The Bell-LaPadula Model

CSM27 Computer Security

Dr Hans Georg Schaathun

University of Surrey

Autumn 2008 – Week 6
Outline

1. The session
2. Finite Automata
3. Bell-LaPadula
4. Security Properties
5. Limitations
6. Multics
7. Conclusion
Session objectives

- Be able to use the principle of finite automata to describe security models.
- Understand the confidentiality policy of Bell-LaPadula
- Understand the limitations of Bell-LaPadula
Outline

1. The session
2. Finite Automata
3. Bell-LaPadula
4. Security Properties
5. Limitations
6. Multics
7. Conclusion
A finite automata

- **state-machine** \(\approx\) automata
- A set of **states**, \(Q\)
- An **input alphabet** \(\Sigma\)
  - labels for the state **transitions**
- **initial state** \(q_0 \in Q\)
- **accepting states** \(A \subset Q\)
- **transition function** \(\delta : Q \times \Sigma \rightarrow Q\)
  - equivalent to the edges (arrows)
A finite automata

- A state can be **good** or **bad**
  - **secure** or **insecure**
- Transitions from good to bad states are dangerous.
- Two criteria
  - Start state be secure
  - No transition from secure to insecure
A state can be **good** or **bad**
- **secure** or **insecure**

Transitions from good to bad states are dangerous.

Two criteria
- Start state be secure
- No transition from secure to insecure
A finite automata

- A state can be **good** or **bad**
  - **secure** or **insecure**
- Transitions from good to bad states are dangerous.
- Two criteria
  - Start state be secure
  - No transition from secure to insecure
Outline

1. The session
2. Finite Automata
3. Bell-LaPadula
4. Security Properties
5. Limitations
6. Multics
7. Conclusion
The principle of an automata model

1. Describe all secure states
2. Describe transitions from secure states
3. Prove that no transition leads from secure to insecure

- If this is possible, the system is provably secure.
- Bell-LaPadula is one description of secure states.

Similar principles apply to e.g. database development
- Database has to be maintained in a consistent state
- No operation (transition) allowed to bring the database to an inconsistent state
Elements of Access Control

- a set of subjects $S$
- a set of objects $O$
- set of access operations $A = \{\text{execute, read, append, write}\}$
- A set of security levels $L$, with a partial ordering $\leq$
The State Set

- A state: \((b, M, f)\), includes
  - Access operations currently in use \(b\)
    - List of tuples \((s, o, a)\), \(s \in S\), \(o \in O\), \(a \in A\).
- Access permission matrix
  - \(M = (M_{s,o})_{s \in S, o \in O}\), where \(M_{s,o} \subset A\)
- Clearance and classification \(f = (f_S, f_C, f_O)\)
  - \(f_S : S \rightarrow L\) maximal security level of a subject
  - \(f_C : S \rightarrow L\) current security level of a subject (\(f_C \leq f_S\))
  - \(f_O : O \rightarrow L\) classification of an object
Outline

1. The session
2. Finite Automata
3. Bell-LaPadula
4. Security Properties
5. Limitations
6. Multics
7. Conclusion
A state \((b, M, f)\) satisfies the SS-property if
- \(\forall (s, o, a) \in b, \text{ such that } a \in \{\text{read, write}\}\)
- \(f_O(o) \leq f_S(s)\)

I.e. a subject can only observe objects of lower classification
What about write access?

- What policy do we need for write access?
  - Integrity: no write-up (to higher security levels)
  - Confidentiality: no write-down (to lower security levels)
  - Bell-LaPadula concerns confidentiality
  - Subject must not transmit messages to subjects at lower levels
  - Current security level allows communications
    - A subject has to be downgraded to send messages
  - Because subjects are computer programs
    - they can be made to forget their knowledge when downgraded
What about write access?

- What policy do we need for write access?
- Integrity: no write-up (to higher security levels)
- Confidentiality: no write-down (to lower security levels)

Bell-LaPadula concerns confidentiality
- Subject must not transmit messages to subjects at lower levels
- Current security level allows communications
  - A subject has to be downgraded to send messages
- Because subjects are computer programs
  - they can be made to forget their knowledge when downgraded
What about write access?

- What policy do we need for write access?
- Integrity: no write-up (to higher security levels)
- **Confidentiality:** no write-down (to lower security levels)
- Bell-LaPadula concerns confidentiality
  - Subject must not transmit messages to subjects at lower levels
  - **Current security level** allows communications
    - A subject has to be downgraded to send messages
  - Because subjects are computer programs
    - they can be made to forget their knowledge when downgraded
What about write access?

- What policy do we need for write access?
- Integrity: no write-up (to higher security levels)
- Confidentiality: no write-down (to lower security levels)
- Bell-LaPadula concerns confidentiality
- Subject must not transmit messages to subjects at lower levels
  - Current security level allows communications
    - A subject has to be downgraded to send messages
  - Because subjects are computer programs
    - they can be made to forget their knowledge when downgraded
What about write access?

- What policy do we need for write access?
- Integrity: no write-up (to higher security levels)
- Confidentiality: no write-down (to lower security levels)
- Bell-LaPadula concerns confidentiality
- Subject must not transmit messages to subjects at lower levels
- **Current security level** allows communications
  - A subject has to be downgraded to send messages
- Because subjects are computer programs
  - they can be made to forget their knowledge when downgraded
What about write access?

- What policy do we need for write access?
- Integrity: no write-up (to higher security levels)
- Confidentiality: no write-down (to lower security levels)
- Bell-LaPadula concerns confidentiality
- Subject must not transmit messages to subjects at lower levels
- **Current security level** allows communications
  - A subject has to be downgraded to send messages
- Because subjects are computer programs
  - they can be made to forget their knowledge when downgraded
*-property

A state \((b, M, f)\) satisfies the *-property if

- \(\forall(s, o, a) \in b\), such that \(a \in \{\text{append, write}\}\)
- \(f_C(s) \leq f_O(o)\)

and

- if \(\exists(s, o, a) \in b\) where \(a \in \{\text{append, write}\}\),
- then \(\forall o', a' \in \{\text{read, write}\}\), such that \((s, o', a') \in b\)
- \(f_O(o') \leq f_O(o)\)

i.e. a subject can only alter objects of higher classification,
and cannot read a high-level object while writing to a low-level object.
Security Properties

Discretionary Security Property

- Previous security properties provide **Mandatory Access Control**
  - i.e. a centrally defined access policy
- The security levels are defined by a central policy
- **Discretionary Access Control** (DAC) decentralises the control
- The access control matrix $M$ allows DAC in Bell-LaPadula

A state $(b, M, f)$ satisfies the DS-property if
- $\forall (s, o, a) \in b$
- $a \in M_{s,o}$. 
Discretionary Security Property

- Previous security properties provide Mandatory Access Control
  - i.e. a centrally defined access policy
- The security levels are defined by a central policy
- Discretionary Access Control (DAC) decentralises the control
- The access control matrix $M$ allows DAC in Bell-LaPadula

A state $(b, M, f)$ satisfies the DS-property if

- $\forall (s, o, a) \in b$
- $a \in M_{s,o}$. 
Discretionary Security Property

- Previous security properties provide **Mandatory Access Control**
  - i.e. a centrally defined access policy
- The security levels are defined by a central policy
- **Discretionary Access Control** (DAC) decentralises the control
- The access control matrix $M$ allows DAC in Bell-LaPadula

- A state $(b, M, f)$ satisfies the DS-property if
  - $\forall (s, o, a) \in b$
  - $a \in M_{s,o}$. 
Discretionary Security Property

- Previous security properties provide **Mandatory Access Control**
  - i.e. a centrally defined access policy
- The security levels are defined by a central policy
- **Discretionary Access Control** (DAC) decentralises the control
- The access control matrix \( M \) allows DAC in Bell-LaPadula

A state \( (b, M, f) \) satisfies the DS-property if
- \( \forall (s, o, a) \in b \)
- \( a \in M_{s,o} \).
Outline

1. The session
2. Finite Automata
3. Bell-LaPadula
4. Security Properties
5. Limitations
6. Multics
7. Conclusion
The Criticism of McLean

What happens if we . . .
- downgrade all subjects to lowest security level
- downgrade all objects to lowest security level
- enter all access rights in the ACM $M$

Is the system secure?
- It satisfies every security property of BLP!
The Criticism of McLean

- What happens if we . . .
  - downgrade all subjects to lowest security level
  - downgrade all objects to lowest security level
  - enter all access rights in the ACM $M$

- Is the system secure?
- It satisfies every security property of BLP!
The sides of the conflict

A system which can be brought to a state with no restrictions cannot be secure.

McLean

This is application dependent. If the users need it, it should be possible. Otherwise it should not be implemented.

Bell
McLean’s scenario is really out of scope for BLP
BLP considered *tranquil* systems,
  where permissions do not change
Either a system or an operation may be *tranquil*
  A tranquil operation does not change access rights.
  A tranquil system has no non-tranquil operations.
Tranquility is a particular concern when
  operation tries to remove an access right currently in use
  How should this be resolved?
Covert Channels

- **Low-level subject** $s_l$ creates object $o$
- High-level accomplice $s_h$ either
  - reclassifies $o$ to its own level (Message 1)
  - leaves $o$ unchanged (Message 0)
- $s_l$ tries to access $o$, which is either
  - success (Message 0)
  - access denied (Message 1)
- One bit of information is transmitted $s_h \rightarrow s_l$
Covert Channels

- Low-level subject $s_l$ creates object $o$
- High-level accomplice $s_h$ either
  - reclassifies $o$ to its own level (Message 1)
  - leaves $o$ unchanged (Message 0)
- $s_l$ tries to access $o$, which is either
  - success (Message 0)
  - access denied (Message 1)
- One bit of information is transmitted $s_h \rightarrow s_l$
Covert Channels

- Low-level subject $s_l$ creates object $o$
- High-level accomplice $s_h$ either
  - reclassifies $o$ to its own level (Message 1)
  - leaves $o$ unchanged (Message 0)
- $s_l$ tries to access $o$, which is either
  - success (Message 0)
  - access denied (Message 1)
- One bit of information is transmitted $s_h \rightarrow s_l$
Covert Channels

- Low-level subject $s_l$ creates object $o$
- High-level accomplice $s_h$ either
  - reclassifies $o$ to its own level (Message 1)
  - leaves $o$ unchanged (Message 0)
- $s_l$ tries to access $o$, which is either
  - success (Message 0)
  - access denied (Message 1)
- One bit of information is transmitted $s_h \rightarrow s_l$
Limitations

- BLP’s concern is confidentiality
  - limits the access and sharing of information
  - no integrity policy
  - no availability policy
- BLP assumes a fixed rights
  - assumes tranquility
  - no model for access management
  - no model for policy making
- Allows Covert Channels
Outline

1. The session
2. Finite Automata
3. Bell-LaPadula
4. Security Properties
5. Limitations
6. Multics
7. Conclusion
What and when was Multics?

- Massive research project in early 70-s
- Objective: Secure, reliable, etc multiuser OS
  - i.e. Multics
- The Bell-LaPadula model was a result of the research
- The ambitions made Multics too heavy-weight for most
  - Unix is a spin-off by some project members
  - simpler and more user-friendly,
Objects

- Objects: memory segments, I/O devices, et c.
  - hierarchically organised in directory tree
  - information stored in parent directory
    - Access Control List (ACL) representing $M$
    - Security level (classification) $f_O$
- Access to an object traverses path from root
  - Access requires access to all ancestors
  - Low-level object in high-level directory makes little sense
- Compatibility: Every object has security-level dominating that of the parent directory
Objects

- Objects: memory segments, I/O devices, et c.
  - hierarchically organised in directory tree
  - information stored in parent directory
    - Access Control List (ACL) representing $M$
    - Security level (classification) $f_O$

- Access to an object traverses path from root
  - Access requires access to all ancestors
  - Low-level object in high-level directory makes little sense

- *Compatibility*: Every object has security-level dominating that of the parent directory
Multics

Objects

- Objects: memory segments, I/O devices, et c.
  - hierarchically organised in directory tree
  - information stored in parent directory
    - Access Control List (ACL) representing $M$
    - Security level (classification) $f_O$

- Access to an object traverses path from root
  - Access requires access to all ancestors
  - Low-level object in high-level directory makes little sense

- **Compatibility**: Every object has security-level dominating that of the parent directory
Subjects

- Subject = process
  - Descriptor segment describes the process, its rights, and its accesses
  - Segment Descriptor Word (SDW) [representing $b$] for each object currently accessed
    - segment-id  |  ptr  |  r: on  |  e: off  |  w: one  

- $f_C$ : Current-level table
- $f_S$ : Process-level table
- active segment table : which processes are active
Translating policies

- Every parameter of the BLP state
  - has a representation in Multics data fields
- Security policies can be rephrased,
  - referring to Multics data fields
Translating policies

- Every parameter of the BLP state
  - has a representation in Multics data fields
- Security policies can be rephrased,
  - referring to Multics data fields
Example: The SS-property
Kernel primitives

- Kernel primitives ~ state transitions
  - release_read
  - give_read
  - create_object
  - delete_object
  - revoke_read
  - change_subject_current_security_level

- Ideally CPU instructions and OS kernel primitives match
Kernel primitive: `get_read`

- Example: `get_read`
  - Subject `s` wants to read object `o`
  - so `s` asks OS to add `(s, o, read)` to `b`
- The OS has to check this with the security policy, i.e.
  - The ACL of `o` includes `(s, read)`
  - \( f_O(o) \leq f_S(s) \)
  - Either
    - \( f_O(o) \leq f_C(s) \); or
    - Subject `s` is trusted.

- Access permitted if and only if all three conditions are met.
Multics and security models

- The state-machine is an effective model of a computer system
- Bell-LaPadula describes secure states and transitions
- If all transitions (and starting state) are secure, the system has to be secure
- In multics,
  - data-fields correspond to state parameters
  - kernel primitives correspond to transitions
Exercise sheet

- Write a short essay stating your position in the Bell vs McLean debate.

It is helpful to address as many of the strengths and weaknesses of BLP as possible, in order to build an argument for your view. Suggested length $\frac{1}{2}$-2 pages. Longer is not necessarily better.