CSCI 7000-001 — Trusted Systems: Models and Design Principles

November 20, 2001

1 Trusted System

What is a *trusted system*?
As usual, just like communication security, system security is defined in relation to a *security policy*.
A trusted system is a system that is believed to realize its defined security policy.

2 Design Principles

- policy definition: security models
- system design: security as a "first class" requirement
- security techniques
- evaluation

3 Models

- military security
- commercial security policies

- multi-level security
  - confidentiality (Bell–La Padula)
  - integrity (Biba)
- general capability-based models

4 Military model

The classic military security model is in fact simply a *confidentiality* model.
It is based on two combined principles:

- sensitivity levels
- need-to-know

Information (objects) is classified according to its level of sensitivity level (*unclassified*, *restricted*, *confidential*, *secret*, *top secret*, ...) within a set of *compartments*. The *class* or *classification level* of an object o is given by the pair (rank_o,compartments_o)
A subject (a person) has an identical classification called *clearance*: (rank_s,compartments_s)
A subject s may access an object o only if and only if $\text{rank}_s \geq \text{rank}_o$ *and* $\text{compartments}_s \subseteq \text{compartments}_o$.

5 General multi-level security

The military security model can be generalized to what is commonly referred to as *multi-level security*.
The principle behind multi-level security is that both objects and subjects are rated according to some combination of security parameters:

$$R : O \to L$$
$$C : S \to L$$

R is the rating function for objects, C is the clearance function for subjects, and L is the (common) range of security levels for both C and R. A *partial order* relation $\geq$ relation is also defined on L.
A subject s may access an object o iff $C(s) \geq R(o)$.
Notice that:

- for consistency, it is good to define the partial order over L in such a way that there exist an upper bound u (such that $\forall x \in L : u \geq x$) and a lower bound l (such that $\forall x \in L : x \geq l$)
- a multi-level security scheme is parametric with respect to the type of access

6 Bell-La Padula confidentiality model

Assuming a multi-level security classification:

- *simple security property*: a subject s may have *read* access to an object o only if $C(s) \geq R(o)$
- **-property*: a subject s with *read* access to an object o may have *write* access to another object p only if $R(p) \geq R(o)$

The first security property defines the intuitive multi-level confidentiality policy: you can only read information at your level or below.
The second property prevents information from leaking from higher levels to lower levels. Notice that the second property allows a subject s to write information o at a lower level ($C(s) \geq R(o)$), provided that s does not have access to any other object at a higher level than o.

7 Biba integrity model

The Biba model is a dual model of the Bell-La Padula. Biba assumes a multi-level security classification, however this time the clearance function for subjects must be interpreted as integrity clearance (or *trustworthiness* to produce and modify information).

- *simple integrity property*: a subject s may have *write* access to an object o only if $T(s) \geq I(o)$
- **-property*: a subject s with *write* access to an object o may have *read* access to another object p only if $I(p) \geq I(o)$

I use two different rating and clearance functions (I and T) not to confuse them with their confidentiality counterparts.
The first security property defines the intuitive multi-level integrity policy: you can only write information at your integrity level or below.
The second property prevents less trustworthy information from polluting information with higher trust level.
Notice also that it is not immediately clear how to combine this integrity model with its dual model of confidentiality.

8 General capability models

Notice that neither Bell-La Padula nor Biba define how the rating and clearance functions are defined or modified. Also neither of them expresses the ability to *delegate* or transfer access rights.
A number of more general models have been developed to address these issues. The general idea (Graham-Denning) is to define an *access matrix* (A)to model a set of access rights R for each combination of objects and subjects.
More precisely, the Graham-Denning model defines a set of basic *rights* in terms of commands that a given subject x can execute on an object o ($A[x, o]$) or on another subject y ($A[x, y]$). These are:

- *create object*: x creates a new object o
- *delete object*: x destroys an existing object o
- *create subject*: x creates a new subject s
- *delete subject*: x destroys an existing subject s
- *read access rights*: x reads access rights of subject s on object o
- *grant access rights*: x grants access right r to subject s on object o
- *delete access rights*: x removes access right r to subject s on object o
- *transfer access rights*: x transfer access right r or $r*$ to subject s on object o. $r*$ means that access right r is *transferable*, that is, s may in turn transfer r or $r*$ to other subjects

9 Access rules

The general model defines preconditions for the execution of every command (access rights). Another way of looking at access rights is to define them as *access rules*. For example:

$$\textbf{if} r_1 \in A[s_1, o_1] \textbf{and} r_2 \in A[s_2, o_2] \ldots \textbf{then} \ldots$$

10 Model checking

One of the interesting aspects of these general models is that they are suitable for automatic verification. The idea is to use *model checking* techniques to prove properties regarding one particular set of rules.

11 Ordinary security features in operating systems

Most operating systems provide some basic security features:

- *user identification and authentication of users*
- *virtual memory*
- *mediated I/O*
- *(fair) resource allocation*
- *mediated inter-process communication*

12 Security features of trusted systems

- *user identification and authentication*
- *mandatory access control*
- *discretionary access control*
- *object resue protection*

- *complete mediation*
- *audit*
- *audit log reduction*
- *trusted path*
- *intrusion detection*

13 Security kernel

The idea behind a *security kernel* is to concentrate all the security enforcement mechanisms in a single operating system component. This idea has the following advantages:

- easier protection of security mechanisms themselves
- usual benefits of modular design
- compact codebase, and therefore less bugs

The most important component within the security kernel is the *reference monitor*. The reference monitor is what implements the access control function, granting or denying access.

14 Trusted computing base

Another fundamental concept in the design of secure (trusted) operating systems is that of *trusted computing base*. The trusted computing base contains all the security-critical components of a system. These are some common functions managed by the TCB:

- *basic hardware management*: CPU modes, memory access, critical registers, basic interrupt handling, clock, primitive I/O access
- *processes*: the basic notion of process as a basis to identify subjects and to allocate resources
- *interprocess communication*: basic communication mechanisms

15 Virtualization

Another fundamental concept/technique is the design of secure systems is *virtualization*. The ideas behind virtualization are:

- *complete mediation*: the application executes in a virtual environment in which every operation is mediated by some form of virtual machine layer
- *complete separation*: virtual environments are completely independent and separated from each other. Interaction is possible through some very specific, system-mediated, application-level mechanisms