topic 6

Security policy models

A bit of theory- Integrity models

Reading: chapter 6

1

Requirements of Policies

1. Users will not write their own programs, but will use existing production programs and databases.
2. Programmers will develop and test programs on a non-production system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
3. A special process must be followed to install a program from the development system onto the production system.
4. The special process in requirement 3 must be controlled and audited.
5. The managers and auditors must have access to both the system state and the system logs that are generated.

2

Bell-LaPadula Confidentiality Model

- When is it OK to release information?
- Two Properties (with silly names)
  - » Simple security property
    - – A subject S may read object O only if $C(O) \leq C(S)$
  - » *-Property
    - – A subject S with read access to O may write object P only if $C(O) \leq C(P)$
- In words,
  - » You may only *read below* your classification and only *write above* your classification

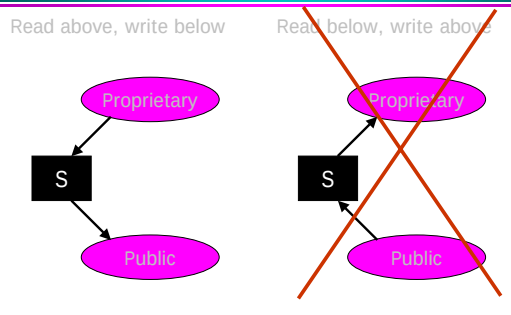
3

Biba Integrity Model

- Rules that preserve integrity of information
- Two Properties (with silly names)
  - » Simple integrity property
    - – A subject S may write object O only if $C(S) \geq C(O)$
    - (Only trust S to modify O if S has higher rank …)
  - » *-Property
    - – A subject S with read access to O may read object P only if $C(O) \geq C(P)$
    - (Only move info from O to P if O is more trusted than P)
- In words,
  - » You may only *write below* your classification and only *read above* your classification

4

Picture: Integrity



5

Problem: Models are contradictory

- Bell-LaPadula Confidentiality
  - » Read down, write up
- Biba Integrity
  - » Read up, write down
- Want both confidentiality and integrity
  - » Only way to satisfy both models is only allow read and write at same classification

In reality: Bell-LaPadula used more than Biba model
Example: Common Criteria

6

Other policy concepts

- Separation of duty
  - » If amount is over $10,000, check is only valid if signed by two authorized people
  - » Two people must be *different*
  - » Policy involves role membership and $\neq$
- Chinese Wall Policy
  - » Lawyers L1, L2 in Firm F are experts in banking
  - » If bank B1 sues bank B2,
    - – L1 and L2 can each work for either B1 or B2
    - – No lawyer can work for opposite sides in any case
  - » Permission depends on use of other permissions

7

So..

- In the previous slides, we saw that an access matrix could be altered.
- Leads to fundamental questions:
  - » What is “secure”?
  - » What policy defines “secure”?
  - » How do we know a state transition from a secure state results in a secure state?

8

The Safety Question

- Use an access control matrix to state policy.
  - » No *copy* or *grant* or *own* rights, attenuation of privilege does not apply.
- A right r is *leaked* if it is added to a cell of the matrix that doesn’t already contain r.
- If the system can never leak the right r, the system is called *safe with respect to the right r*.
  - » If the system can leak the right r, it is unsafe.

9

The Safety Question

- Safety of the model doesn’t guarantee security of the system.
  - » A secure system corresponds to a model safe with respect to all rights.
  - » Not vice versa
- Why Not??
  - » The model of a safe system creates a secure design
  - » The secure design does not guarantee a correct implementation.

10

The Safety Question

Is there a way to determine whether a given protection system with a defined initial state is safe with respect to a right *r* ?

it is undecidable in the general case.

Can the safety of a particular system with specific rules be established or disproved?

Sometimes.

11

Biba Integrity Model

Basis for all 3 models:

- Set of subjects S, objects O, integrity levels I, relation $\leq \subseteq I \times I$ holding when second dominates first
- *min*: $I \times I \rightarrow I$ returns lesser of integrity levels
- *i*: $S \cup O \rightarrow I$ gives integrity level of entity
- r: $S \times O$ means $s \in S$ can read $o \in O$
- w, x defined similarly

12

Intuition for Integrity Levels

- The higher the level, the more confidence
  - » That a program will execute correctly
  - » That data is accurate and/or reliable
- Note relationship between integrity and trustworthiness
- Important point: *integrity levels are* ***not*** *security levels*

13

Information Transfer Path

- An *information transfer path* is a sequence of objects $o_1, ..., o_{n+1}$ and corresponding sequence of subjects $s_1, ..., s_n$ such that $s_i \underline{r} o_i$ and $s_i \underline{w} o_{i+1}$ for all i, $1 \le i \le n$.
- Idea: information can flow from o_1 to o_{n+1} along this path by successive reads and writes

14

Low-Water-Mark Policy

- Idea: when s reads o, $i(s) = min(i(s), i(o))$; s can only write objects at lower levels
- Rules
  1. $s \in S$ can write to $o \in O$ if and only if $i(o) \le i(s)$.
  2. If $s \in S$ reads $o \in O$, then $i'(s) = min(i(s), i(o))$, where $i'(s)$ is the subject's integrity level after the read.
  3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \le i(s_1)$.

15

Information Flow and Model

- If there is information transfer path from $o_1 \in O$ to $o_{n+1} \in O$, enforcement of low-water-mark policy requires $i(o_{n+1}) \le i(o_1)$ for all $n > 1$.
  - » Idea of proof: Assume information transfer path exists between o_1 and o_{n+1}. Assume that each read and write was performed in the order of the indices of the vertices. By induction, the integrity level for each subject is the minimum of the integrity levels for all objects preceding it in path, so $i(s_n) \le i(o_1)$. As nth write succeeds, $i(o_{n+1}) \le i(s_n)$. Hence $i(o_{n+1}) \le i(o_1)$.

16

Problems

- Subjects' integrity levels decrease as system runs
  - » Soon no subject will be able to access objects at high integrity levels
- Alternative: change object levels rather than subject levels
  - » Soon all objects will be at the lowest integrity level
- Crux of problem is model prevents indirect modification
  - » Because subject levels lowered when subject reads from low-integrity object

17

Ring Policy

- Idea: subject integrity levels static
- Rules
  1. $s \in S$ can write to $o \in O$ if and only if $i(o) \le i(s)$.
  2. Any subject can read any object.
  3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \le i(s_1)$.
- Eliminates indirect modification problem
- Same information flow result holds

18

Strict Integrity Policy

- Similar to Bell-LaPadula model
  1. $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$
  2. $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$
  3. $s_1 \in S$ can execute $s_2 \in S$ iff $i(s_2) \leq i(s_1)$
- Add compartments and discretionary controls to get full dual of Bell-LaPadula model
- Information flow result holds
  - » Different proof, though
- Term "Biba Model" refers to this

19

LOCUS and Biba

- Goal: prevent untrusted software from altering data or other software
- Approach: make levels of trust explicit
  - » *credibility rating* based on estimate of software's trustworthiness (0 untrusted, *n* highly trusted)
  - » *trusted file systems* contain software with a single credibility level
  - » Process has *risk level* or highest credibility level at which process can execute
  - » Must use *run-untrusted* command to run software at lower credibility level

20

Integrity Matrix Model

- Lipner proposed this as first realistic commercial model
- Combines Bell-LaPadula, Biba models to obtain model conforming to requirements
- Do it in two steps
  - » Bell-LaPadula component first
  - » Add in Biba component

21

Bell-LaPadula Clearances

- 2 security clearances/classifications
  - » AM (Audit Manager): system audit, management functions
  - » SL (System Low): any process can read at this level

22

Bell-LaPadula Categories

- 5 categories
  - » D (Development): production programs in development but not yet in use
  - » PC (Production Code): production processes, programs
  - » PD (Production Data): data covered by integrity policy
  - » SD (System Development): system programs in development but not yet in use
  - » T (Software Tools): programs on production system not related to protected data

23

Users and Security Levels

Subjects	Security Level
Ordinary users	(SL, { PC, PD })
Application developers	(SL, { D, T })
System programmers	(SL, { SD, T })
System managers and auditors	(AM, { D, OC, OD, SD, T })
System controllers	(SL, {D, PC, PD, SD, T}) and downgrade privilege

24

Objects and Classifications

Objects	Security Level
Development code/test data	(SL, { D, T })
Production code	(SL, { PC })
Production data	(SL, { PC, PD })
Software tools	(SL, { T })
System programs	(SL, ∅)
System programs in modification	(SL, { SD, T })
System and application logs	(AM, { *appropriate* })

25

Ideas

- Ordinary users can execute (read) production code but cannot alter it
- Ordinary users can alter and read production data
- System managers need access to all logs but cannot change levels of objects
- System controllers need to install code (hence downgrade capability)
- Logs are append only, so must dominate subjects writing them

26

Check Requirements

1. Users have no access to T, so cannot write their own programs
2. Applications programmers have no access to PD, so cannot access production data; if needed, it must be put into D, requiring the system controller to intervene
3. Installing a program requires downgrade procedure (from D to PC), so only system controllers can do it

27

More Requirements

4. Control: only system controllers can downgrade; audit: any such downgrading must be altered
5. System management and audit users are in AM and so have access to system styate and logs

28

Problem

- Too inflexible
  - » System managers cannot run programs for repairing inconsistent or erroneous production database
    - – System managers at AM, production data at SL
- So add more …

29

Adding Biba

- 3 integrity classifications
  - » ISP(System Program): for system programs
  - » IO (Operational): production programs, development software
  - » ISL (System Low): users get this on log in
- 2 integrity categories
  - » ID (Development): development entities
  - » IP (Production): production entities

30

Simplify Bell-LaPadula

- Reduce security categories to 3:
  - » SP (Production): production code, data
  - » SD (Development): same as D
  - » SSD (System Development): same as old SD

31

Users and Levels

Subjects	Security Level	Integrity Level
Ordinary users	(SL, { SP })	(ISL, { IP })
Application developers	(SL, { SD })	(ISL, { ID })
System programmers	(SL, { SSD })	(ISL, { ID })
System managers and auditors	(AM, { SP, SD, SSD })	(ISL, { IP, ID})
System controllers	(SL, { SP, SD }) and downgrade privilege	(ISP, { IP, ID})
Repair	(SL, { SP })	(ISL, { IP })

32

Objects and Classifications

Objects	Security Level	Integrity Level
Development code/test data	(SL, { SD })	(ISL, { IP})
Production code	(SL, { SP })	(IO, { IP })
Production data	(SL, { SP })	(ISL, { IP })
Software tools	(SL, ∅)	(IO, { ID })
System programs	(SL, ∅)	(ISP, { IP, ID })
System programs in modification	(SL, { SSD })	(ISL, { ID })
System and application logs	(AM, { *appropriate* })	(ISL, ∅)
Repair	(SL, {SP})	(ISL, { IP })

33

Ideas

- Security clearances of subjects same as without integrity levels
- Ordinary users need to modify production data, so ordinary users must have write access to integrity category IP
- Ordinary users must be able to write production data but not production code; integrity classes allow this
  - » Note writing constraints removed from security classes

34

Key Points

- Integrity policies deal with trust
  - » As trust is hard to quantify, these policies are hard to evaluate completely
  - » Look for assumptions and trusted users to find possible weak points in their implementation
- Biba, Lipner based on multilevel integrity
- Clark-Wilson focuses on separation of duty and transactions

35