Sifting UNICOS MLS Features

Francis Belot and *Francois Gélineau,* Commissariat à l'Énergie Atomique (CEA), Centre d'Etudes de Limeil-Valenton (CEL-V), Villeneuve St. Georges, France

Abstract

Dealing with sensitive information requires a trusted security policy. Many security requirements can be satisfied by using separate networks and machines for the different security domains; however this leads to duplicate resources.
Two years ago, we began to study the possibility of keeping, on a single CRAY, objects and subjects of different security domains in a secure way.
In this paper, we describe how MLS features could be used for that purpose. We discuss the issues encountered when building such a trusted environment, the expected benefits and the impact on operations and users.

Introduction

Among the 12 CEA research centers, CEL-V is dedicated to numerical simulations of physical phenomena. Most of the computational power is provided by CRAY machines either PVP (Y-MP 8/128 and Y-MP 4I/128) or MPP (T3D 128 PEs). Hundreds of physicists are using these machines, running codes dealing with sensitive data of different security levels.
Although the CEL-V local area network is fully disconnected from public networks, internal security rules (clearance ...) have always been necessary to manage users and sensitive data. So, the common security principle of *need to know* was early applied to enforce security controls : every physicist is given access only to information necessary to complete his job.
But the lack of trusted systems and networks, able to process simultaneously a range of sensitive or classified information, has led us, for several years, to separate resources and users on different physical networks and machines inside the research center. Of course, this entails many drawbacks :

- it leads to duplicate resources
- it prevents sharing computational power in case of unbalanced activity
- it increases management and maintenance costs

Moreover, CEA has always been interested in security software enhancements. For example, CEL-V was one of the first sites installing a UNICOS secure system in 1988.
Later, we continually took interest in MLS experiences reported by other CRAY sites [6].
More recently, in 1993, CEL-V computer center started to examine the security mechanisms provided by UNICOS MLS.
We intended to handle objects and subjects of different security domains on a single machine without generating too many changes in users' environment.
However, we did not install MLS features on our production computers, at that time, for the following reasons :

- lack of features in the earliest MLS versions
- UNICOS 8.0 enhancements expected
- avoiding any incidents involving production machines
- problems encountered by other CRAY sites

Consequently, important feasibility studies were carried out, in partnership with Cray Research, France.
A test platform (CRAY X-MP28) was dedicated to security experiments.
This paper in mainly based on reports done at CEL-V about UNICOS MLS as part of the management of a CRAY system between several sets of users and customizing specific needs.

The first section gives an overview of MLS mechanisms available since UNICOS 8.0.2. The following sections present the test platform configuration, the customization of some system services and issues encountered when using MLS features.

1 MLS security mechanisms

This section could be skipped by readers familiar with security concepts and MLS mechanisms.

1.1 Security objectives

The UNICOS MLS implementation is based upon security criteria defined in the Trusted Computer System Evaluation Criteria (TCSEC).

In the TCSEC terminology [5], security objectives are classified in three main items :

- security policy
  It is a set of rules and practices by which a system regulates the processing of sensitive information
- accountability
  It represents all the controls included in any mechanism processing or trying to process information
- assurance
  The system must contain hardware and software mechanisms that can be independently evaluated to provide sufficient assurance that the system enforces the security policy and accountability objectives.

For UNICOS MLS, these objectives are enforced by mechanisms as shown in figure 1 [2].

```
            Security policy

. Discretionnary access control (DAC) mechanism
        - file permission bits
        - access control lists (ACLs)
. Mandatory access control (MAC) mechanism
        - security labels for subjects and objects
        - network access list (NAL)
        - multilevel directories (mld)
. Object reuse

            Accountability

. Identification and authentication
        - login/passwd sequence
. Auditing

            Assurance

. Privilege mechanism
        - Privilege assignment list (PAL) mechanism
. Trusted facility management (TFMgmt)
        - separation of traditional root capabilities
        - definition of administrative roles and tasks
```

Figure 1: UNICOS MLS mechanisms

1.2 Trusted UNICOS

The Trusted UNICOS system is a specific configuration of UNICOS MLS designed to meet all the B1 criteria of the TCSEC and of the Trusted Network Interpretation (TNI) for the network side.
Indeed, the Trusted UNICOS system has been evaluated as a network component to ensure that the system can maintain the B1 rating when integrated into a heterogeneous network trusted computing base (NTCB).

As they have not been evaluated as trusted software, some of the UNICOS functionalities are not included in "Trusted UNICOS" and should not be used in the operating system configuration. Among all the non evaluated system software, we can find :

- services : `gated(8)`, `snmpd(8)`, `tftpd(8)` ...
- products : USCP, FTA, DNS, NIS, Kerberos ...

1.3 Trusted Facility Management (TFMgmt)

The UNICOS 8.0 MLS system supports several versions of TFMgmt :

- the PAL-based privilege mechanism (PAL_only)
- a super-user mechanism (PRIV_SU : traditional UNIX root mechanism)
- the old category-based TFMgmt used in UNICOS 7.0 (PRIV_TFM)

The two last mechanisms do not meet TCSEC requirements. Only the PAL-based privilege mechanism can be used on Trusted UNICOS systems.
This mechanism enforces the principle of *least privilege* : a process must be granted the most restrictive set of privileges only as long as needed to perform its tasks.

In the following subsections, we will give an overview of MLS important features.
For more details about all MLS security mechanisms see [1, 2].

1.4 Security labels

Each object or subject in the system is assigned a *security label*. It is made of a security level and compartments, and is used during mandatory access control[1](MAC) :

- an object can be accessed or executed only if the security label of the subject dominates the security label of the object.
- an object can only be modified by a subject of the same security label.

Two special security labels, *syslow* and *syshigh*, are used to protect the system files from unauthorized use or modification.
The *syslow* security label ensures that only trusted processes can modify some system files. The *syshigh* label ensures that only trusted processes can access and/or modify some other system files.

1.5 Categories, privileges and PALs

A *category* is a nonhierarchical authorization, either effective or permitted, defined to identify an administrative role : `secadm` (security administrator), `sysadm` (system administrator), `sysops` (system operator), ...
Permitted categories are usually granted to an administrator at login time according to the `valcat` UDB field for this user.

[1] A privileged process can override this control.

A permitted category can be turned to effective at any moment.

A *privilege*, associated with a process, represents some special abilities available during the execution of the process. A privilege can be either permitted or effective and is checked at kernel level.
A process can also be assigned a *privilege text*, tested at user level to know if the program is executed by an administrator. The privilege text allows the process to manage private security processing.

A *privilege assignment list* (PAL) is an attribute given to an executable file which defines privileges that are assigned to the associated process during its execution.
This list specifies for each active category a set of privileges and sometimes a privilege text to be given to the process.

1.6 Multilevel directories (mlds)

A *multilevel directory* provides a method for sharing a common directory name while partitioning the directory contents according to security labels.

When creating a mld, a directory tree structure is built : a subdirectory is created for each security label, named by a kernel-generated representation of the label.
A new type of symbolic link called multilevel symbolic link is used : in a path name, this kind of link is automatically expanded to the mld's subdirectory corresponding to the security label of the process using the pathname. So, users with different security labels are transparently redirected to the appropriate subdirectory.

2 MLS configuration

This section gives some details about the system configuration of the test platform.

2.1 Objectives

Our goal is to build a system able to separate logically two[2]sets of users on a single machine. We intend to connect the platform to two physically separated networks representing two different security domains.

2.2 Security labels

We assume that :

- two security levels : `admin` and `users` are defined for administrators and end users
- users can connect the CRAY host through two networks named `A` and `B`
- a dedicated network is available for administrators

Figure 2 shows the compartments and levels defined for the test platform.
Each user is assigned a security label at login time: either

```
                                    MLS Compartments

      Name     Mask
      ------   ----
E->   A        0040
      B        0400


                                       MLS Levels

      Name       Number
      -------    ------
E->   admin      0
      users      1
```

Figure 2: Compartments and levels options

`users,A` or `users,B` depending on the label defined in the UDB and the label of the remote host used to access the CRAY system, as shown in figure 3.

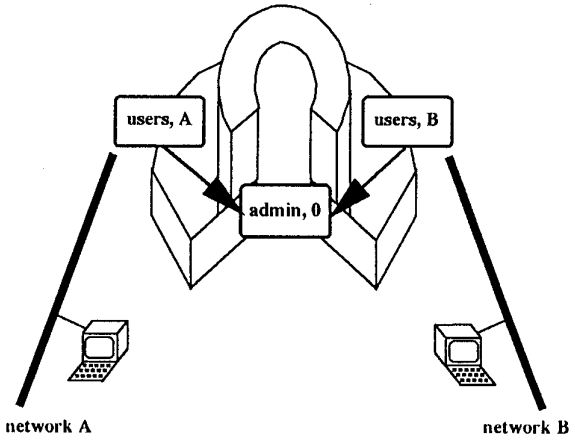

Figure 3: Security labels

Users at label `users,A` and users at label `users,B` cannot communicate with each other but all of them can execute the system commands at label `admin,0`.
Obviously, non administrative users are not allowed to change their labels. Depending on the `comparts` field in the UDB, a user can be restricted to connect from a specific network or allowed to connect from any of the two networks. However, in this last case, there will be two different environments[3], and it will be impossible to transfer information between `A` and `B`.

2.3 MLS options

All the security options were chosen as close as possible to "Trusted UNICOS" and by the way, conformed with the TCSEC requirements.

[2] or more

[3] using a mld

2.3.1 Kernel options

Figure 4 shows the kernel security mechanisms options.
The following remarks give some reasons for setting these

```
          MLS System Options

S-> Secstat system call restricted              ON
    Minimum security level                      0
    Maximum security level                      16
    Valid system compartment mask (octal)       077777777777777777777
    /tmp and /usr/tmp minimum security level    0
    /tmp and /usr/tmp maximum security level    1
    /tmp and /usr/tmp compartment mask (octal)  0440
    Secure system console - administration
    Default system console - administration     /dev/console
    Secure operator console
    Enforce strict device labeling rules?       ON
    Enforce secure mkdir behavior?              SECURE
    Enforce secure kernel information display?  ON
    Enforce socket usage for syslogd?           ON
    Enforce system high/low security labels?    OFF
    Super-user privilege policy?                OFF
    UNICOS 7.0 version of TFMgmt?               OFF
```

Figure 4: Kernel options

options :

- Enforce secure kernel information display
  When set to "ON", this selection enables enforcement of security policy for commands such as: `jstat(1)`, `ps(1)`, ... These commands use privileges to access `/dev/mem` or `/dev/kmem`. The enforcement of security policy means the information displayed by these commands should be dominated (levels and compartments) by the invoking process.
  For example, while executing the `ps` command, a user at label `users,A` will not see processes belonging to users at label `users,B`. This enforces the *need to know* principle.

- Enforce system high/low security labels
  When this option is ON, the UNICOS system files (for example /etc directory) are only maintained by the security administrator (secadm).
  This option should be activated with "Trusted UNICOS", but we keep it turned off because we want the system administrator (sysadm) to be able to manage the system files.

- Super-user privilege policy *and* UNICOS 7.0 version of TFMgmt
  These options are set to OFF in order to get a PAL-based privilege mechanism and suppress the capability of the super user (uid 0) to override system restrictions, since different administrative roles fit our computer center organization.

2.3.2 Network options

Figure 5 describes the main network options.
We explain the reasons for setting some of these options :

```
          MLS Network Security Options

S-> OK to export sensitive data via NFS         YES
    OK to write sensitive data via NFS          YES
    Strict B1 evaluation rules                  YES
    Default to multi-level privileged
       sockets for compatibility                NO
    Traditional hosts.equiv & .rhosts           NO
    Network-Protocols Security Configuration ==>
```

Figure 5: Network options

- Strict B1 evaluation rules
  This option prevents changing the label of a connected socket regardless of the type of remote host. The connection is interrupted while a difference exists between the socket and the host label, preventing unauthorized data transfer between networks `A` and `B`.

- Default to multi-level privileged sockets for compatibility
  This option is set to NO, thus a server process cannot handle multiple connections with different security labels. It must create a child process for each one. This conforms with Trusted UNICOS configuration.

- Traditional hosts.equiv & .rhosts
  We set this option to NO to limit the use of `.rhosts` files by requiring to use the same login name on the client and server. Moreover, the user is required to have an entry in `.rhosts` file for the login, **and** the client system must be in `/etc/hosts.equiv` file.
  Thus, using someone else's account without authentication cannot be done. Sharing resources must conform to local policy.

3 MLS customization

3.1 System management

As mentioned in the previous section, we chose a trusted facility management based on the PAL mechanism, and inhibited root-based and UNICOS 7.0 compatible TFMgmt.

3.1.1 Administrative roles

A good use of PAL-based privilege mechanism requires the ability to define on-site administrative roles (associated with categories) and divide on-site administrative tasks among those roles.
The main problem occurs when trying to classify tasks requiring the `secadm` or `sysadm` categories.

3.1.2 Evolution of the `root` login

Even if the `root` login still exists in a PAL-based privilege mechanism, this user can no longer override system

restrictions : tasks must be done by appropriate users with appropriate categories.
This is true in both interactive and batch environments :

- administrators, used to executing traditional system tasks[4]through the `root` capabilities, have to change their practices.
- most of the shell scripts running in the `root` crontab must be dispatched in `sysadm` or `secadm` crontabs.

3.1.3 File access control

Some commands, for instance `cc(1)`, `make(1)`, do not have a PAL. Consequently, although an administrator with an active category can remove or visualize a read-protected source file, he will not be able to compile it. The only solution is to change its permission bits[5].
More generally, a good use of PAL-based privilege mechanism implies a strict management of users and groups permissions for files and directories.
Hiding a poor file permission management by using too many privileges is strongly discouraged.

3.2 DNS and NIS substitution

Instead of using full non evaluated software, we developed secure software dedicated to specific needs. We were able to get rid of DNS and NIS services on our CRAY machines because we currently used a restricted number of capabilities of these products.
However, these services are still running on remote servers and so can be accessed through remote commands to get basic information.

- DNS
  Domain name service (DNS) is the distributed name and address mechanism used to associate names of computers with addresses.
  When DNS is enabled, the `/etc/hosts` file is no more used for this translation.
  The DNS configuration allows to define a machine as a secondary server : local address resolution is made from data periodically updated from primary servers.

  Instead of using this mechanism, we wrote a trusted process getting[6]information from DNS servers located on the two networks and updating the `/etc/hosts` file. The same program manages `/etc/hosts.equiv` and `/etc/config/spnet.conf` (NAL) files. It was given the "PRIV_MAC_RELABEL_SUBJECT" privilege to be able to communicate with servers having different security labels.

  In this program, control features were developed, to screen the machines that will be allowed to connect to the CRAY, and restrict each of them to a specific label.
  So, this product can control the networks better than standard DNS services.

- NIS
  The network information service (NIS) provides a simple network look-up service.
  On our Cray machines, this service was only used to get information for a locally defined map and only through a local library call.
  Thus, we wrote a process periodically getting these information and replaced the `yp_match(3C)` call in our library routine.

3.3 Changes in local software

The PAL-only mechanism implies some changes in local system software. In this section, we present the main changes that we have done.

3.3.1 Daemons and commands

In a secure environment, system daemons and commands responsibility is extended to privileges management, requesting user recognition and data security label management.

To conform with the *least privilege* principle, privileges have to be managed by the process depending on the current active category.

Whenever privileges are necessary to user requests, privilege text can be used to provide special processing to administrators (for instance, a user will be able to display his own daemon requests and an administrator will be able to display all of them).
Uids do not have to be checked anymore.

For example, the following C sequence :

```
if ( getuid() != 0 )
```

should be replaced by

```
if ( cmptext(PRIVTEXT, ROOT_EFFECTIVE|ROOT_REAL) != 0 )
```

3.3.2 Client/server application

To conform with the TCSEC rules which do not allow a process to handle multiple connections with different security labels and to change a connected socket security label, server programs may need to be restructured : a server process must fork as many child processes as labels[7], and then open a connection socket in each one.

[4] updating system files, reading system logfiles ...
[5] using `chmod(1)`
[6] with a `remsh(1B)` command
[7] more generally, the server process may fork a child process for each connection

3.3.3 System shell scripts

System shell scripts are developed and used to automate administrative tasks : UDB management, dump/restore, DMF auditing, spooling mechanism , logfile processing ... To perform such operations, administrative categories can be required[8].

Moreover, for shell scripts dealing with labelled data (dump/restore), changes[9]are needed to match the script security label with the data security label.
In some cases, a privileged shell will be necessary.

4 Using MLS

This section focuses on technical issues encountered during general use of MLS features.

4.1 Privileges

Effective privileges are a subset of permitted privileges and allow a process to effectively override system restrictions during system call execution. For example, a process with effective privilege PRIV_KILL can send a signal to any process.

The privileges of a process can change in the following ways :

- a process can switch a privilege from permitted to effective
- every process can lower its privileges
- the PRIV_MAC_RELABEL_SUBJECT privilege allows a process to increase its privileges
- during an `exec(2)` system call, the process' privileges change, depending on the active category of the calling process and the executed file's security attributes (PAL and file privilege set).
  The mechanism is quite complex : some privileges can be inherited (although it seems to be rarely used) or gained.

Note that privileges don't change during a `fork(2)` system call. The child process inherits its parent's privileges.

4.2 Privileged shell

A shell is a process and so, can have privileges. For instance, the file `/bin/ksh` has the following PAL :

```
other:PRIV_NULL:TEXT_NULL
secadm:PRIV_KILL,PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE,
PRIV_MAC_RELABEL_SUBJECT,PRIV_RESOURCE:TEXT_NULL
sysadm:PRIV_KILL,PRIV_DAC_OVERRIDE,PRIV_RESOURCE:TEXT_NULL
sysops:PRIV_KILL,PRIV_RESOURCE:TEXT_NULL
system:PRIV_KILL,PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE,
PRIV_MAC_RELABEL_SUBJECT,PRIV_RESOURCE:TEXT_NULL
```

When `/bin/ksh` is executed with an active category, the resulting process is a privileged shell.

[8] add `setucat(1)` command
[9] add `setulvl(1)` and `setucmp(1)` commands

4.2.1 How to get a privileged shell

After setting an active category, a command will be executed by a privileged shell when :

- interactively entering ksh and then typing in the command
- executing `ksh -c "command"`
- calling the system call `system(3C)` within a C program
- inserting the command in a shell script file beginning with #!/bin/ksh
- inserting the command in a shell script called as an argument to `/bin/ksh` :
  `/bin/ksh shell_script_file_name`.

4.2.2 When to use a privileged shell

When you type in a command, some system calls are made by the shell process itself :

- opening files to deal with redirection of input/output
- reading the directory structures to expand the metacharacters ('*', '?', ...)
- processing some built-in commands `cd`, `kill`, `setulvl`, `setucmp`, `setusrv` ...

If privileges are needed for these system calls, the shell has to be privileged.

4.2.3 Examples

The `secadm` security label is 0,0.

```
secadm >setucat secadm     #secadm as active category
secadm >setulvl 1
ksh: cannot set security level: 1
```

A privileged shell is needed to change one's security label.

```
secadm >cd /tmp.mld/001400
ksh: /tmp.mld/001400: permission denied
```

We needed a privileged shell because the system call `chdir(2)` needs a privilege in order to change to a directory with a different security label.

```
secadm >ls /tmp.mld/001400
foo
```

It works, because `ls(1)` command has a PAL.

```
secadm >cd ~sam/test
secadm >ls -la
total 24
drwx--x--x   2 sam      users      4096 Jun  2 10:18 .
drwx--x--x  14 sam      users      4096 Jun  2 10:18 ..
-rw-------   1 sam      users         7 Jun  2 10:18 foo
```

`ls` command has a PAL allowing to override discretionary access control (reading a protected directory).

```
secadm >ls -la *
*: No such file or directory
```

The shell could not expand character '*', because it is not privileged, and current directory is read-protected.

```
secadm >ksh -c "ls -la *"
-rw-------   1 sam      users        7 Jun  2 10:18 foo
```

No problem with a privileged shell!

4.3 Playing with mlds

4.3.1 Example

Mlds are used when the same directory name has to be used by subjects from different security labels.
For example, `/tmp` is a mld : `/tmp` is a multilevel symbolic link to `/tmp.mld`, which contents a subdirectory for each security label as shown in figure 6.

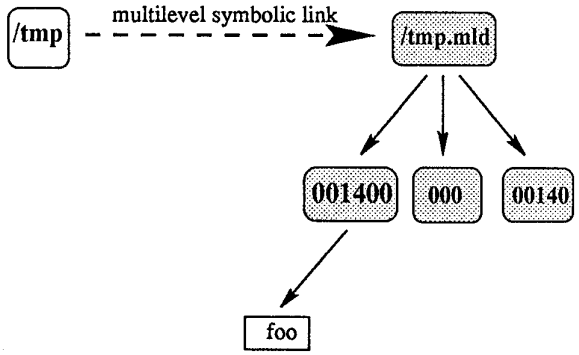

Figure 6: Example of mld

A user at security label `users,B` opening file `/tmp/foo` will in fact access `/tmp.mld/001400/foo`, because `001400` is the internal representation of label `users,B`.

4.3.2 Mld definition in terms of symbolic links

For a user at label `users,B`, everything works as if `/tmp` was a symbolic link to `/tmp.mld`, the latter being a symbolic link to directory `/tmp.mld/001400`.

4.3.3 Possible issues

Specifying `/tmp/foo`, a user at label `users,B` will transparently access `/tmp.mld/001400/foo`.
But, if the same user wants to access[10] file ".." from `/tmp` directory (in fact `/tmp.mld/001400`), he will get `/tmp.mld`.

This behavior is compatible with mld definition in terms of symbolic link, but is not transparent.

As well, while `ls /tmp` command gives the list of files in directory `/tmp.mld/001400`, `ls -l /tmp` will give information about symbolic link `/tmp` itself :

```
lrwxrwxrw    1 root     root       7 May 22 16:47 /tmp -> tmp.mld
```

To get a standard behavior (as if there was no mld), you must specify the `-L` option to "follow" the symbolic links. This option can be useful when trying `ls -l /tmp | grep sam` to show files belonging to user `sam` in `/tmp` directory. This will not work as expected without the `-L` option.

The `cd` and `pwd` built-in commands processing is shell-dependent:

- with csh(1)
  The `pwd` command gives the current directory as a "physical path" which includes all the subdirectory names. The `cd ..` command does not process symbolic links.

```
$ cd /tmp
$ pwd
/tmp.mld/001400
$ cd ..
$ pwd
/tmp.mld
```

- with ksh(1)
  The shell remembers what symbolic links were used to access current directory. The `pwd` command gives the "logical path" made of these links and the `cd ..` command takes them into account. These behaviors hide the mld structure to most users.
  Note that it is still possible to get the "physical path" by calling `/bin/pwd(1)`.

```
$ cd /tmp
$ pwd
/tmp
$ /bin/pwd
/tmp.mld/001400
$ cd ..
$ pwd
/
```

4.3.4 Conclusion

The use of mld is transparent to users in most cases, if they call `ls` command with `-L` option and use `/bin/ksh`. In other cases, they must refer to the mld definition in terms of symbolic links.

5 Results and discussion

In this section, we discuss about benefits and impacts of bringing up a UNICOS MLS system.
MLS features provide capabilities of managing user and administrative environments.

5.1 Users

This study has shown the ability to share resources on a CRAY computer while conforming to specific security requirements. Because of the very few resulting changes in the system behavior, end users can easily switch to a MLS system. Moreover, users no longer have to deal with

[10] for instance by calling `open("..")`

different machines but only with different environments[11]; data privacy is automatically enforced by security labels.

5.2 Administrators

MLS security features provide administrators with many access control capabilities :

- MAC on system files
  By using security labels, one more control (MAC) is added, which prevents unauthorized acces to system files.

- network access control
  The use of standard MLS mechanisms (NAL, WAL, ...), as well as our DNS replacement tool, allows a strict control of the network connections. Administrators have the ability to allow or deny connection to and from a remote computer, and to prevent unauthorized accesses.

The PAL-based privilege mechanism entails the defining of different administrative roles and the modifying of system procedures : this enforces the *least privilege* principle.
Getting out of using root functionalities is required but security is increased.

With MLS features enabled, a very complete and powerful auditing mechanism [4] is available.

All these controls lead to a better reliability on system and data integrity.
On the other hand, bringing up UNICOS MLS system requires some system developments and to get used to new mechanisms.

6 Conclusion

We have shown that all of our security requirements can be addressed by using the MLS security features.
The system security and control capabilities are enhanced.
These improvements can be obtained at reasonable costs.
Impacts on users are not significant but some modifications in software programs and changes in administrative usage are required.

Using MLS security features can lead to a better organization of the administrative work (structuring needs and solutions, analyze, programming and coordination of system tools).

After this first experience, we will focus on auditing mechanisms (needs evaluation, customization, ...).

As a conclusion of this study, we are now able to develop secure software, install and manage a secure environment on a Cray system shared by several communities of users. CEL-V will now have the possibility to choose one powerful computer instead of duplicating resources.

[11] based on the concept of mlds

Acknowledgements

This paper is based on several studies done at CEL-V by J.L Tsirony from Cray Research France. We would like to thank him for his helpful comments and guidance during our work.

References

[1] Inc. Cray Research. *Unicos System Administration , SG-2113 8.0.*

[2] Inc. Cray Research. *UNICOS System Security Overview for Administrators, SG-2141 8.0.*

[3] Paul Falde. Security futures. In *CUG 1995 Spring Proceedings*, pages 285–286, 1995.

[4] Don Hankins. Important aspect of unicos 8.0 auditing. In *CUG 1994 Fall Proceedings*, pages 379–381, 1994.

[5] NCSC. *DoD Trusted Computer System Evalution Criteria.*

[6] William D. Ralph. Digging for clams in a minefield: Experiences with unicos security. In *CUG 1991 Spring Proceedings*, pages 258–264, 1991.