Bell-LaPadula Model

Why Security Models?

When we have implemented a security policy, do we know that it will (and can) be enforced?

E.g., if policies get too intricate, contradicting rules may apply to a given access request.

To prove that a policy will be enforced, the policy has to be formally specified.

A security model is a formal description of a security policy.

Security Models

Models are used in high assurance security evaluations (smart cards are currently a fruitful area of application).

Models are important historic milestones in computer security (e.g. Bell-LaPadula).

The models presented today are not recipes for security but can be a starting point when you have to define a model yourself.

Agenda

State Machine Models (short review)

Bell-LaPadula (BLP) model

- BLP security policies
- BLP Basic Security Theorem
- Tranquility
- Covert channels

Case Study: Multics interpretation of BLP

State Machine Models

State Machines (automata) are abstract models that record relevant features, like the security of a computer system, in their state.

States change at discrete points in time, e.g. triggered by a clock or an input event.

State machine models have numerous applications in computer science: processor design, programming languages, or security.

Examples

Switch:

- two states, ‘on’ and ‘off’
- One input ‘press’

Ticket machine:

- State: ticket requested, money to be paid
- inputs: ticket requests, coins;
- output: ticket, change

Microprocessors:

- state: register contents,
- inputs: machine instructions

Basic Security Theorems

To design a secure system with the help of state machine models,

– define state set so that it captures 'security',

– check that all state transitions starting in a 'secure' state yield a 'secure' state,

– check that initial state of system is 'secure'.

Security is then preserved by all state transitions. The system will always be 'secure'.

This Basic Security Theorem has been derived without any definition of 'security'!

Bell-LaPadula Model (BLP)

State machine model developed in the 1970s for the analysis of MLS operating systems.

Subjects and objects labeled with security levels that form a partial ordering.

The policy: No information flow from ‘high’ security levels down to ‘low’ security level (confidentiality).

Only considers information flows that occur when a subject observes or alters an object.

Access permissions defined through an access control matrix and security levels.

Constructing the State Set

1. All current access operations:
   - an access operation is described by a triple *(s,o,a),* s *S*, o *O*, a *A*

     E.g.: (Alice, fun.com, read)
   - The set of all current access operations is an element of ***P*** (*S* *O* *A*)

     E.g.: {(Alice, fun.com, read), (Bob, fun.com, write), …}

Constructing the State Set

2. Current assignment of security levels:
   - maximal security level: f_S: $S \quad L$ (L … labels)
   - current security level: f_C: $S \quad L$
   - classification: f_o: $O \quad L$

The security level of a user is the user's clearance.

Current security level allows subjects to be down-graded temporarily (more later).

$F \quad L^S \quad L^S \quad L^O$ is the set of security level assignments; $f = (f_S, f_C, f_O)$ denotes an element of F.

Constructing the State Set

3. Current permissions:

defined by the access control matrix M.

$\mathcal{M}$ is the set of access control matrices.

The state set of BLP: $V = B \quad \mathcal{M} \quad F$

– B is our shorthand for $\mathcal{P}(S \quad O \quad A)$

– b denotes a set of current access operations

– a state is denoted by (b,M,f)

BLP Policies

Discretionary Security Property (ds-property): Access must be permitted by the access control matrix: $(s,o,a) \quad M_{so}$

Simple Security (ss)-Property (<u>no read-up</u>): if $(s,o,a) \quad b$, then $f_S(s) \quad f_O(o)$ if access is in observe mode.

The ss-property is a familiar policy for controlling access to classified paper documents.

On Subjects

In the ss-property, subjects act as observers.

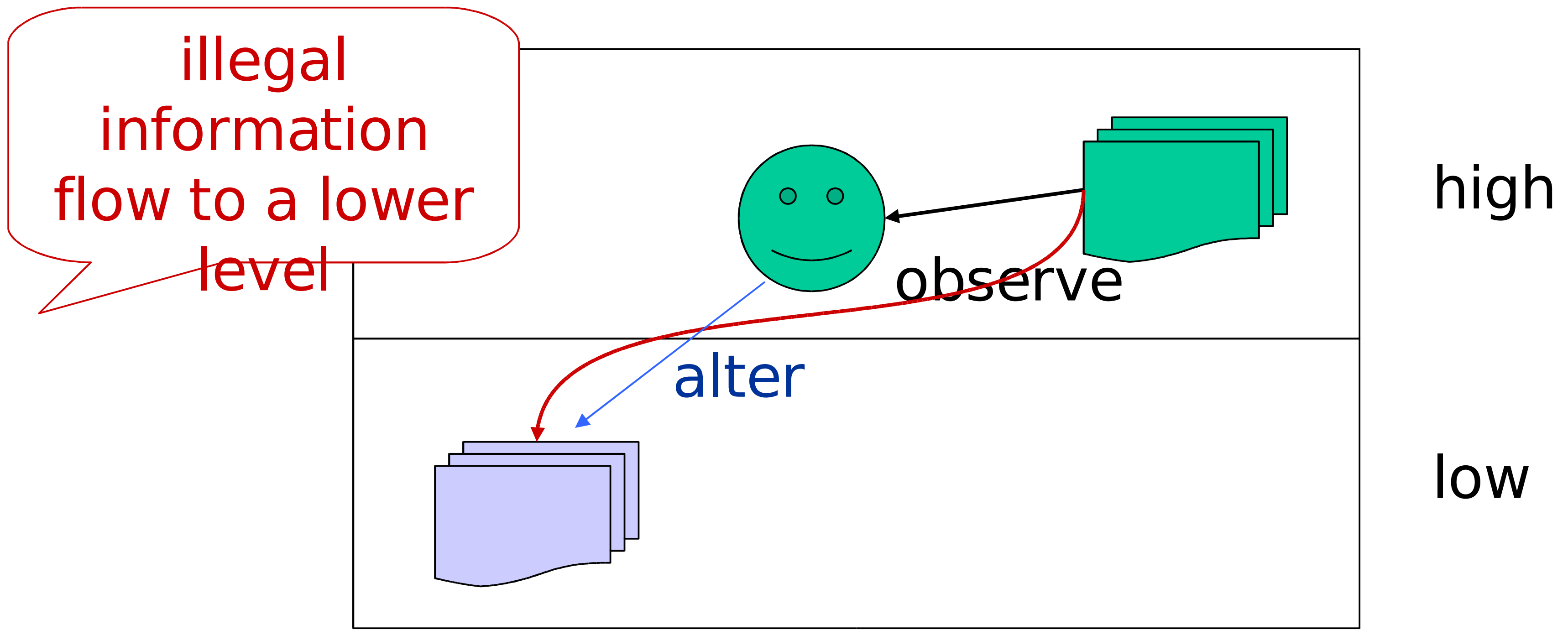
In a computer system, subjects are processes and have no memory of their own.

Subjects have access to memory objects.

Subjects can act as channels by reading one memory object and transferring information to another memory object.

In this way, data may be declassified improperly.

Subjects as Channels



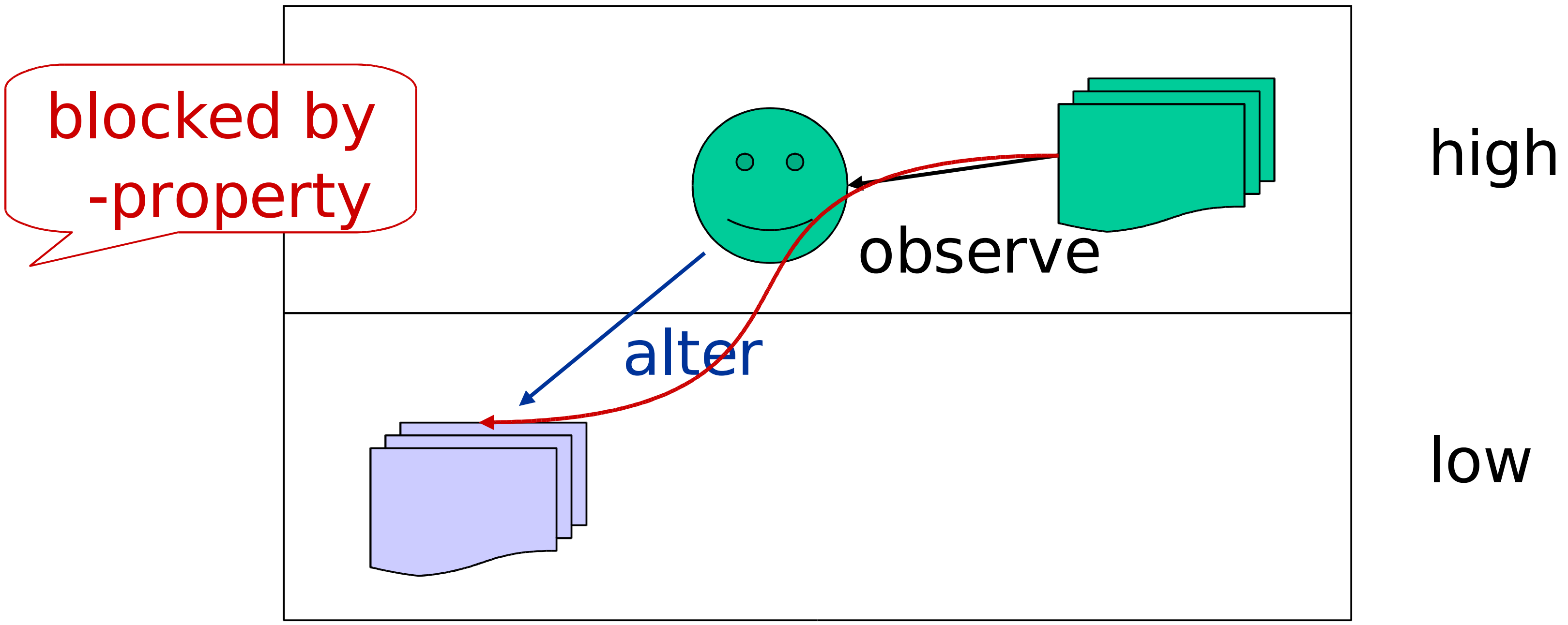
Star Property

-Property (star property) (<u>no write-down</u>): if (s,o,a) b and access is in alter mode then $f_C(s)$ $f_O(o)$; also, if subject s has access to object o in alter mode, then $f_O(o')$ $f_O(o)$ for all objects o' accessed by s in observe mode.

The very first version of BLP did not have the -property.

Mandatory BLP policies: ss-property and -property.

Blocking the Channel



No Write-Down

The -property prevents high level subjects from sending legitimate messages to low level subjects.

Two ways to escape from this restriction:

- Temporarily downgrade high level subject; hence the current security level f_C; BLP subjects have no memory of their own!
- Exempt trusted subjects from the -property.

Redefine the -property and demand it only for subjects that are not trusted.

Trusted Subjects

Trusted subjects may violate security policies! Distinguish between trusted subjects and trustworthy subjects.

Basic Security Theorem

A state is secure, if all current access tuples *(s,o,a)* are permitted by the ss-,   -, and ds-properties.

A state transition is secure if it goes from a secure state to a secure state.

Basic Security Theorem: If the initial state of a system is secure and if all state transitions are secure, then the system will always be secure.

Basic Security Theorem

This Basic Security Theorem has nothing to do with the BLP security policies, only with state machine modeling.

BLP & Security

Construct system with operation *downgrade*:

- downgrades all subjects and objects to system low.
- enters all access rights in all positions of the access control matrix.

As a result, any state is secure in the BLP model.

Should such a system be regarded secure?

- McLean: no, everybody is allowed to do everything.
- Bell: yes, if *downgrade* was part of the system specification.

Tranquility

No BLP policies for changing access control data.

BLP assumes tranquility: access control data do not change.

Operational model: users get clearances and objects are classified following given rules.

The system is set up to enforce MLS policies for the given clearances and classifications.

Changes to clearances and classifications requires external input.

Covert Channels

Communications channels that allow transfer of information in a manner that violates the system's security policy.

- Storage channels: e.g. through operating system messages, file names, etc.
- Timing channels: e.g. through monitoring system performance

Orange Book: 100 bits per second is 'high' bandwidth for storage channels, no upper limit on timing channels.

Covert Channels

The bandwidth of some covert channels can be reduced by reducing the performance of the system.

Covert channels are not detected by BLP modeling.

Applying BLP

Multics

Multics was designed to be a secure, reliable, ..., multi-user O/S.

Multics became too cumbersome for some project members, who then created something much simpler, viz Unix.

The history of the two systems illustrates for relation between commercial success and the balance between usability and security.

We will sketch how the Bell-LaPadula model can be used in the design of a secure O/S.

Multics Interpretation of BLP

The inductive definition of security in BLP makes it relatively easy to check whether a system is secure.

To show that Multics is secure, we have to find a description of the O/S which is consistent with BLP, and verify that all state transitions are secure.

Subjects

Subjects in Multics are processes. Each subject has a descriptor segment containing information about the process

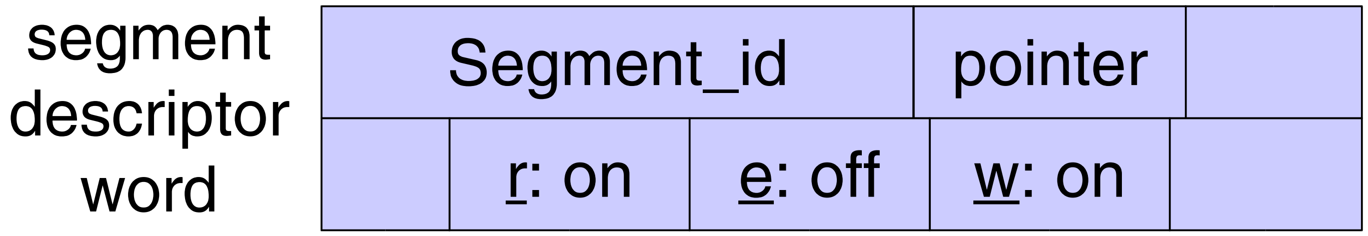
The security level of subjects are kept in a process level table and a current-level table.

The active segment table records all active processes; only active processes have access to an object.

Objects

For each object the subject currently has access to, there is a segment descriptor word (SDW) in the subject's descriptor segment.

The SDW contains the name of the object, a pointer to the object, and flags for read, execute, and write access.



Directories

Objects are memory segments, I/O devices, ...

Objects are organized hierarchically in a directory tree; directories are again segments.

Information about an object, like its security level or its access control list (ACL), are kept in the object's parent directory.

To change an object's access control parameters and to create or delete an object requires write or append access rights to the parent directory.

Compatibility

To access an object, a process has to traverse the directory tree from the root directory to the target object.

If any directory in this path is not accessible to the process, the target object is not accessible either.

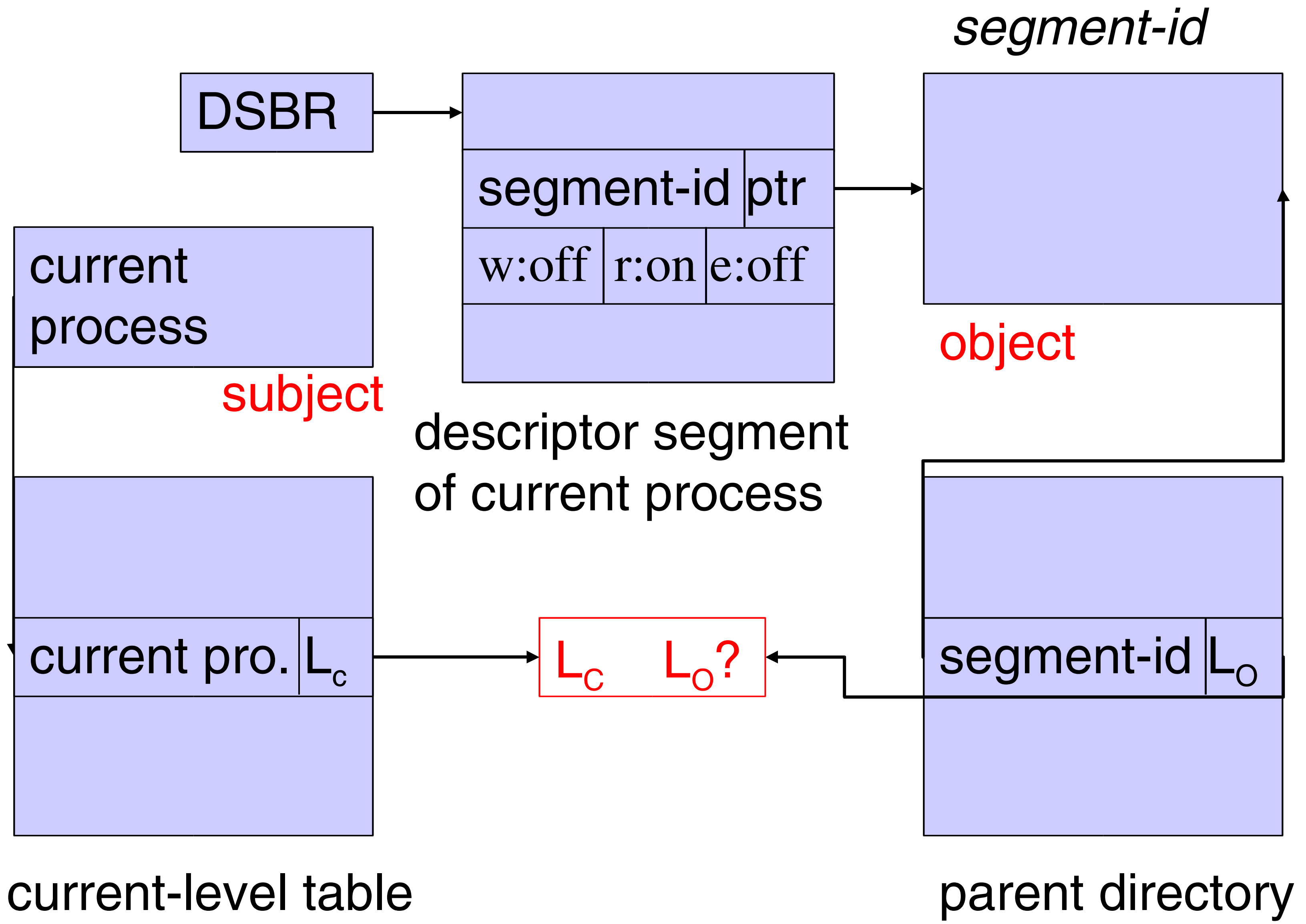
<u>Compatibility</u>: The security level of an object must always dominate the security level of its parent directory.

BLP State in Multics

Current access *b*: stored in the SDWs in the descriptor segments of the active processes; the active processes are found in the active segment table. The descriptor segment base register (DSBR) points to the descriptor segment of the current process.

Level function *f*: security levels of the subjects are stored in the process level table and the current-level table; the security level of an object is stored in its parent directory.

Access control matrix *M*: represented by the ACLs; for each object, the ACL is stored in its parent directory; each ACL entry specifies a process and the access rights the process has on that object.


segment-id
DSBR
segment-id
ptr
w:off
r:on
e:off
current process
subject
descriptor segment of current process
object
current pro.
L_c
L_C L_O?
segment-id
L_O
current-level table
parent directory

MAC in Multics

Multics access attributes for data segments with translation to BLP access rights:

– read	r
– execute	e, r
– read & write	w
– write	a

The -property for Multics

For any SDW in the descriptor segment of an active process, the current level of the process

- dominates the level of the segment if the read or execute flags are on and the write flag is off,
- is dominated by the level of the segment if the read flag is off and the write flag is on,
- is equal to the level of the segment if the read flag is on and the write flag is on.

Kernel Primitives

Kernel primitives are the input operations in Multics

Example: the get-read primitive requests read access to an object

It takes as its parameters a process-id and a segment-id.

If the state transitions in an abstract model of the Multics kernel preserve the BLP security policies, then the BLP Basic Security Theorem proves the 'security' of Multics.

Conditions for get-read

The O/S has to check whether

- the ACL of segment-id, stored in the segment's parent directory, lists process-id with read permission,
- the security level of process-id dominates the security level of segment-id,
- process-id is a trusted subject, or the current security level of process-id dominates the security level of segment-id.

If all three conditions are met, access is permitted and a SDW in the descriptor segment of process-id is added/updated.

More Kernel Primitives

release-read: release an object; the read flag in the corresponding SDW is turned off; if thereafter no flag is on, the SDW is removed from the descriptor segment.

give-read: grant read access to another process (DAC).

rescind-read: withdraw a read permission given to another process.

More Kernel Primitives

create-object: create an object; the O/S has to check that write access on the object's directory segment is permitted and that the security level of the segment dominates the security level of the process.

change-subject-current-security-level: the O/S has to check that no security violations are created by the change

This kernel primitive, as well as the primitive change-object-security-level were not intended for implementation (tranquility).

Aspects of BLP

Descriptive capability of its state machine model: can be used for other properties, e.g. for integrity.

Its access control structures, access control matrix and security levels: can be replaced by other structures, e.g. by $S \quad S \quad O$ to capture ‘delegation’.

The actual security policies, the ss-,   -, and ds-properties: can be replaced by other policies (see Biba model).

A specific application of BLP, e.g. its Multics interpretation.

Limitations of BLP

Restricted to confidentiality.

No policies for changing access rights; a complete general downgrade is secure; BLP intended for systems with static security levels.

BLP contains covert channels: a low subject can detect the existence of high objects when it is denied access.

Sometimes, it is not sufficient to hide only the contents of objects. Also their existence may have to be hidden.