Security Models

- A security model is a formal description of a security policy.
- What is a security policy?
- A security policy could capture the security requirements of an enterprise or describe the steps that have to be taken to achieve security.
- Security models are used in security evaluation, sometimes for proofs of security.
- The Bell-LaPadula model (BLP) is an important historic milestone in computer security.

Agenda

- The Bell-LaPadula model
- Try to be more general: Harrison-Ruzo-Ullman
- Change of access rights: Chinese Wall model
- Integrity models: Biba, Clark-Wilson
- Perfection: information flow and non-interference models

State Machine Models

- Capture the state of a system. States change only at discrete points in time, e.g. triggered by a clock or an input event.
- The state should capture the essential features of the system under investigation, e.g. the security of a computer system.
- How to use state machine models?
  - Define the state set so that it captures `security'.
  - Check that all state transitions starting in a `secure' state yield a `secure' state.
  - Check that the initial state of the 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)

- BLP is a state machine model capturing confidentiality aspects of access control.
- Access permissions are defined through an access control matrix and through a partial ordering of security levels.
- Security policies prevent information flowing down-wards from a high security level to a low security level.
- BLP only considers the information flow that occurs when a subject observes or alters an object.

What do we have to model?

1 All current access operations:
   - an access operation is described by a tuple *(s,o,a)*, s ∈ *S(ubjects)*, o ∈ *O(bjects)*, a ∈ *A(ccess_Operations)*
   - The set of all current access operations is an element of $P(S \times O \times A)$.
   - We use B as shorthand for $P(S \times O \times A)$.
   - We use b to denote a particular set of access operations.

2 The current permissions as defined by the access control matrix M
   - $\boldsymbol{M}$ is the set of access control matrices.

What do we have to model?

3 The current assignment of security levels:
   - maximal security level: $f_S: S \rightarrow L$ (L … labels)
   - current security level: $f_C: S \rightarrow L$
   - classification: $f_o: O \rightarrow L$

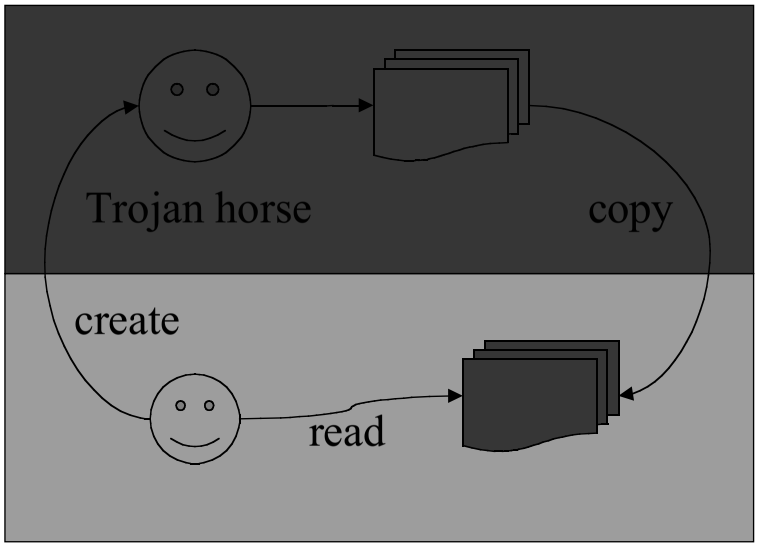
- The security level of a user is the user's clearance.
- The current security level allows subjects to be down-graded temporarily.
- $F \subseteq L^S \times L^S \times L^O$ is the set of security level assignments.
- $f = (f_S, f_C, f_O)$ denotes an element of F.
- The state set of BLP: $V = B \times \boldsymbol{M} \times F$
- A state is denoted by *(b,M,f)*.

BLP Policies

- Prevent information flow from `high' security levels to `low' security levels.
- In BLP, information flow can only occur directly through access operations.
- **Simple Security Property (ss-property)**

  No read-up: $f_S(s) \geq f_O(o)$ if access is in observe mode

- Information flow is still possible.
  - A low subject could create a high level Trojan horse program that reads a high level document and copies its contents to a low level file.
  - This would constitute an improper `declassification' of the document.



BLP Policies ctd.

- ∗ **- Property (star property)**

  No write-down: $f_C(s) \leq f_O(o)$ if access is in alter mode; also, if subject s has access to an object o in alter mode, then $f_O(o') \leq f_O(o)$ for all objects o' accessed by s in observe mode.

- The very first version of BLP did not consider the ∗ - property.
- Mandatory BLP policies: ss- property and ∗ - property.
- **Discretionary Security Property (ds-property)**

  Access must be permitted by the access control matrix: $(s,o,a) \in M_{so}$.

No Write-Down

- The ∗ - property implies that a high level subject is not able to send messages to a low level subject.
- There are two ways to escape from this restriction.
  - Temporarily downgrade a high level subject. This is the reason for the current security level f_C. BLP assumes that subjects have no memory of their own!
  - Identify a set of trusted subjects, which are permitted to violate the ∗ - property.
- We redefine the ∗ - property and demand it only for subjects, which are not trusted. 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-property, * - property, and ds-property.
- 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.

- ***This has nothing to do with security, only with state machine modeling.***

Tranquility

- McLean: consider a system with an operation *downgrade*:
  - downgrades all subjects to system low
  - downgrades all objects to system low
  - enters all access rights in all positions of the access control matrix.
- The resulting state is secure according to BLP.
- Should such a system be regarded as secure?
  - McLean: no, everybody is allowed to do everything
  - Bell: yes, if *downgrade* was part of the system specification
- Problem: BLP has no policies for changing access control data.
- Fact: BLP assumes tranquility, i.e. access control data do not change.

Covert Channels

- Covert Channel: a communications channel that allows 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.
- The bandwidth of some covert channels can be reduced by reducing the performance of the system.
- ***Covert channels are not detected by BLP modeling.***

Aspects of BLP

- The descriptive capability of its state machine model;
  - can be used for other properties, e.g. for integrity
- The access control structures proposed, access control matrix and security levels;
  - can be replaced by other structures, e.g. on $S \times S \times O$ to capture `delegation'.
- The actual security policies, the ss-property, $*$ - property, and ds-property;
  - 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 general and complete downgrade is secure; BLP is 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.***

Harrison-Ruzo-Ullman Model

- BLP does not state policies for changing access rights or for the creation and deletion of subjects and objects.
- The Harrison-Ruzzo-Ullman (HRU) model defines authorisation systems that address these issues.
- The components of the HRU model:
  - a set of subjects S,
  - a set of objects O,
  - a 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 subject s has on object o.

Primitive Operations in HRU

- There exist six primitive operations for manipulating subjects, objects, and the access matrix:
  - enter *r* into M_{so}
  - delete *r* from M_{so}
  - create subject *s*
  - delete subject *s*
  - create object *o*
  - delete object *o*

- Commands in HRU model:

$c(x_1,\ldots,x_k)$
  if r_1 in $M_{s1,o1}$ and
  if r_2 in $M_{s2,o2}$ and
  :
  if r_m in $M_{sm,om}$
  then
    op_1
    op_2
    :
    op_n
end

- s_i and o_i taken from $x_1,\ldots,x_k$

Examples

- Subject *s* creates a file *f* so that *s* owns the file (access right *o*) and has read and write permission to the file (access rights *r* and *w*).

command create_file(*s,f*)
  create *f*
  enter *o* into $M_{s,f}$
  enter *r* into $M_{s,f}$
  enter *w* into $M_{s,f}$
end

- The owner *s* of file *f* grants read access to another subject *p* with

command grant_read(*s,p,f*)
  if *o* in $M_{s,f}$
  then enter *r* in $M_{p,f}$
end

Leaking of Rights in HRU

- The effect of a command is recorded as a change to the access matrix.
- Hence, the access matrix describes the state of the system.
- The HRU model can capture security policies regulating the allocation of access rights. To verify that a system complies with such a policy, you have to check that there exists no way for undesirable access rights to be granted.
- An access matrix *M* is said to leak the right *r* if there exists a command *c* that adds the right *r* into a position of the access matrix that previously did not contain *r*.
- An access matrix *M* is said to be safe with respect to the right *r* if no sequence of commands can transform *M* into a state that leaks *r*.

Safety Properties of HRU

- **Theorem**. Given an access matrix *M* and a right *r*, verifying the safety of *M* with respect to *r* is undecidable.
- The safety problem cannot be tackled in its full generality. For restricted models, there is a better chance of success.
- Mono-operational commands contain a single operation.
- **Theorem**. Given a mono-operational authorisation system, an access matrix *M*, and a right *r*, verifying the safety of *M* with respect to *r* is decidable.
- With two operations per command, the safety problem is again undecidable. Limiting the size of the authorisation system is another way of making the safety problem tractable.
- **Theorem**. The safety problem for arbitrary authorisation systems is decidable if the number of subjects is finite.

The 3rd Design Principle.

- If you design complex systems that can only be described by complex models, it becomes difficult to find proofs of security. In the worst case (undecidability), there does not exist an universal algorithm that verifies security in all cases.
- If you want verifiable security properties, you are better off when the complexity of the security model is limited. Such a model may not describe all desirable security properties, but you may gain efficient methods for verifying `security'. In turn, you are advised to design simple systems that can be adequately described in the simple model.
- ***The more expressive a security model is, both with respect to the security properties and the systems it can describe, the more difficult it is usually to verify security properties.***

Chinese Wall Model

- In financial institutions analysts deal with a number of clients and have to avoid conflicts of interest.
- The model has the following components
  - subjects: analysts
  - objects: data item for a single client
  - company datasets: $y:O \rightarrow C$ gives for each object its company dataset
  - conflict of interest classes: companies that are competitors; $x: O \rightarrow C$ gives for each object o the companies with a conflict on interest on o
  - `labels': company dataset + conflict of interest class
  - sanitized information: no access restrictions.

Chinese Wall Model - Policies

- **Simple Security Property:** Access is only granted if the object requested
  - is in the same company dataset as an object already accessed by that subject
  - belongs not to any of the conflict of interest classes of objects already accessed by that subject
- Formally:
  - $N = (N_{so})_{s \in S, o \in O}$, Boolean matrix, N_{so} = true if *s* has accessed *o*
  - ss-property: subject *s* gets access to object *o* only if for all objects *o'* with $N_{so'}$ = true, $y(o) \notin x(o')$ or $y(o)=y(o')$.

Chinese Wall Model - Policies

- Indirect information flow: two competitors, *A* and *B*, have their account with the same *Bank*. *Analyst_A*, dealing with *A* and the *Bank*, updates the *Bank* portfolio with sensitive information about *A*. *Analyst_B*, dealing with *B* and the *Bank*, now has access to information about a competitor.
- $*$ - **Property:** A subject s will be permitted write access to an object only if s has no read access to any object o', which is in a different company dataset and is unsanitized.
- Formally: subject *s* gets write access to object *o* only if *s* has no read access to an object *o'* with $y(o) \neq y(o')$ or $x(o') \neq \{\}$.
- Access rights of subjects change dynamically with every access operation.

Biba Model

- Biba is a state machine model similar to BLP, capturing integrity aspects of access control.
- Integrity ≡ prevent unauthorized information
- Integrity levels are assigned to subjects and objects
- **Simple Integrity Property**

  No write-up: If subject s can modify (alter) object o, then $f_S(s) \geq f_O(o)$.

- **Integrity $*$ - Property**

  If subject *s* can read (observe) object *o*, then *s* can have write access to some other object *o'* only if $f_O(o) \geq f_O(o')$.

Biba Model (more policies)

- The previous two policies are the dual of the mandatory BLP policies. Integrity labels are static (tranquility).
- Low watermark policies automatically adjust integrity levels, like in the Chinese Wall model.
- **Subject Low Watermark Policy**

  Subject *s* can read (observe) an object *o* at any integrity level. The new integrity level of *s* is g.l.b.($f_S(s)$, $f_O(o)$.

- **Object Low Watermark Policy**

  Subject *s* can modify (alter) an object *o* at any integrity level. The new integrity level of *o* is g.l.b.($f_S(s)$, $f_O(o)$.

Biba Model (policies for invoke)

- Invoke: access operation between subjects
- **Invoke Property**

  Subject s_1 can invoke subject s_2 only if $f_S(s_1) \geq f_S(s_2)$.

- Adds to the first two mandatory integrity policies. A `dirty' subject $s_1$ cannot touch a `clean' object indirectly by invoking s_2.
- **Ring Property**

  Subject s_1 can read objects at all integrity levels, modify objects o with $f_S(s_1) \geq f_O(o)$, and invoke a subject s_2 only if $f_S(s_1) \leq f_O(s_2)$.

- A `dirty' subject $s_1$ can invoke a `clean' tool s_2 to touch a `clean' object. The ring property is the opposite of the invoke property!

Clark-Wilson Model

- This model attempts to capture security requirements of commercial applications. `Military' and `commercial' are shorthand for different ways of using computers.
- Emphasis on integrity
  - internal consistency: properties of the internal state of a system
  - external consistency: relation of the internal state of a system to the outside world.
- Mechanisms for maintaining integrity
  - well-formed transactions
  - separation of duties

Clark-Wilson: Access Control

- Subjects and objects are `labeled' with programs.
- Programs serve as an intermediate layer between subjects and objects.
- Access control:
  - define the access operations (transformation procedures) that can be performed on each data item (data types).
  - define the access operations that can be performed by subjects (roles).
- Note the difference between a general purpose operating system (BLP) and an application oriented IT system (Clark-Wilson).

Clark-Wilson: Certification Rules

- Security properties are partly defined through five certification rules, suggesting the checks that should be conducted so that the security policy is consistent with the application requirements.
  - IVPs (initial verification procedures) must ensure that all CDIs (constrained data items) are in a valid state when the IVP is run.
  - TPs (transformation procedures) must be certified to be valid, i.e. valid CDIs must always be transformed into valid CDIs. Each TP is certified to access a specific set of CDIs.
  - The access rules must satisfy any separation of duties requirements.
  - All TPs must write to an append-only log.
  - Any TP that takes an UDI (unconstrained data item) as input must either convert the UDI into a CDI or reject the UDI and perform no transformation at all.

Clark-Wilson: Enforcement Rules

- Four enforcement rules describe the security mechanisms within the computer system that should enforce the security policy. These rules are similar to discretionary access control in BLP.
  - The system must maintain and protect the list of entries (TPi:CDIa,CDIb,...)} giving the CDIs the TP is certified to access.
  - The system must maintain and protect the list of entries (UserID,TPi:CDIa,CDIb,...)} specifying the TPs users can execute.
  - The system must authenticate each user requesting to execute a TP.
  - \Only a subject that may certify an access rule for a TP may modify the respective entry in the list. This subject must not have execute rights on that TP.
- Clark-Wilson is less formal than BLP but much more than an access control model.

Information Flow Models

- Similar framework as BLP: objects are labeled with security classes (form a lattice), information may flow upwards only.
- Information flow is described in terms of conditional entropy (equivocation $\rightarrow$ information theory)
- Information flows from x to y if we learn something about x by observing y:
  - explicit information flow: $y := x$
  - implicit information flow: IF $x=0$ THEN $y:=1$
  - covert channels
- Proving security is now undecidable.

Exercises

- BLP does not specify policies for changing access rights. Which policies would you suggest?
- Should the star-property in the Chinese Wall model refer to current read access only or to any past read access?
- Give examples for application areas where a Biba policy with static integrity labels, a policy with dynamically changing integrity labels, or the ring-property is appropriate.
- Can you use BLP and Biba to model confidentiality and integrity simultaneously? Can you use the same labels for both policies?
- Develop a security model for documents, which are declassified after 30 years.
- In a medical information system that controls access to patient records and prescriptions, doctors may read and write patient records and prescriptions, nurses may read and write prescriptions only but should learn nothing about the contents of patient records. How can you capture this policy in a lattice model that prevents information flow from patient records to prescriptions? In your opinion, which security model is most appropriate for this policy?

Further reading

- Sterne, D.F., *On the Buzzword `Security Policy'*, Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy, pages 219-230, 1991
- Bell, D. and LaPadula, L., *MITRE Technical Report 2547 (Secure Computer System): Volume II*, Journal of Computer Security, vol. 4, no. 2/3, pages 239-263, 1996
- Clark, D.R. and Wilson, D.R., *A Comparison of Commercial and Military Computer Security Policies*, Proceedings of the 1987 IEEE Symposium on Security and Privacy, pages 184-194, 1987
- Goguen, J.A. and Meseguer, J., *Security Policies and Security Models*, Proceedings of the 1982 IEEE Symposium on Security and Privacy, pages 11-20, 1982