ABOUT INTEGRITY IN SECURITY MODELS

by
Grigor Moldovan and Mădălina Văleanu

Abstract. Security models are an important concept in the design and analysis of secure systems. They capture the security policy that should be enforced in the system.

Keywords: security, integrity, integrity levels

Introduction

State machine are a most popular tool for modeling computing systems. A state is a representation of the system investigation at one moment in time. The possible state transition can be specified by a state transition function which defines the next state depending on the present state and an input.

A computer security policy consists of a clearly defined and precise set of rules, for determining authorization as a basis for making access control decisions. A security policy captures the security requirements of an establishment or describes the steps that have to be taken to archive the desired level of security.

A security policy is typically stated in terms of subjects and objects, given the desired subject and object there must be a set of rules that are used by the system to determine whether a given subject can be given access to a specific object.

A *security model* is a formal or an informal way of capturing such policies. Security models are an important concept in the design of a system. The implementation of the system is then based on the desired security model. Formal security models such as Bell-LaPadula have a prominent place in high assurance security evaluations. Informal models, such as Clark-Wilson, are a more of a descriptive framework for expressing security policies.

Most security models cannot support a wide range of security policies. They either support static policies or a limited set of security policies.

The Bell-LaPadula Model

Bell-LaPadula (BLP) model is a state machine model capturing the confidentiality aspects of access control. Access permissions are defined both through an access control matrix and through security levels. Bell-LaPadula prevents information flowing downwards from high security level to a low security level.

This model is based on:

- a set of subjects S;
- a set of objects 0;
- the set of access operations A = {execute, read, append, write } that directly mirror the access rights*;*
- a set L of security levels with a partial ordering ≤.

We want to use the state of the system for checking its security, so the state set of out model has to capture all current permission and all current instances of subjects accessing objects. This leads to a rather complicated state $B\times M\times F$, where*:*

- $B = P(S\times O\times A)$ is the set of current accesses. An element $b\in B$ is a collection of tuples (s, o, a), indicating that subject s currently performs operation a on object o.
- M is the set of access permission matrix $M=(M_{SO})_{s\in S,o\in O}$.
- $F \subset L^S\times L^S \times L^O$ is the set of security level assignments. An element $f\in F$ is a triple (f_S, f_C, f_O), where
  - o f_S: S→L gives the *maximal security level* each subject can have;
  - o f_C: S→L gives the *currentl security level* of each subject;
  - o f_O: O→L gives the *classification* of all subject.

Bell-LaPadula model defines security as the property of states. It consists of three properties:

1. The simple security property: (ss-property) The ss-property defines no-read up. Therefore a subject is not allowed to read an object higher than its own security level.

However if a low level subject can read a high level object it could create a (high-level) Trojan horse, which can read high-level objects, an copy the information into a low-level object.

Thus BLP has to control the write access through the *-property.

2. The star-property: (-property)* A subject may not write to an object with a lower classification that the subject has clearance for. Therefore it prevents an authorized subject (user) from declassifying higher-level information and prevents against Trojan-horse attacks.

In addition a higher-level subject is not able to send messages to a lower level subject. However there are two ways this restriction can be escaped:

-Temporarily downgrade a high-level subject, which assumes that a subject forgets all it knew at a higher level, at the moment it is downgraded.

-Identify a set of trusted subjects that can be relied on not to compromise the information.

3. Discretionary security property: (ds-property) Defines a policy where access control is based on named users and named objects. Subject holding access permission may pass that permission on to other subjects at their discretions.

A transition from state $v_1(b_1, M_1, f_1)$ to state $v_2(b_2, M_2, f_2)$ is said to be secure, if both v_1 and v_2 are secure. The state transition *preserves* the ss-property if and only if:

- each $(s, o, a) \in b_2 \setminus b_1$ satisfies the ss-property with respect to f_2, and
- if $(s, o, a) \in b_1$ does not satisfy the ss-property with respect to f_2, then $(s, o, a) \notin b_2$

Basic security theorem

If all state transitions in a system are secure and if the initial state of the system is secure, then every subsequent state will also he secure, no matter which inputs occur.

In practice, the basic security theorem limits the effort needed to verify the security of a system. You are allowed to check each state transition individually to show that it preserves security and you have to identify a secure initial state. As long as you start your system in this secure initial state, it will remain secure.

The Harrison-Ruzzo-Ulman Model

The Harrison-Ruzzo-Ulman (HRU) model defines authorization systems. This model is based on:

- a set of subjects S;
- a set of objects 0;
- the set of access rights R;
- an access matrix $M=(M_{SO})_{s \in S, o \in O}$; the entry M_{SO} is the subset of R specifying the rights subjects *s* has an object *o*.

There exist some basis operations for manipulating the set of subjects, the set of objects and the set of matrix:

enter r into M_{SO}
delete r from M_{SO}
create subject s
delete subject s
create object o
delete object o

If you design complex systems that can only he described by complex models, it becomes difficult to find proofs of security. In the worst case (undecidability), there does not exist a universal algorithm that verifies security for all problem instances. If you want verifiable security properties, you are better off when you limit the complexity of the security model. Such a model may not describe all desirable security properties, but you may gain efficient methods for verifying security.

The Chinese Wall Model

The Chinese Wall model proposed by Brewer and Nash models access rules in a consultancy business where analysts have to make sure that no conflicts of interest arise when they are dealing with different clients (companies). Analysts have to adhere to the following security policy: *There must be no information flow that causes a conflict of interest.*

The Biba Model

The Biba model addresses integrity in terms of access by subjects to objects using a state machine model very similar to that of BLP. There is a lattice (L, $\leq$) of integrity levels. The functions f_S: S$\rightarrow$L and f_O: O$\rightarrow$L assign integrity levels to subjects and objects. These levels form the basis for expressing integrity policies that refer to the corruption of 'clean' high level entities by dirty' low level entities. In the integrity lattice, information may only flow downwards. Unlike BLP, there is no single high-level integrity policy. Instead, you find a variety of approaches. Some even yield mutually incompatible policies.

Static Integrity Levels

The following two integrity properties are the dual of the mandatory BLP policies:

- *Simple integrity property***:** if subjects s can modify (alter) object a, then f_S (s) $\geq$ f_O(o) .
- *Integrity *-property***:** if subject *s* can read (observe) object a, then *s* can have write access to some other object p only if f_O (p) $\leq f_O$ (o).

These two policies prevent clean subjects and objects from being contaminated by dirty information.

Dynamic Integrity Levels

Similar to the Chinese Wall model, the next two integrity properties automatically *adjust* the integrity level of an entity if it has come into contact with low-level information.

- *Subject low watermark property*: subject *s* can read (observe) an object a at any integrity level. The new integrity level of the subject is inf(f_S (s), f_O(o)), where f_S (s) and f_O(o) are the integrity levels before the operation.
- *Object low watermark property*: subject s can modify (alter) an object o at any integrity level. The new integrity level of the object is inf(f_S (s), f_O(o)), where f_S (s) and f_O(o) are the integrity levels before the

operation.

The integrity level inf(f_S (s), f_O(o)), the greatest lower bound of f_S (s) and f_O(o), is well defined because we are dealing with a lattice of integrity levels.

Policies for Invocation

The Biba model can be extended to include an access operation *invoke*. A subject can invoke another subject to access an object. A distinct policy must then exist for invocation. Two additional properties add this functionality.

Invoke property: subject s_1 can invoke subject s_2 only if $f_S (s_2) \leq f_S (s_1)$
Subjects are only allowed to invoke tools at a lower level. Otherwise a dirty (low-level) subject could invoke a clean (high-level) tool and contaminate a clean object. Alternatively we might want to do this. Dirty subjects will be able to access clean objects but only if they use a clean tool to do so. This tool may perform a number of consistency checks to ensure that the object remains clean. In this scenario we would not want a dirty subject to use a dirty tool and we could adopt the Ring Property.

Ring property: a subject s_1 can read objects at all integrity levels. It can only modify object o with $f_O (o) \leq f_S (s_1)$ and it can invoke a subject s_2 only if $f_S (s_1) \leq f_S (s_2)$.

The Clark-Wilson Model

Clark and Wilson address the security requirements of commercial applications. They argue that these requirements are predominantly about ***data integrity,*** i.e. about preventing unauthorized modification of data, fraud, and errors. This is a rather wide definition of integrity. As a matter of fact, the authors even include issues of *concurrency control,* which are beyond our scope of security. Integrity requirements are divided into two parts:

- Internal consistency: refers to properties of the internal state of a system and can be enforced by the computing system;
- External consistency: refers to the relation of the internal state of a system to the real world and has to be enforced by means outside the computing system.

The general mechanisms for enforcing integrity are:

- well-formed transactions: data items can be manipulated only by a specific set of programs; users have access tt, programs rather th an to data items;
- Separation of duties users have to collaborate to manipulate data and to collude to penetrate the security system.

Information-Flow Models

In the Bell LaPadula model, information can flow from a high security level to a low security level through a covert channel.

A precise and quantitative definition of information flow can be given in terms of information theory. The information flow from x to y would be measured by the change in the equivocation of *x* given the value of y. The components of the information flow model are:

- a lattice (L, ≤) of security labels,
- a set of labeled objects,
- the security policy: information flow from an object with label c, to an object with label c_2 is permitted only if $c_1 \leq c_2$; any information flow that violates this rule is illegal.

References

[1].Bobak Distributed and Multi-Database Systems , Bantam Books, 1993
[2].Cartell R. – Object Data Management , Addison –Wesley – ACM Press Books, 1992
[3].Castrano S., Fugini M., Martella G., Samarati P. : Database Security, Addison-Wesley, 1999
[4].Ceri S., Pelagatti G. – Distributed Databases. Principles and Systems, International Student Edition, McGraw Hill, 1985
[5].Connolly T. M., Begg C. E. : Database Systems, Addison-Wesley, 1999
[6].Date C. J. – An Introduction to Database Systems , Addison –Wesley – ACM Press Books, 1982
[7].Gilula M.– The Set Model for Database and Informational Systems, Addison–Wesley – ACM Press Books, 1999
[8].Sandhu R. S., Jajodia S. : Integrity mechanism in database management systems in Proc. 13th National Computer Security Conference, Oct. 1990

Authors:
Grigor Moldovan, Department of Computer Science, Universitatea Babes-Bolyai Cluj-Napoca, Romania
Mădălina Văleanu, Department of Medical Informatics and Biostatistics,Universitatea de Medicina si Farmacie „Iuliu Hatieganu” Cluj-Napoca, Romania