

4.3 Management of level objects

Levels are themselves objects in the system. So they are also submitted to the control of dependencies. A classification level is assigned to them: we have chosen to give the value "*Low*" to the level of a level object.Then, the fact that an information is secret is not itself a secret. That is not a doctrine, but only a work assumption that we made in order to simplify.

Being submitted to the control of dependencies, the rule *(3)* must be applied to levels and then, given a level l_i:

$$x \rightarrow l_i \Rightarrow l(x) \leq l(l_i) \Rightarrow l(x) = Low$$

In other words, the value of a level and the instant at which this level gets a given value only depend on low level information. This condition is sometimes difficult to enforce, for example, when the value of a level decreases from a *high* to a *low* value. This change of the level value must have been planned and declared at low level.

In our system architecture, the value of the level of M , and time at which this level takes a given value must be generated from *Low* level points. Then, the value of the level of the medium and the time spent to this level are stated at *Low* level. Therefore, the use of M is time sliced between levels. And slices are declared or computed at *Low* level. A *High* process never acts on the value of a level (by maintainig it or by changing it).

Similarly, the level of U must be declared at low level. And the time spent by U at this given level is also declared in advance at *Low* level. So at the beginning, U is at *Low* level. If a user wants to use the host and U at a level *High*, this user (and not a process running on the untrusted host) must firstly declare at *Low* level (not *High*) that he requires to use the unit U at level *High* during time t, in order to achieve communications at level *High*. This can be viewed as a constraint for the user. In fact, it is no more inconvenient than doing a login procedure. Of course, it is sometimes difficult to estimate exactly the amount of time that he will need. But experiments on the architecture that was developped on these principles show that light overestimations do not degrade performances tragically [16].

4.4 Security SubSystem: S^3

Because they are simple, the controls expressed in *Condition-receive* an *Condition-send* can be enforced in U by a subset of hardware features which are driven by a subset of software. These two subsets constitute the *Security SubSystem* or S^3 of the system. This S^3, so called by ITSEC [8] in Europe, is in fact the TCB, as formalized in the Orange Book [12] and later the Red Book [13], of the interface unit U and acts as a reference monitor.

Fact 2. If Condition-receive and Condition-send are enforced in U by S^3 then the system is secure.

Proof. The system is secure if condition *(4)* is always satisfied. The points that a user (or a process) can observe in an interface U_i are b_i and d_i. From 4.2 and 4.3 we can state that

$$cone\ (b_i,d_i) \supseteq \{CS,CD,B\} \cup delay_i \cup l(U_i) \cup l(M)\ and$$
$$cone\ (CS,CD,B) \supseteq l(M) \cup [\{b_j,d_j,delay_j\} \cup l(U_j)]\ \forall j,\ U_j\ sending$$

Then,

$$cone\ (b_i,d_i) = \{CS,CD,B\} \cup delay_i \cup \{l(U_i),\ l(M)\} \cup [\{b_j,d_j,delay_j\} \cup l(U_j)]\ \forall j,\ U_j\ sending$$

And

$$l(CS) = l(CD) = l(B) = l(M);$$
$$l(delay_i) = l(b_i) = l(d_i) = l(U_i);$$
$$l(b_j) = l(d_j) = l(delay_j) = l(U_j) = l(M) \text{ because } U_j \text{ is sending}$$
$$l(l(M)) = l(l(U_i) = l(l(U_j)) = Low$$
$$l(M) \leq l(Ui) \text{ from condition-receive and condition-send}$$

So,

$$x \in cone(b_i,d_i) \Rightarrow l(x) \leq l(Ui) \blacksquare$$

The S^3 functionning ensures that values of points observed in U_i and time t at which these points take these values depend only on information that are allowed to be observed. Some modulations on values or durations of elementary *send* and *receive* operations can be observed in U_i: these modulations may be created in order to generate information flows, but these flows are inefficient and do not strike a blow at the security, thanks to the controls done by local S^3.

5 Implementation of a Distributed S^3 over a LAN

5.1 Security conditions enforced in a local S^3

The *local* S^3 is in charge of enforcing the controls defined by the two send and receive conditions and regulating the access of these interface units to the communication medium according to these. This local S^3 keeps values of levels for the interface unit and for M. It grants or denies to the interface unit the access right to M according to those values of levels.

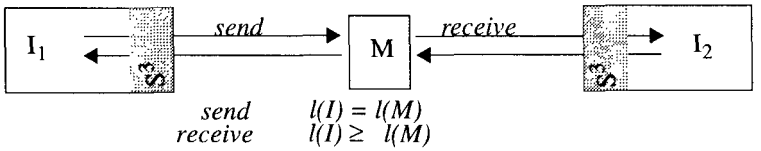

Fig. 4 Rules to access the medium in a network interface unit

In fact, it can intervene by hardware on elementary operations exerted in order to deposit or sample information on M. So, for the interface unit, the ability to send or receive at any instant t is given by its own level and the level of the medium. An interface unit equipped with its local S^3 constitutes a Trusted Network Interface Unit or TNIU.

5.2 Trusted paths to local S^3

There is a need for building a trusted path between users and local S^3 of the network interface unit. The mechanism of a Secure Interface Device (SID) is used and permits to implement the principle of reservation of resources in advance. It is shown in Fig. 5 . A quite simple dialogue between users and local S^3 permits:

- to declare the value of the current level of the connected host for the next session and the required duration for this session; this fixes the level of the interface unit and the time needed for exchanges at this level;

- to initialize then the local S^3 functioning in accordance with these declarations.

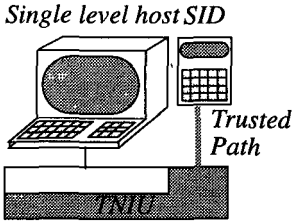

Fig. 5 Trusted Path

This is insufficient. Indeed, the local S^3 must be able to require a level for the medium in accordance with the reservations made by the user, and to know its current level value. A real security subnetwork is needed.

5.3 Security Subnetwork

Exchanges between hosts running at various current levels may occur only if the level of the medium can change. In fact, this value is time sliced in accordance with rules defined in 4.3: this slicing is based on level reservations which are produced and emitted at low level by user through the trusted path.

Then, two conditions must be satisfied. Firstly, the value assigned to the level of the medium must be known by every local S^3. Secondly, the time slicing of this value must be enforced in a synchronous way over the LAN.

Satisfying the first condition requires a communication subnetwork between all the local S^3. In fact, in this case, this subnetwork uses the same medium of communication as hosts. The local S^3 which are interconnected by this way constitute the *security subnetwork* of the system. This security subnetwork is used to exchange security data between local S^3.

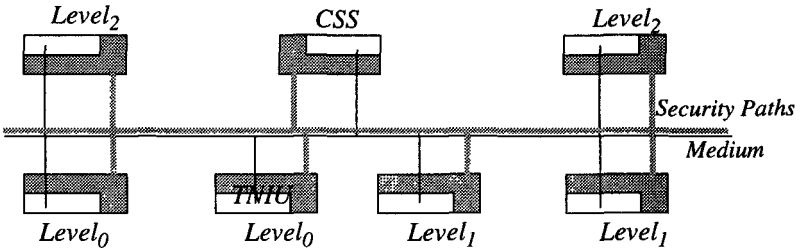

Fig. 6 Security Subnetwork

A *centralized security station* (or *CSS*) manages the data of security for the network. In particular, it manages levels which are assigned to interface units and to the communication medium according to reservations made by hosts and users through the SID and emitted to CSS by local S^3. It broadcasts also these data to all the local S^3 over the security subnetwork.

Satisfying the second condition requires the existence of a protocol in charge of regulating the exchanges of security data. It is also in charge of ensuring that the time slicing of the medium level is known by all the local S^3 in a synchronous manner. So, the rules which are used to access the medium in order to exchange security data are not the same as the rules used by hosts in order to exchange user data. These rules constitute the *Security Medium Access Control (SMAC)* protocol.

6 SMAC protocol and multilevel LAN

6.1 The SMAC Protocol

It enforces time slicing for the level of the medium according to reservations made to the CSS. It manages also the exchange of security data under the authority of the CSS. These data include particularly reservation data emitted from local S^3 and level settings for the medium which are emitted from the CSS. In few words, the SMAC protocol is reservation based.

It manages two functioning modes for the interface unit: a user mode and a security mode. In the security mode, only local S^3 can use the medium M to exchange security data with CSS. In user mode, operations to send and receive user data can be performed by the interface units according to values of their own level and of the level of M. The CSS computes time slices for sessions of exchanges in user mode which correspond to various values assigned to the level of M. These values are set in accordance with reservations previously received. At the end of a slice, the interface unit always returns to the security mode. In security mode, the CSS may ask to local S^3 if reservations are pending. If yes, local S^3 may answer by giving the content of their pending reservations. The protocol for this dialogue is a synchronous one. The CSS fixes a transmission slot for each local S^3 to answer and each local S^3 may answer during its reserved slot. The CSS broadcasts then a new value for the level of the medium and a new session in user mode is started. In user mode, a Medium Access Control (MAC) protocol arbitrates the access to the medium between units which are allowed to access it: this protocol is CSMA/CD is in our case.

The SMAC protocol is similar to protocols used in the real time world where requirements on the amount of delay between the time a packet is ready and the time it is received at destination are stringent. In these protocols, some sources must reserve transmission slots before they can begin transmission [19].

6.2 Architecture of the interface unit

So, this protocol leads to a quite simple architecture for secure interface units. Two components make up them.

The first component is a classical one which enforces a standard MAC protocol. In our case, this protocol is CSMA/CD. This component achieves the *send* and *receive* requests issued by upper communication layers in hosts. These two operations are achieved by activating the *Rec* and $Send_1$ modules in accordance with the CSAM/CD protocol.

The second component enforces the operations of the local S^3. Four functions are needed: they are achieved by activation of four modules. The *Rec* module is similar to the *Rec* module of the CSMA/CD component: it listens to the medium and recovers frames from it. The $Send_0$ module enforces sending operation for the local S^3. But these operations are done in a synchronous way by getting transmission slots computed by the CSS. So this module is quite much simpler than the $Send_1$ asynchronous module of the CSMA/CD component. Then the *Int* interpretation module achieves the interpretation of security commands emitted by the CSS (*set_level_medium*, *set_level_niu*...) or by the user through the Trusted Path and its SID (*reserv_level*). The last *Inhib* module drives the physical connection of the CSMA/CD component to the medium. It inhibits *Rec* or $Send_1$ accesses to the medium according to values of the medium level and of the interface level.

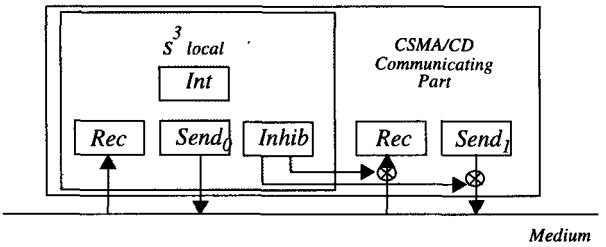

Fig. 7 SMAC interface unit

These are all elementary modules, whose functions are simple and not complex. So the local S^3 is in fact a much lighter component than the CSMA/CD component. It can be connected with an existing standard CSMA/CD component.

6.3 DS^3 and multilevel LAN

The CSS, the local S^3 and the medium which is accessed in accordance with the rules of the SMAC protocol constitute the *Distributed* S^3 of the LAN (or DS^3). The DS^3 and the local S^3 cooperate in enforcing the control of information flows in the more concrete layer of the system: the hardware layer. In particular, this control is enforced by programming the local S^3. This programming is done in security mode by exchanging security frames between the trusted CSS and local S^3. So, these exchanges are isolated from the behaviouring of the untrusted interface components

A multilevel station, built above the same principles (more details in[4]) may be added to ensure a secure sharing of data between levels. Because such a station is able to manage multilevel data structures and processes, it permits to monolevel stations to access data through levels in a quite secure manner.

The global architecture of such a system constitutes a secure LAN which is said to have a multilevel functioning mode. Such an architecture satisfies the required security property: all information flows, including timing flows, are controlled exhaustively.

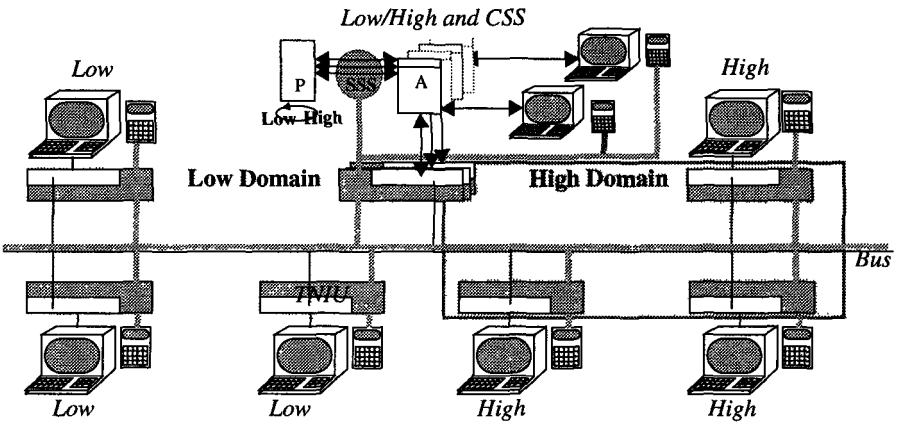

Fig. 8 Multilevel LAN architecture with two levels

It is obvious that this architecture is insufficient if the communication medium is vulnerable: that is not the addressed problem in this paper. Cryptographic techniques may be added to preserve the confidentiality and integrity of messages transmitted over the network. These techniques may rely on cryptographic devices and functions which can be driven by the Distributed S^3 (local S^3 and CSS). They can be viewed as an external protection layer, by opposite to the internal protection layer described here.

7 Discussion

Such an architecture enforces the rules of multilevel security. The DS^3 aim at controlling internal information flows which are involved when communications are achieved over the medium by ensuring that the involved causal dependencies are secure. This control of information flows may be used in order to enforce confidentiality, integrity and availability properties.

7.1 With respect to confidentiality

Let x_{low} and y_{high} two points that belong to two different domains D_{low} and D_{high} in the system. These domains may be defined, when multilevel security is the applied security policy, by $D_l = \{x / l_c(x) = l\}$ with $l(x)$ the confidentiality level of x and $D_{high} \cap D_{low} = \varnothing$. The cone-lemma ensures that:

$$cone(x_{low}) \subseteq D_{low} \quad (5)$$

$$cone(y_{high}) \subseteq D_{low} \cup D_{high} \quad (6)$$

These conditions ensure that the observation of any point in D_{low} will reveal no information about points in D_{high}. But points of D_{high} may be built from points of D_{low}. The only allowed flows of information are from *low* to *high*. It is a classical result in confidentiality. In this case, all information flows are controlled.

By managing levels of interface units and of the medium, the SMAC protocol permits the local S^3 to authorize or not any send or receive accesses of interface units

to the medium. The enforced rules authorize an interface unit to send data to the medium when both levels of the interface and of the medium are the same. In this case, the state of the medium at a given time depends on the previous state and operations of interface units at the same level only. This fact implements the condition (5) and in particular, at *Low* level, the state of the medium depend only on points at the same *Low* level.

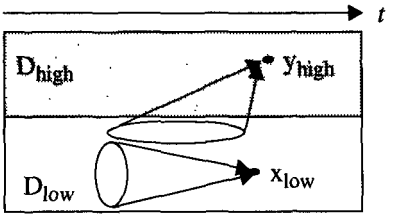

Fig. 9 *Graphical translation of confidentiality properties*

Conversely, the enforced rules authorize an interface unit to receive data from the medium when the level of the interface dominates (*high*) the level of the medium (*low*). So a point y_{high} in the interface may depend on the state of the medium and it was showed that this state of the medium only depends on points at the same level. This state is a point x_{low} in D_{low} and $x_{low} \in cone(y_{high})$. This fact is in accordance with the condition *(6)*.

The definition of points includes objects and timing components. The conditions *(5)* and *(6)* with respect to timing components are ensured by the time slicing enforced by the SMAC protocol on the level of the medium. Slices are computed on a *Low* level information basis, from reservations. So, for example, durations assigned to send or receive operations exerted by an interface unit depend on: firstly, durations of time slices which are assigned to each level of the medium, and secondly, on the state of the medium. We have showed that both depend on points whose level is dominated by the level of the interface.

7.2 With respect to integrity

When integrity is the addressed property, the same approach can be used. The set Alt_s contains the points that a user, or more generally a subject s is able to alter and the set R_s contains the points the subject s has the right to alter in accordance with the security policy. So, the definition of the security given in *(1)* is the same here: the system is secure if a subject s can act only on the objects he has the right to act:

$$A_s \subseteq R_s \qquad (7)$$

The set R_s is also given by

$$R_s = \{x / l_i(x) \leq l_i(s)\}$$

where $l_i(s)$ denotes the clearance of the user in integrity and $l_i(x)$ is the integrity level of point x. The interface rule expresses conditions on the integrity level of an input point x_i and the clearance level of the subject s who can alter this point:

$$\forall s, x_i \in A_s \Rightarrow l_i(s) \geq l_i(x_i) \qquad (8)$$

The second condition requires monotonic decreasing of levels over causal dependencies. If values of levels increase with integrity of points:

$$\forall x, \forall y, x \rightarrow y \Rightarrow l_i(x) \geq l_i(y) \qquad (9)$$

Dep-Lemma. If condition *(3)* is enforced

$$x \in dep(y) \Rightarrow l_i(y) \geq l_i(x)$$

Proof. The proof is trivially similar to the proof of cone-lemma. ■

Fact 3. If in a system Conditions *(2)* and *(3)* are enforced then the system is secure.

Proof. We must show that *(2)* and *(2)* $\Rightarrow$ *(7)* . If $x \in Alt_s$ then $x \in A_s$ or $\exists y \in A_s / x \in dep(y)$. If $x \in A_s$, then by *(2)* , $l_i(x) \leq l_i(s)$.. In the other case $\exists y \in A_s / x \in dep(y)$, and then $l_i(x) \leq l_i(y) \leq l_i(s)$ by *(2)* and Dep-Lemma. So x is in R_s. ■

With respect to integrity, the both rules *(2)* and *(3)* ensure that any subject s who has the right to alter an input point x_i is allowed to alter any point of $dep(x_i)$. So the alteration of x_i by s will have an impact only on points that s has the right to alter.

A convention on levels can be chosen: a level l is a pair (l_c,l_i) where l_c denotes a confidentiality level and l_i is an integrity level and a comparison rule on levels may be:

$$(l_1 \leq l_2) \Leftrightarrow (l_{c1},l_{i1}) \leq (l_{c2},l_{i2}) \Leftrightarrow (l_{c1} \leq l_{c2}) \wedge (l_{i1} \geq l_{i2})$$

With such a convention, the confidentiality conditions *(3)* can be extended with condition *(3)* in a simple way by using a level l for integrity and confidentiality:

$$\forall x, \forall y, x \rightarrow y \Rightarrow l(x) \leq l(y) \qquad (10)$$

Then the send and receive conditions can be expressed in the same way for both confidentiality and integrity.

Let x_{low} and y_{high} two points that belong to two different domains of integrity D_{low} and D_{high} in the system. These domains may be defined by $D_l = \{x / l_i(x) = l\}$ if $l_i(x)$ is the integrity level of x and $D_{high} \cap D_{low} = \varnothing$. So D_{low} denotes a domain of low integrity and D_{high} denotes a domain of high integrity. Referring to the rule *(3)* , then $dep(x_{low})$ and $dep(y_{high})$ are sets of points in the system and the rules ensure that

$$dep(x_{low}) \subseteq D_{low} \qquad (11)$$

$$dep(y_{high}) \subseteq D_{low} \cup D_{high} \qquad (12)$$

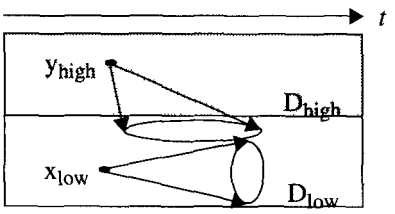

Fig. 10 Graphical translation of integrity properties

The condition *(11)* express that the alteration of any point in D_{low} will alter no information about points in D_{high}. So the only allowed flows of information are from *high* to *low* (condition *(12)*). This is in accordance with classical results as expressed by Biba [2] for example.

This is achieved in the context of the multilevel LAN by enforcing the same mechanisms of control as for confidentiality. Each communicating host and interface unit belongs to an integrity domain, and every elementary transfer of information is submitted to this control of information flows.

By defining integrity domains and by controlling flows between these domains according to the previous rules, we ensure there is no way, at a low integrity domain, to use any input covert channel in order to insert corrupted instructions or data in a high integrity domain.

These results may be applied to isolate and minimize functions which are vital to run a critical process inside. Criticality levels may be defined; they reflect the degree of criticality of functions or data with respect to the system objective. So a *High* critical domain is fully protected from eventually malicious operations exerted from a *Low* critical domain. This scheme is interesting in a security point of view, but also for cost considerations. Indeed, it permits to minimize the *High* critical domain by including in it the only really critical functions and data. Techniques used during the development of such a system and during its running in order to ensure dependability properties may be reduced by limiting them to the only critical domain.

7.3 With respect to availability

A particular case of the integrity property which was previously described offers some kind of availability. Indeed, the SMAC protocol which is used to share the communication medium of the multilevel LAN tends to separate domains of integrity/ criticality and to regulate flows between these domains according to multilevel rules.

In particular, the time slicing exerted on the level of the medium coupled with the ability assigned to the interface units of sending or receiving according to time slices make impossible for an interface at a *Low* integrity level to disrupt the use of the communication medium by interfaces at a *High* level of integrity. When communications occur at a given level, there is no way for interface units at an other level to get any send access to the medium.

So the availability of services inside the domain of *High* integrity can not be countered by malicious processes at a *Low* level of integrity or by a crash or a bad functioning occurring on an host at a lower level of integrity.

So, some mechanisms may be employed to ensure high availability inside the high integrity domain itself. But their use is limited inside this domain only, and the availability property is not put in danger by lower integrity levels, thanks to the separation enforced by the DS^3 and the SMAC protocol.

8 Conclusion

Techniques and mechanisms suggested here were firstly designed and developed in order to protect the confidentiality of data, processes and communications over a LAN. This protection is based on a control of dependencies that enforces an exhaustive control of information flows. It relies upon a distributed security subsystem composed of a particularly restricted subset of hardware

mechanisms: they are in charge of ensuring that accesses of interface units to the medium are done in accordance with multilevel rules. This leads to share the medium in a particular way which defines a secure medium access control (or SMAC) protocol. This protocol may be viewed as an extension of an existing MAC protocol, as CSMA/CD.

This logical separation, achieved by means of this protocol, may be also used in order to separate integrity levels. In particular, the extremely strong control of information flows which is enforced can isolate some domain where a high level of integrity may be needed drastically. This domain is then protected from other domains of low integrity that can not corrupt its behaviour: in particular they can not enforce any communication channel to send malicious data or pieces of code. Such levels of integrity can be used in critical applications to protect some vital functions. As a particular case of the application of control of dependencies to integrity, some needs in availability may be answered also. The separation between high integrity and low integrity domains ensure that any (malicious or not) failure in a low integrity domain will not disrupt the good functioning inside a high integrity domain.

This whole security protects efficiently all the information that needs to be protected, and only this information. We feel that this approach is well adapted to the real world, where in fact, few informations and functions necessitate to be protected. So, such a system does not penalize the use and processing of most of the data which belong to an unprotected domain. Rather, it makes lighter the amount of protected processing by reserving it to the only data which necessitate it.

A real system is actually under development upon these principles. Some mechanisms and functions of distributed operating systems are beeing built above this basic architecture. They implement classical distributed operating services, but, taking account of the underlying architecture and of its multilevel functioning, they implement also new multilevel distributed operating services: sharing files between hosts running at different levels, or accessing remote files, running processes on remote hosts. Then the challenge is no longer building a secure distributed operating system but to building some distributed operating services upon a secure architecture, and taking advantage of its security features.

9 References

1. P. Bieber, F. Cuppens, "A logical view of secure dependencies." In *Journal of Computer Security*, Vol. 1, Nr. 1, IOS Press, 1992

2. K. J. Biba, "Integrity Considerations for Secure Computer Systems", Technical Report ESD-TR-76-372, ESD/AFSC, Hanscom AFB, Bedford, Mass., 1977. Also MITRE MTR-3153.

3. D. E. Bell and L. J. Padula "Secure Computer Systems: Unified Exposition and Multics Interpretation", MTR-2997, MITRE Corporation, Bedford, Mass. (1975).

4. B. d'Ausbourg and J.H. Llareus, "M^2S: A machine for multilevel security", *in Proceedings of ESORICS92*, Toulouse, France, 1992

5. G.Eizenberg, "Mandatory policy: secure system model". In AFCET,editor, *European Workshop on Computer Security*, Paris,1989.

6. G.King "A survey of commercially available secure LAN product" , *in Proc. Int. IEEE Conf. on Computer Security Applications*, Tucson, Arizona, December 1989

7. ISO 7498-2, Organization for Standardization, Information Processing Systems - Open System Interconnection Reference Model - Security Architecture, 1988

8. Information Technology Security Evaluation Criteria, Harmonized Criteria of France, Germany, the Netherlaands, and the United Kingdom, 1990

9. H.L Johnson et al. "Integrity and Assurance of service Protection in a large, multipurpose, critical System" , *In Proceedings of the 15th National Computer Security Conference, Baltimore,* MD, October 1992

10. E.S. Lee, B. Thomson, Peter I.P. Boulton and M. Stumm "An architecture for a Trusted Network" *European Symposium on Research in Computer Security, ESORICS90*, Toulouse, France, 1990

11. J. McLean, "Security Models and Information Flow" ,*IEEE Symposium on Security and Privacy, Oakland,* 1990.

12. NCSC. Department of Defense. Trusted Computer Systems Evaluation Criteria. Technical report DoD 5200.28-STD, National Computer Security Center, Fort Meade, MD, December 1985

13. National Computer Security Center Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, NCSC-TG-005, July 1987

14. N. E. Proctor and P. G. Neumann "Architectural implications of covert channels", *In Proceedings of the 15th National Computer Security Conference, Baltimore,* MD, October 1992

15. J.M. Rushby and B. Randell, "A Distributed Secure System" Computer vol 16 no 7, IEEE, July 1983

16. P.Siron and B.d'Ausbourg "A Secure Medium Access Control Protocol: Security versus Performances" *in Proceedings of ESORICS 94*, Brighton, UK, November 1994.

17. J. Wood and D.H. Barnes "A Practical Distributed System" *in Proceedings of the International Conference on System Security, London,* October 1985

18. J. Wood "A practical Distributed System" in *Proceedings of the Second International Conference on Secure Communication Syst*ems, IEE, London, October 1986

19. R. Yavatkar, P. Pai and R. Finkel "A reservation based CSMA Protocol for Integrated Manufacturing networks", *Tecn. Rep. 216-92, Department of Comp. Sc., Univeristy of Kentucky,* Lexington, KY