Information Security and Security Architecture

(Informasjonssikkerhet og sikkerhetsarkitektur)

Hanno Langweg
Norwegian Information Security Laboratory – NISlab
Department of Computer Science and Media Technology
Gjøvik University College

Administrivia

- Exercises
  * Solutions for first assignment are on-line
  * Second assignment can be downloaded
- Books available?
- Mini exam is available in Fronter on request
- IMT4051 Cryptology exercise offer
  * 1545-1800 on Thursdays in lecture weeks (room A126)
  * Today.

Mini exam #1

- 100 user accounts with 100 passwords (hash file is salted). Each password is chosen from an alphabet of 100 characters and is exactly 10 characters long. An attacker is able to try 1 billion (1 000 million) passwords per second.
  * How many possible passwords are there?
  * How long does it take the attacker to guess one of the passwords, how long does it take to guess all the passwords?
  * Give a minimum, maximum and average number (in seconds)
  * Explain why your 7 numbers are correct
- 15 minutes if you write in Norwegian, 20 if you write in English
- Good luck! (And do not forget your name)

Break

Access Control Models and Policies

Access Control Policies

- General Models
  - HRU Harrison Ruzzo Ullman
  - Take-Grant
- Confidentiality Policies
  - BLP Bell-La Padula
  - Chinese Wall
- Integrity Policies
  - Biba
  - Clark-Wilson
- RBAC Role-Based Access Control

HRU Harrison Ruzzo Ullman Model – Motivation

- Access control modelling in computer security started in 1970s
- Harrison, Ruzzo, Ullman (1975):
  Abstract general model of protection mechanisms
- Not dependent on specific policy
  - ∗ Many policies can be modelled in HRU
  - ∗ Need a policy to be useful
- Safety question:
  Can a subject acquire a particular right to an object?
- Result of HRU: Safety question undecidable in general case!

HRU – Definition

- S set of subjects
- O set of objects, $S \subseteq O$
- A finite set of access rights
- $R = (R_{SO})_{s \in S, o \in O}$ access matrix, $r_{so} \subseteq A$ rights subject s has on object o
- 6 primitive operations
  * enter r into r_{so}, delete r from r_{so} $(r \in A)$
  * create subject s, delete subject s
  * create object o, delete object o

HRU – Definition (cont.)

- C set of commands
  - $c(X_1, \ldots, X_k)$, c name of command, $X_1, \ldots, X_k$ parameters (objects)
  - Conditions: conjunction of triples (r, s, o)
  - If for all triples $r \in (s, o)$ in the access matrix, command may be executed
  - Interpretation I maps C into sequences of primitive operations
  - Similar to batch job, database transaction

HRU – Examples

- Command $CREATE(s, o)$

  // no conditions

  create object o
  enter own into (s, o)

- Command $GRANT_r(s_1, s_2, o)$

  condition: $own \in (s_1, o)$

  enter r into (s_2, o)

- Policy defined by S, O, R, C

HRU – State changes in access matrix (i)

- State change by primitive operation

  (S, O, R), (S', O', R') configurations of a protection system, c primitive operation

  Then $(S, O, R) \Rightarrow_c (S', O', R')$ if one of the following holds

i) c = enter r into (s, o) and $S = S'$, $O = O'$, $s \in S$, $o \in O$, $R'[s_1, o_1] = R[s_1, o_1]$ if $(s_1, o_1) \neq (s, o)$ and $R'[s, o] = R[s, o] \cup \{r\}$

ii) c = delete r from (s, o) and $S = S'$, $O = O'$, $s \in S$, $o \in O$, $R'[s_1, o_1] = R[s_1, o_1]$ if $(s_1, o_1) \neq (s, o)$ and $R'[s, o] = R[s, o] - \{r\}$

HRU – State changes in access matrix (ii)

iii) c = create subject s', s' is a new symbol not in O, $S' = S \cup \{s'\}$, $O' = O \cup \{s'\}$, $R'[s,o] = R[s,o] \forall (s,o) \in S \times O$, $R'[s',o] = \varnothing \forall o \in O'$ and $R'[s,s'] = \varnothing \forall s \in S'$

iv) c = create object o', o' is a new symbol not in O, $S' = S$, $O' = O \cup \{o'\}$, $R'[s,o] = R[s,o] \forall (s,o) \in S \times O$ and $R'[s,o'] = \varnothing \forall s \in S$

v) c = destroy subject s', $s' \in S$, $S' = S - \{s'\}$, $O' = O - \{s'\}$ and $R'[s,o] = R[s,o] \forall (s,o) \in S' \times O'$

vi) c = destroy object o', $o' \in O - S$, $S' = S$, $O' = O - \{o'\}$ and $R'[s,o] = R[s,o] \forall (s,o) \in S' \times O'$

HRU – State changes in access matrix (iii)

- State change by command

  (S, O, R), (S', O', R') configurations of a protection system, C command

  Then $(S, O, R) \rightarrow_C (S', O', R')$ if

i) $\forall (r, s, o) \in conditions(C)\ r \in R[s, o]$

ii) $I(C) = c_1, \ldots, c_m$, c_i primitive operations, then $\exists m \geq 0$, configurations (S_i, O_i, R_i) such that

a) $(S, O, R) = (S_0, O_0, R_0)$

b) $(S_{i-1}, O_{i-1}, R_{i-1}) \Rightarrow_{c_i} (S_i, O_i, R_i)$ for $0 < i \leq m$

c) $(S_m, O_m, R_m) = (S', O', R')$

HRU – State changes in access matrix (iv)

- $(S, O, R) \rightarrow (S', O', R')$ if there is some command C such that $(S, O, R) \rightarrow_C (S', O', R')$
- $(S, O, R) \rightarrow^* (S', O', R')$ for zero or more applications of $\rightarrow$

HRU – Example Unix

- Simple Unix protection mechanism
  - ∗ Owner of file specifies privileges r, w, x for himself and others
  - ∗ (superuser disregarded here)
- Two challenges
  - ∗ No bound on number of subjects
    ⇝ not possible to “give all subjects privilege”
  - ∗ No disjunction of conditions
    Owner or has privilege

HRU – Example Unix (cont.)

- Place access rights in (o, o) entry of matrix
- Command $ADDownerREAD(s, o)$
  * $own \in R[s, o]$: enter $oread$ into (o, o)
- Command $ADDanyoneREAD(s, o)$
  * $own \in R[s, o]$: enter $aread$ into (o, o)
- Commands $READ(s, o)$
  * $own \in R[s, o] \wedge oread \in R[o, o]$ or $aread \in R[o, o]$
  * enter $read$ into (s, o) – temporary addition to matrix
  * delete $read$ from (s, o)

Two $READ$ commands simulate disjunction of conditions

HRU – Safety question

System is "safe" when access to objects is impossible without concurrence of owner

⇝ User should be able to tell impact of an action

- Can a generic right be "leaked" to an "unreliable" subject?
  - Owner can give away right
  - Reliable subjects
  - Can right be added to matrix where it is not initially?

OBS: Safety usually used with respect to causing or preventing injury

HRU – Safety question, particular object

- Safety question concerned with leakage of right
- Leakage of right r to object o_1
  - Two new rights: r', r''
  - Add r' to (o_1, o_1)
  - Add command $DUMMY(s, o)$
    conditions: $r' \in (o, o) \wedge r \in (s, o)$
    enter r'' into (o, o)
  - Leaking r to o_1 now equivalent with leaking r'' to anybody

HRU – Safety question, definitions (i)

i) Definition
Given a protection system, we say command $c(X_1, \ldots, X_n)$ leaks **right** r if its interpretation has a primitive operation of the form enter r into (s, o) for some s and o.

ii) Definition
Given a protection system and right r, we say that initial configuration (S_0, O_0, R_0) is **safe** for r if there does not exist configuration (S, O, R) such that $(S_0, O_0, R_0) \rightarrow *(S, O, R)$ and there is a command $c(X_1, \ldots, X_n)$ whose conditions are satisfied in (S, O, R), and that leaks r via enter r into (s, o) for some subject $s \in S$ and object $o \in O$ with $r \notin R[s, o]$.

HRU – Safety question, definitions (ii)

iii) Definition
A protection system is mono-operational if each command's interpretation is a single primitive operation.

Theorem

There is an algorithm which given a mono-operational protection system, a generic right r and an initial configuration (S_0, O_0, R_0) determines whether or not (S_0, O_0, R_0) is safe for r in this protection system.

Proof ⇝ see second assignment

HRU – Undecidability of safety question (i)

Turing machine TM**:** (Q, T, δ, q_0)

- Q set of states, initial state q_0, final state q_f
- T distinct set of tape symbols
- Blank symbol $\perp$ initially on each cell of tape (infinite to the right)
- Tape head always over some cell of tape
- Moves of TM given by function δ: $Q \times T \rightarrow Q \times T \times \{L, R\}$

  Reading symbol in particular state leads to new state, overwriting with new symbol, moving head to left or right

  (Head never moves off the leftmost cell)

HRU – Undecidability of safety question (ii)

Halting problem

It is undecidable whether a given Turing machine will eventually enter the final state

There is no general algorithm to determine halting for arbitrary Turing machines. There is not even a finite set of algorithms.

HRU – Undecidability of safety question (iii)

Theorem

It is undecidable whether a given configuration of a given protection system is safe for a given generic right.

Proof

- Protection system can simulate behaviour of arbitrary TM
- Leakage of right corresponds to TM entering q_f
- Halting problem is undecidable, hence the theorem is proved

HRU – Undecidability of safety question (iv)

Simulation of $TM\ (Q, T, \delta, q_0)$ **with protection system** (S, O, R, C)

- Set of rights $A := Q \cup T \cup \{own\} \cup \{end\}$, R access matrix
- Set of subjects S represents cells; s_i cell number i
- $S = O$
- Tape represented by list of subjects, s_i owns s_{i+1}
  $own \in R[s_i, s_{i+1}]$
- Last cell, subject s_k, marked by special right: $end \in R[s_k, s_k]$
- Tape symbol X in cell i represented by right to itself: $X \in R[s_i, s_i]$
- Current state q and tape head over cell j: $q \in R[s_j, s_j]$

HRU – Undecidability of safety question (v)

Example

- TM in state q with cell contents W, X, Y, Z, tape head at cell 2
- Representing tape content, current state and tape head position in access matrix

$$\begin{array}{c|cccc} & s_1 & s_2 & s_3 & s_4 \\ \hline s_1 & \{W\} & \{own\} & & \\ s_2 & & \{X, q\} & \{own\} & \\ s_3 & & & \{Y\} & \{own\} \\ s_4 & & & & \{Z, end\} \end{array}$$

HRU – Undecidability of safety question (vi)

Moves δ

- $\delta(q, X) \rightarrow (p, Y, L)$ left move

  Command $C_{qX}(s, s')$
  Conditions: $own \in (s, s') \wedge q \in (s', s') \wedge X \in (s', s')$

  Interpretation:
  delete q from (s', s')
  delete X from (s', s')
  enter p into (s, s)
  enter Y into (s', s')

HRU – Undecidability of safety question (vii)

- $\delta(q, X) \rightarrow (p, Y, R)$ right move

  Ordinary right move command $C_{qX}(s, s')$
  Conditions: $own \in (s, s') \wedge q \in (s, s) \wedge X \in (s, s)$
  Interpretation:
  delete q from (s, s), delete X from (s, s)
  enter p into (s', s'), enter Y into (s, s)

  Moving beyond current end of tape command $D_{qX}(s, s')$
  Conditions: $end \in (s, s) \wedge q \in (s, s) \wedge X \in (s, s)$
  Interpretation:
  delete q from (s, s), delete X from (s, s),
  delete end from (s, s), enter Y into (s, s), create subject s',
  enter $\perp$ into (s', s'), enter p into (s', s'), enter end into (s', s')

Example

- TM from previous example, $\delta(q, X) \rightarrow (p, Y, L)$

$$
\begin{array}{c|cccc}
 & s_1 & s_2 & s_3 & s_4 \\
\hline
s_1 & \{W\} & \{own\} & & \\
s_2 & & \{X, q\} & \{own\} & \\
s_3 & & & \{Y\} & \{own\} \\
s_4 & & & & \{Z, end\}
\end{array}
\qquad
\begin{array}{c|cccc}
 & s_1 & s_2 & s_3 & s_4 \\
\hline
s_1 & \{W, p\} & \{own\} & & \\
s_2 & & \{Y\} & \{own\} & \\
s_3 & & & \{Y\} & \{own\} \\
s_4 & & & & \{Z, end\}
\end{array}
$$

- Applying command C_{qX}

HRU – Undecidability of safety question (ix)

- Initial matrix has one subject s_1, $R[s_1, s_1] = \{q_0, \perp, end\}$
- Each command deletes and adds one state
- Each entry contains at most one tape symbol
- Only one entry contains end

⇝ **In each reachable configuration of the protection system at most one command is applicable. The protection system therefore exactly simulates TM.**
If TM enters q_f, right q_f is leaked, otherwise (S, O, R, C) is safe.
Since it is undecidable whether TM enters q_f, it must be undecidable whether the protection system is safe for q_f.

This concludes the proof.

HRU – Undecidability of safety question (x)

Although we can give different algorithms to decide safety for different classes of systems, we can never hope even to cover all systems with a finite, or even infinite, collection of algorithms.

Open question:

- Where is the boundary between decidable and undecidable safety questions in access control models?

The Take-Grant model

Author not known (ca. 1970s)

- Based on directed graph
- Change of protection state is represented as change of graph
- Safety decidable in linear time

Take-grant – Definitions

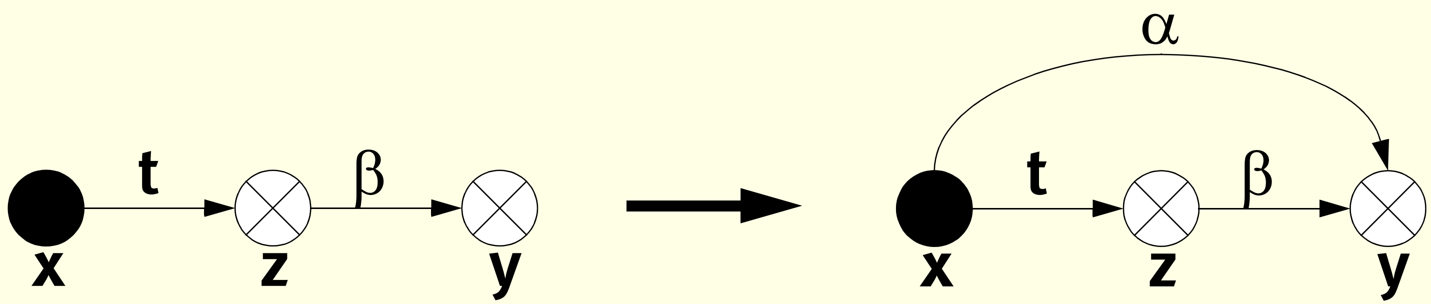
- G directed graph
- Vertices are subjects (●), objects (O), subjects/objects (⊗)
- Labelled edges indicate rights that source has over destination
- R set of rights including $\{t, g\}$ (take, grant)
- 4 graph rewriting rules ("de iure")
  - ∗ Take
  - ∗ Grant
  - ∗ Create
  - ∗ Remove

Take-grant – Graph rewriting rules (i) – Take

x, y, z distinct vertices, x subject, $\alpha \subseteq \beta \subseteq R$ set of rights

Edge x to z labelled t, edge z to y labelled β

Then edge x to y is added and labelled α



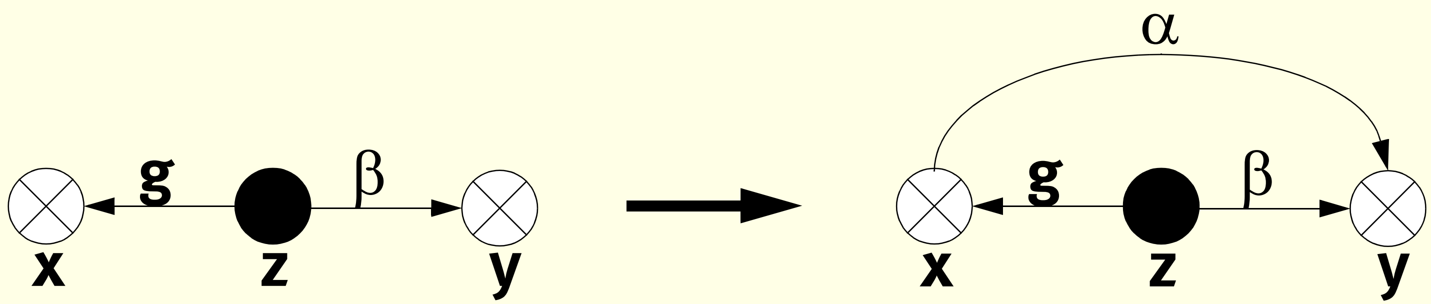
x takes (α to y) from z

Take-grant – Graph rewriting rules (ii) – Grant

x, y, z distinct vertices, z subject, $\alpha \subseteq \beta \subseteq R$ set of rights

Edge z to x labelled g, edge z to y labelled β

Then edge x to y is added and labelled α

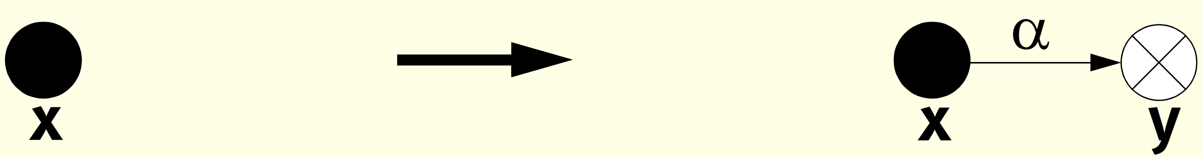


z grants (α to y) to x

Take-grant – Graph rewriting rules (iii) – Create

x **subject,** $\alpha \subseteq R$ **set of rights**

Add a new vertex y **and an edge** x **to** y **labelled** α



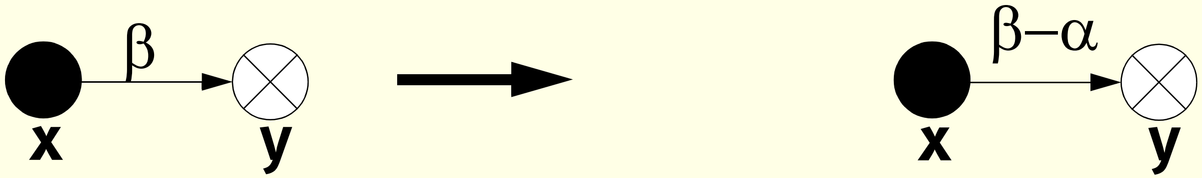
x **creates (**α **to new vertex)** y

Take-grant – Graph rewriting rules (iv) – Remove

x, y distinct vertices, x subject, $\alpha \subseteq \beta \subseteq R$ set of rights

Edge x to y labelled α

Then α labels of edge x to y are deleted; edge is deleted if label=$\emptyset$



x removes (α to) y

Take-grant – De facto rules – Can-share

Can x obtain α rights over y?

- Predicate $can-share(\alpha, x, y, G_0)$ true if there exists sequence of protection graphs $G_1, \ldots, G_n$ such that $G_0 \to {}^*G_n$ using only de iure rules and in G_n there is an edge x to y labelled α
- Theorem stating requirements for $can-share$ involves definition of tg-connectedness, islands, bridges
- Only tg-paths discussed here

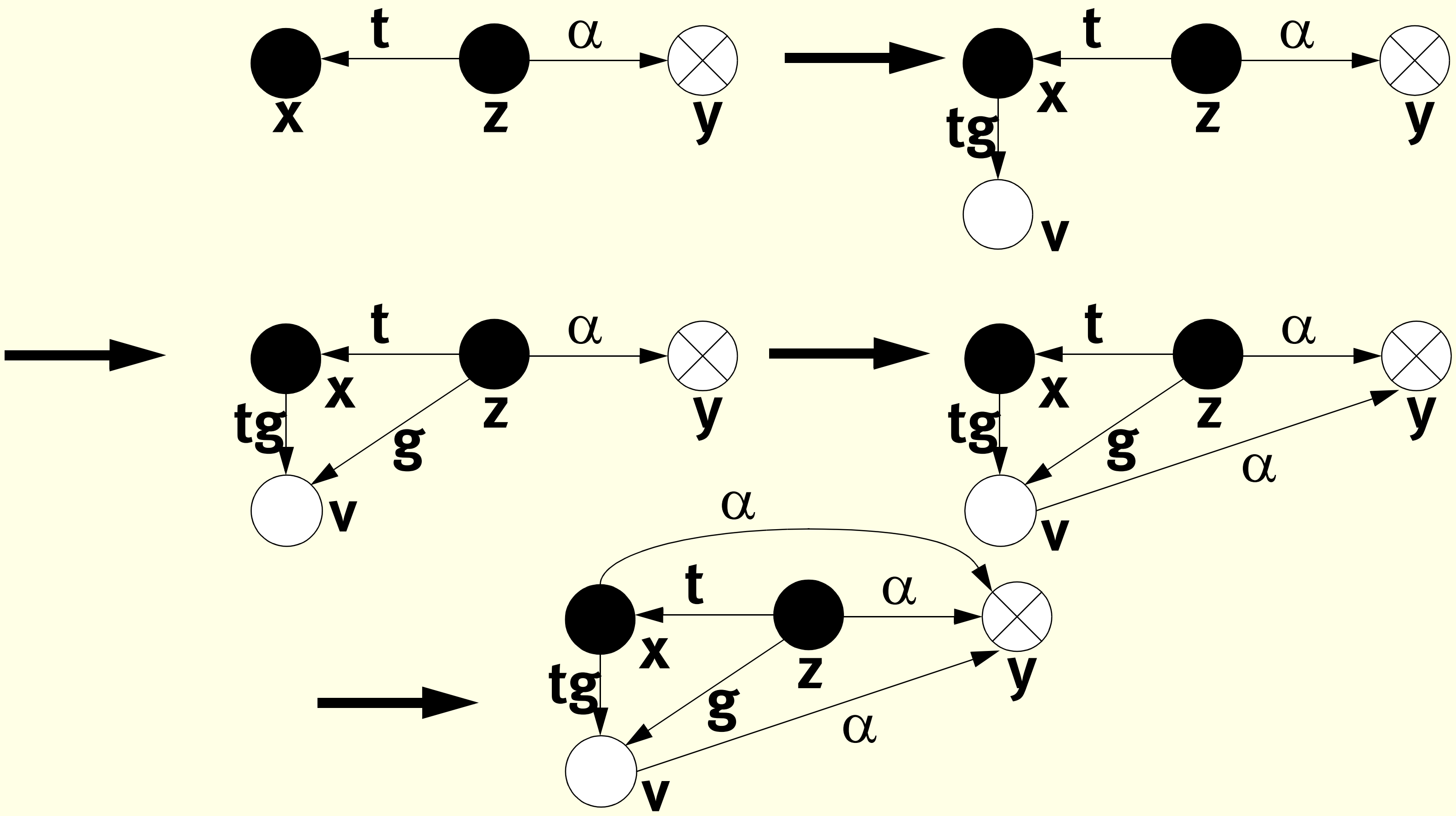
Explored at length e.g. in Bishop 3.3.1

Take-grant – tg-connected

tg-path is sequence of connected vertices with edges labelled t or g.
Vertices are tg-connected if there is a tg-path between them.

- tg-paths of length 1
  - Take
  - Grant
  - Reversed take
  - Reversed grant

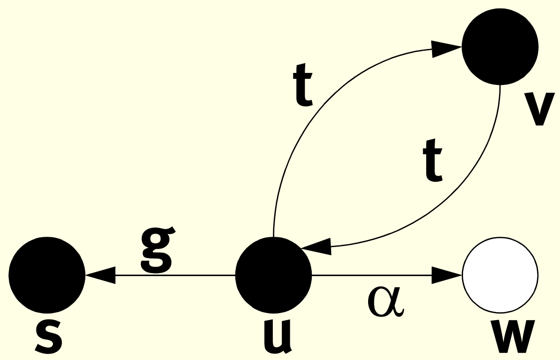
Take-grant – Reversed take



Similar proof for reversed grant ⇝ homework

Take-grant – De-facto rules – Can-steal

- Similar to can-share
- No grant rights may be stolen



i)u grants (t to v) to s

ii)s takes (t to u) from v

iii)s takes (α to w) from u

- $can - steal(\alpha, s, w, G_0)$ is true

Take-grant – Safety question

- Safety decidable in linear time with respect to graph size
- Take-grant less expressive than HRU
  (special case of HRU)
- Relation to other access models, e.g. TG is also special case of SPM Schematic Protection Model

⇝ **Could be a project topic**

Confidentiality Policies

Confidentiality policies – Bell La Padula

Bell, LaPadula (1976)

- Motivated by military security
- Significant security model
- Played important role in design of secure operating systems
- New models often compared with BLP

- Deals with confidentiality
- Information flow when subject alters object
- Supports multi-level security policies

BLP – Definitions

- S set of subjects, O set of objects
- A set of access operations, $A = \{execute, read, append, write\}$
- L set of security levels with a partial ordering $\leq$
- $B = Pow(S \times O \times A)$ set of current accesses
  Set of sets of tuples, $b \in B$ contains (s, o, a) of current accesses
- M set of access control matrices, $M = (M_{SO})_{s \in S, o \in O}$
- $F \subseteq L^S \times L^S \times L^O$ set of security level assignments
  * f_S: $S \to L$ maximal security level of a subject
  * f_C: $S \to L$ current security level of a subject, $f_C \leq f_S$
  * f_O: $O \to L$ classification of an object

BLP – State of a system

- State set $B \times M \times F$
  - Current accesses
  - Access matrix
  - Security level assignments
- Multi-level security: subject level must dominate object level
- State is secure if two (three) properties are satisfied
  - Simple security property: “no read up”
  - *-property: “no write down”
    (pronounced “star property”)
  - (Discretionary security property)

BLP – Security properties

Simple security property

A state (b, M', f) **satisfies the simple security property if for each element** $(s, o, a) \in b$ **with** $a = read \vee a = write$ **the following condition holds:** $f_O(o) \leq f_S(s)$**.**

*-property

A state (b, M', f) **satisfies the *-property if for each element** $(s, o, a) \in b$ **with** $a = write \vee a = append$ **the following condition holds:** $f_C(s) \leq f_O(o)$**.**

In addition $f_O(o') \leq f_O(o)\ \forall o'$ **with** $(s, o', a') \in b$ **and** $a = read \vee a = write$

BLP – Security properties (cont.)

Discretionary security property

A state (b, M', f) **satisfies the discretionary security property if for each element** $(s, o, a) \in b$ **the following condition holds:** $a \in M'_{so}$**.**

BLP – Example

- $S = \{s_1, s_2\}$, $O = \{o_1, o_2, o_3\}$,
  $L = \{unclassified, secret, top\ secret\}$
- $f_S(s_1) = top\ secret$, $f_S(s_2) = unclassified$

  $f_C(s_1) = secret$, $f_C(s_2) = unclassified$
- $f_O(o_1) = top\ secret$, $f_O(o_2) = secret$, $f_O(o_3) = unclassified$
- $b = \{(s_1, o_2, read), (s_1, o_1, write), (s_2, o_1, append),$
  $(s_2, o_3, read), (s_2, o_2, append)\}$
- Secure state?

BLP – Example (cont.)

i) $(s_1, o_2, read)$ [SSP] $f_O(o_2) = secret \leq top\ secret = f_S(s_1)$ (+)

ii) $(s_1, o_1, write)$ [SSP,*]

$f_O(o_1) = top\ secret \leq top\ secret = f_S(s_1)$
$f_C(s_1) = secret \leq top\ secret = f_O(o_1)$
$f_O(o_2) = secret \leq top\ secret = f_O(o_1)$ (+)

iii) $(s_2, o_1, append)$ [*] $f_C(s_1) = secret \leq top\ secret = f_O(o_3)$ (+)

iv) $(s_2, o_3, read)$ [SSP]

$f_O(o_3) = unclassified \leq unclassified = f_S(s_2)$? (+)

v) $(s_2, o_2, append)$ [SSP,*]

$f_C(s_2) = unclassified \leq secret = f_O(o_2)$
$f_O(o_3) = unclassified \leq secret = f_O(o_2)$ (+)

BLP – Information flow

High-level subjects cannot disclose information to low-level subjects

To allow this

- Temporarily downgrade a high-level subject: f_C
  - Processes do not retain memory
  - Choose f_C upon login
- Trusted subjects: can violate *-property
  - Trusted vs trustworthy
  - Security administrator

Confidentiality policies – Chinese wall

Brewer, Nash (1989)

- Motivated by consultancy/banking
- Access based on conflicts of interest
- Modification of BLP

Chinese wall – Definition

- C set of companies
- O set of objects concerning a single company
- S set of subjects ("analysts")
- $y : O \rightarrow C$ company dataset of an object
- $x : O \rightarrow Pow(C)$ conflict of interest class of an object
- $(x(o), y(o))$ security label of an object
- Sanitised information has $x(o) = \emptyset$
- History matrix H of objects accessed in the past

$$H_{s,o} = \begin{Bmatrix} true,\ if\ s\ has\ had\ access\ to\ o \\ false,\ if\ s\ never\ had\ access\ to\ o \end{Bmatrix}$$

Chinese wall – Security properties

Initial state: $H_{S,O}$ **empty**

s **is granted access to** o **if**

- o belongs to company dataset already held by user
- o is in different conflict of interest class

Simple security property

Subject s **is granted access to object** o **only if** $\forall o'$ **with** $H_{s,o'} = true$, $y(o) \notin x(o') \vee y(o) = y(o')$

*-property

Subject s **is granted modifying access to object** o **only if** s **has no read access to** o' **with** $y(o) \neq y(o') \wedge x(o') \neq \emptyset$

Integrity Policies

Integrity policies – Biba

Biba (1977)

- Motivated by Bell LaPadula
- Very similar
  - Integrity levels (vs security levels)
  - Information flow in opposite direction
    Low integrity information must not affect high integrity inform.
- Variants (two discussed here)

Biba – Static integrity levels

Integrity levels do not change

Simple integrity policy

If subject s can modify object o, then
$integrity\text{-}level_O(o) \leq integrity\text{-}level_S(s)$

Integrity *-property

If subject s can observe object o, then s can have modifying access to other object p only if $integrity\text{-}level_O(p) \leq integrity\text{-}level_O(o)$

Biba – Dynamic integrity levels

Integrity levels adjusted after contact with low-integrity information

Subject low watermark property

s **observes** o **at any level. Then** $f_S(s) := inf(f_S(s), f_O(o))$

Object low watermark property

s **modifies** o **at any level. Then** $f_O(o) := inf(f_S(s), f_O(o))$

Integrity policies – Clark-Wilson

Clark, Wilson (1987)

- Motivated by commercial integrity needs (vs military)
- Two integrity levels
- Certification and enforcement rules

Clark-Wilson – Definitions

- CDI constrained data item (high integrity)
  UDI unconstrained data item (low integrity)
- IVP integrity verification procedure
  Confirms that CDIs confirm to integrity specification
- TP transformation procedure
  Change set of CDIs from one valid state to another
- System ensures that only TPs manipulate CDIs
  Validity of TP verified by certification (done for specific policy)

Clark-Wilson – Enforcement rules

4 enforcement rules (abbreviated)

- E1: CDIs are changed only by authorised TP (list of TP, CDIs)
- E2: Users authorised for TP (list of user, TP, CDIs) (makes E1 unnecessary)
- E3: Users are authenticated
- E4: Authorisation lists changed only by security officer

Clark-Wilson – Certification rules

5 certification rules (abbreviated)

- C1: IVP validates CDI state
- C2: TPs preserve valid state
- C3: Suitable separation of duty
- C4: TPs write to append-only log
  (log modelled as CDI)
- C5: TPs validate UDI

More Access Control

RBAC Role-Based Access Control

Ferraiolo, Kuhn (1992), Sandhu et al. (1996)

- Roles are collections of permissions
  - ∗ Simpler management
  - ∗ Users – roles
  - ∗ Permission – roles
  - ∗ Role hierarchies
- Roles vs groups
  - ∗ Groups are administrative collections of users
- Similarity with maximum and current security levels
- Policy-neutral

Information flow models

- Different perspective than access rights
- Similar framework as BLP
  - Objects labelled with security classes (form a lattice)
  - Information may only flow upwards
- Flow from x to y if something learned about x by observing y
  - Explicit information flow: y:=x
  - Implicit information flow: If $x = 0$ then y:=1
- Security in information flow model undecidable
- Little practical use as of today

Access control models and policies – Summary

- Expressiveness of model vs decidability of safety question
- Different representations: matrices, lists, graphs, state machines
- Focus of research
  - Much work on confidentiality policies
  - Less work on integrity policies
  - Even less work on availability policies
- Current systems mostly use DAC, some RBAC
- Management of access control important in commercial sector

Project

Project

- Write and present an essay/research report
- 25% of course performance evaluation
- Choose topic no later than 2004-09-24 1200
- Submit abstract, table of contents and list of used literature ca. 3 weeks later
- Presentation will be 2004-11-11, 0900-1515 (no exercise on that Thursday)
- More information on course homepage http://nislab.hig.no/Courses/IMT4161/project.html

Project topics

Suggestions

- Classical papers in computer security
  - Security models, evaluation and evaluation criteria
- Access control policies
  - RBAC, temporal access control, new approaches
- Software security
  - Architecture principles, hardware support, programming errors
- Copy protection

Find a topic you are interested in! This list provides only ideas! Ask!

Agree with me on a topic no later than 2004-09-24 1200

Project presentation

- Thursday 2004-11-11, 0900-1515
  (no exercise on that Thursday)
- 15 time slots for presentations (15+5 min)
- Form groups
  * ca. 2-3 per group
  * Those not presenting write a summary of another presentation
- Links to web sites giving advice on how to write and present papers are given on the course homepage
  http://nislab.hig.no/Courses/IMT4161/project.html

End of lecture 2
Next exercise on Thursday, 2004-09-16
Next lecture on Thursday, 2004-09-23

Remember deadline for project topics
Friday, 2004-09-24