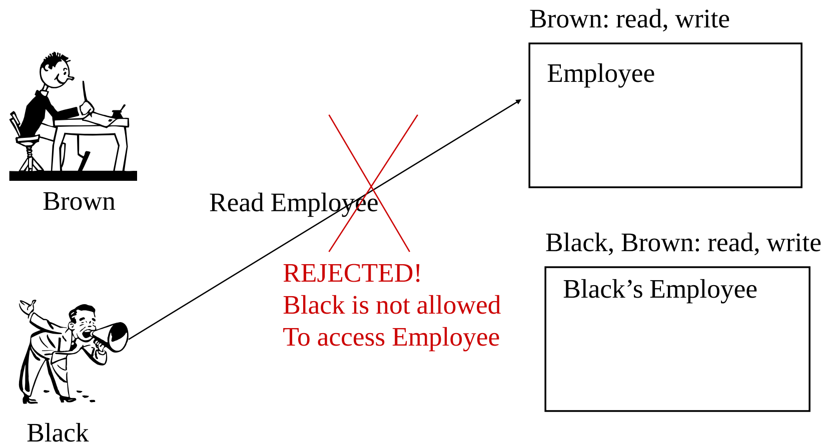
Mandatory Access Control

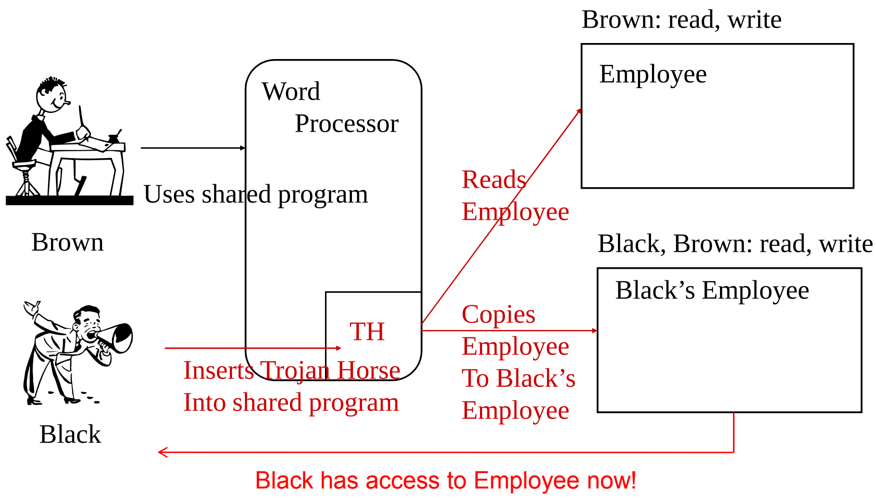
1

DAC and Trojan Horse



2

DAC and Trojan Horse



3

Mandatory Access Control (MAC)

- Security *level of object* (security label): Sensitivity of object
- Security *level of subject* (security class): user's clearance
  - E.g. Top Secret > Secret > Confidential > Unclassified
- MAC specifies the access that subjects have to objects based on the subjects and objects classification
- This type of security has also been referred to as *multilevel* security

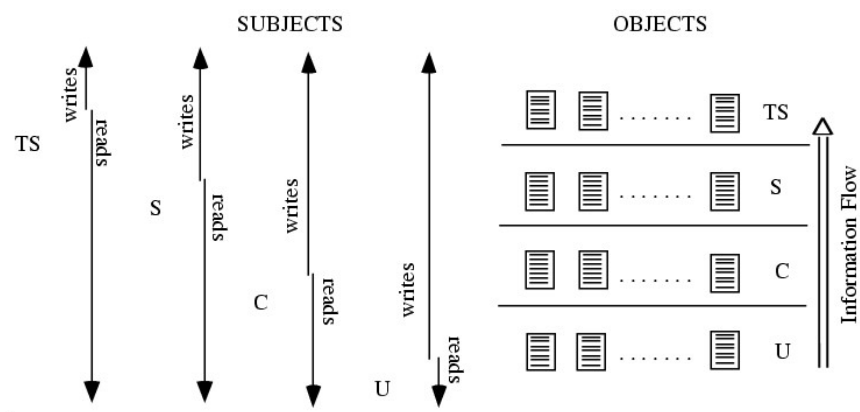
4

Mandatory Access Control (MAC)

- Controlling information flow (Bell-LaPadulla properties BLP):
  - No READ UP: Subject clearance ≥ object security
  - No WRITE DOWN (*-property): Subject clearance ≤ object security
  - Prevent information in high level objects from flowing to low level subjects
  - Tranquility property: The classification of a resource cannot be changed while the resource is in use by any user of the system
- Necessary but not sufficient conditions
- May still have problems – covert channel
  - Indirect means by which info at higher levels passed to lower levels

5

MAC – Controlling Information Flow



6

MAC – Problems?

- Write-up allows destruction of more secure info
  - Limit to same level; disable write-up
- Write-up means cannot send info to lower-level subjects
  - Subject can sign in at lower level
  - Prevent malicious programs from leaking secrets
  - Users are trusted, not programs
- Hierarchy of security levels is too restrictive
  - Consider the notion of "need-to-know"
    - In military applications, someone cleared for TOP SECRET information on OPERATION X may not even need to know about UNCLASSIFED documents on OPERATION Y
  - Lattice of security labels

7

Lattice of Security Labels

- Security level is (*clearance*, *category set*)
- Examples
  - (Top Secret, { NUC, EUR, ASI })
  - (Confidential, { EUR, ASI })
  - (Secret, { NUC, ASI })

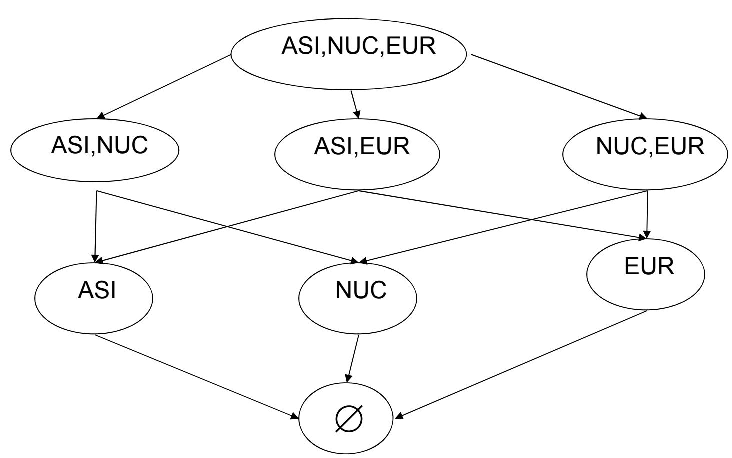
8

Levels and Lattices

- (A, C) *dom* (A', C') iff $A' \leq A$ and $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
  - (Secret, {NUC, EUR}) *dom* (Confidential,{NUC, EUR})
  - (Top Secret, {NUC}) $\neg$*dom* (Confidential, {EUR})
  - (Secret, {NUC}) $\neg$*dom* (Confidential,{NUC, EUR})
- Let C be set of classifications, K set of categories. Set of security levels $L = C \times K$, *dom* form lattice
  - *Partially ordered set*
  - *Any pair of elements*
    - *Has a greatest lower bound*
    - *Has a least upper bound*

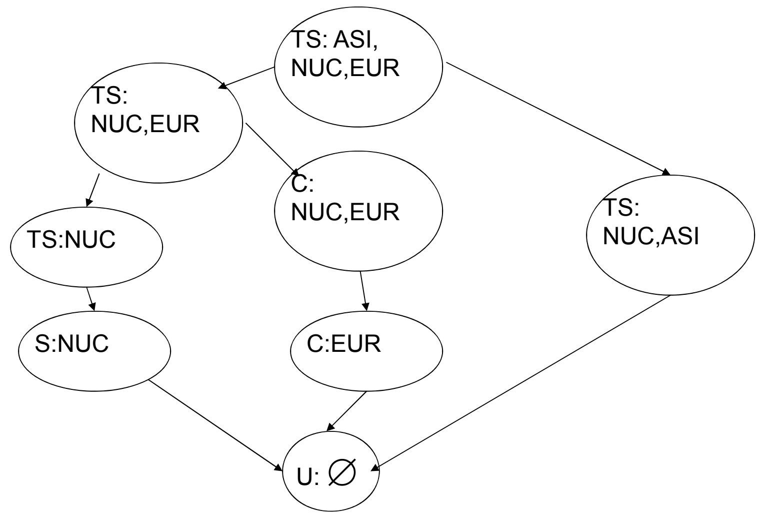
9

Example Lattice



10

Subset Lattice



11

Why Apply MAC to DB?

- Data can be viewed as sensitive for many different reasons. Examples:
  - personal and private matters or communications, professional trade secrets
  - company plans for marketing or finance
  - military information, or government plans
- Such data is often mixed with other, less sensitive information that is legitimately needed by diverse users
- Restricting access to entire tables or segregating sensitive data into separate databases can create a working environment that is costly in hardware, software, user time, and administration.

12

Multilevel Relational (MLR) Model

- The multilevel relational (MLR for short) model results from the application of the BLP model to relational databases
- Several issues
  - Granularity: to which element do we apply the classification?
  - Integrity constraints

13

Traditional Relational Model

Standard relational model – each relation is characterized by two components

- A state-invariant *relation schema* R(A1,…., An) where Ai is an attribute over some domain Di
- A state-dependent *relation* over R composed of distinct tuples of the form (a1,…, an), where each ai is a value in domain Di

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	40
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	30
612-67-4134	Madayan	35	8	10	40

14

Relational Model – keys and FD

- Functional dependencies
  - Let *R* be a relation and let *X* and *Y* be attribute sets, both subsets of the attribute set of R
  - we say that *X* **functionally determines** *Y* if and only if no two tuples may exist in *R* with the same value for *X* but different values for *Y*
- Primary Keys *(entity integrity property)*
  - the primary key uniquely identifies each tuple in the relation
  - A primary key cannot contain attributes with null values
  - A relation cannot contain two tuples with the same value for the primary key

15

Example

- Consider relation Hourly_Emps:
  - Hourly_Emps (<u>*ssn*</u>*, name, lot, rating, hrly_wages*, *hrs_worked*)

- FDs S $\rightarrow$ SNLRWH
  - *ssn* is the key
- FDs give more detail than the mere assertion of a key
  - *rating* determines *hrly_wages*
  - R $\rightarrow$ W

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	40
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	30
612-67-4134	Madayan	35	8	10	40

16

MLR Model

- Given a relation, an access class can be associated with:
  - The entire relation
  - Each tuple in the relation
    - This is the common choice in commercial systems
  - Each attribute value of each tuple in the relation
    - In the remainder we consider this case
      - Toward a Multilevel Secure Relational Data Model. Proc 1991 ACM Int'l. Conf. on Management of Data (SIGMOD), 50-59.

17

Multilevel (ML) relations

A ML relation is characterized by two components

- A state-invariant *relation scheme*

  R(A1,C1,…., An,Cn, TC) where:
  - Ai is an attribute over some domain Di
  - Ci is a classification attribute for Ai; its domain is the set of access classes that can be associated with values of Ai
  - TC is the classification attribute of the tuple
- A set of state-dependent *relation instances Rc* over R for each access class in the access class lattice. Each instance *Rc* is composed of distinct tuples of the form (a1,c1,…, an,cn, tc), where:
  - ai is a value in domain Di
  - ci is the access class for ai
  - tc is the access class of the tuple determined as the least upper bound of all ci in the tuple
  - Classification attributes cannot assume null values

18

ML relations - example

Vessel (AK)	Objective	Destination	TC
Micra U	Shipping U	Moon U	U
Vision U	Spying U	Saturn U	U
Avenger C	Spying C	Mars C	C
Logos S	Shipping S	Venus S	S

ML relations - instances

- A given relation may thus have instances at different access classes
- The relation instance at class c contains all data that are visible to subjects at level c
  - It contains all data whose access classes are *dominated* by c
  - All elements with access classes higher than c, or incomparable, are masked by null values
  - Sometimes, to avoid signaling channels, fictitious values (called *cover story values*) can be used

ML relations - example

Vessel (AK)	Objective	Destination	TC
Micra U	Shipping U	Moon U	U
Vision U	Spying U	Saturn U	U
Avenger C	Spying C	Mars C	C
Logos S	Shipping S	Venus S	S

- Level U users see first 2 tuples
- Level C users see first 3 tuples
- Level S users see all tuples

21

MLS Model

- Entity integrity rule
  - All attributes that are members of the *apparent key* must not be null (i.e., $A_i \in AK \Rightarrow t[A_i] \neq NULL$)
  - All attributes of AK must have the same security classification within each individual tuple (i.e., $A_i, A_j \in AK \Rightarrow t[C_i] = t[C_j]$)
  - For each tuple, the access class associated with the non-key attributes must dominate the access class of the primary key (i.e., $A_i \notin AK \Rightarrow t[C_i] \geq t[C_{AK}]$).
- Null integrity
  - Nulls are classified at the level of the key
  - One tuple does not subsume another (null values subsumed by non-null values)
- Inter-Instance Integrity
  - User can only see portion of relation for which he/she is cleared
  - Data not cleared is set to null
  - Eliminate subsumed tuples

22

MLS Model - Example

S-user view:

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U
Voyager U	Spying S	Mars S	S

U-user view:

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U
Voyager U	Null U	Null U	U

23

ML relations – keys and polyinstantiation

- In the standard relational model, each tuple is uniquely identified, by the values of its key attributes
- When access classes are introduced, there may be the need for the *simultaneous presence of multiple tuples with the same value for the apparent key attributes*!

24

MLS Insert

- What if a U user wants to insert a tuple with vessel = Avenger?
  - If insert is allowed – will there be any problems?
    - We will have 2 Avengers
    - Duplicate primary key - violates unique constraints
  - If we reject the insert – what will happen?
    - Covert channel – U user knows that there is another record with same key value that is not visible to him

Vessel (AK)	Objective	Destination	TC
Micra U	Shipping U	Moon U	U
Vision U	Spying U	Saturn U	U
Avenger C	Spying C	Mars C	C
Logos S	Shipping S	Venus S	S

25

Polyinstantiation

- Phenomenon where *simultaneous presence* of multiple tuples with the *same value* for the key attributes *with different classification*
- Two situations:
  - A low user inserts data in a field which already contains data at higher or incomparable level – *invisible polyinstantiation*
  - A high user inserts data in a field which already contains data at a lower level – *visible polyinstantiation*

26

ML relations – invisible polyinstantiation

Suppose a low user asks to insert a tuple with the same primary key as an existing tuple at a higher level; the DBMS has three choices:

1) Notify the user that a tuple with the same primary key exists at higher level and reject the insertion
   - signaling channel
2) Replace the existing tuple at higher level with the new tuple being inserted at low level
   - allows the low user to overwrite data not visible to him and thus compromising integrity
3) Insert the new tuple at low level without modifying the existing tuple at the higher level (i.e. polyinstantiate the entity)
   - is a reasonable choice; as consequence, it introduces a polyinstantiated entity

27

ML relations – invisible polyinstantiation (Example)

Vessel	Objective	Destination	TC
Micra U	Shipping U	Moon U	U
Vision U	Spying U	Saturn U	U
Avenger U	Shipping U	Mars U	U
Avenger C	Spying C	Mars C	C
Logos S	Shipping S	Venus S	S

A U-user inserts (Avenger, Shipping, Mars)
The tuples with primary key “Avenger” are *polyinstantied*

28

ML relations – visible polyinstantiation

Suppose a high user asks to insert a tuple with the same primary key as an existing tuple at lower level; the DBMS has three choices:

1) Notify the user that a tuple with the same primary key exists and reject the insertion
   - does not introduce a signaling channel; however, rejecting the insertion my result in a DoS problem

2) Replace the existing tuple at lower level with the new tuple being inserted at the high level
   - would result in removing a tuple at lower level and thus introduce a signaling channel

3) Insert the new tuple at high level without modifying the existing tuple at the lower level (i.e. polyinstantiate the entity)
   - is a reasonable choice; as consequence, it introduces a polyinstantiated entity

29

ML relations – visible polyinstantiation (Example)

Vessel	Objective	Destination	TC
Micra U	Shipping U	Moon U	U
Vision U	Spying U	Saturn U	U
Avenger S	Shipping S	Mars S	S
Avenger C	Spying C	Mars C	C
Logos S	Shipping S	Venus S	S

A S-user inserts (Avenger, Shipping, Mars)
The tuples with primary key "Avenger" are *polyinstantied*

30

MLS Model

- Polyinstantiation Integrity

  AK, C_{AK}, C_i -> A_i

  – Implies Primary key in MLS is:
    - AK U C_{AK} U C_R
    - AK are data in PK, C_{AK} is class of PK data, C_R is data not in AK

31

MLS Model - Updates

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

S-user updates Destination to Rigel: ??

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	Rigel S	S

OR

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	null U	U
Enterprise U	Exploration U	Rigel S	S

32

MLS Model - Update

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

S-user updates Objective to Spying: ??

Vessel	Objective	Destination	TC
Enterprise U	Spying S	null U	S

OR

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	null U	U
Enterprise U	Spying S	null U	S

33

More Update Examples

U view:

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

S view:

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	Rigel S	S

U-user wants to update, set Destination = Talos where Vessel = ‘Enterprise”

34

Update

U view:

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U

S View:

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U
Enterprise U	Exploration U	Rigel S	S

Suppose S-users want to update, set objective=spying where Vessel = 'Enterprise" and destination ='Rigel'

35

Update

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U
Enterprise U	Spying S	Rigel S	S

What if S-user set objective=spying where Vessel="Enterprise"?

Vessel	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U
Enterprise U	Spying S	Talos U	S
Enterprise U	Spying S	Rigel S	S

36

Delete

- Because of the *-property, only tuples that satisfy the predicates AND t[TC] = c are deleted from R_c (R_c is table at classification c)
- To maintain inter-instance integrity, polyinstantiated tuples are also deleted from $R_{c'>c}$
  - If t[AK] = c, then any polyinstantiated tuples in $R_{c'>c}$ will be deleted from $R_{c'>c}$
  - If t[AK] < c, then the entity will continue to exist in $R_{t[AK]}$ and in $R_{c'>t[AK]}$

37

Summary

- MAC protects against TH
- Vulnerable to covert channels
- Subjects and objects have to be classified which may not always be feasible

38