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, $S' = S \cup \{s'\}$, $O' = O \cup \{s'\}$, $R'[s,o] = R[s,o] \forall (s,o) \in S \times O$, $R'[s',o] = \emptyset \forall o \in O'$ and $R'[s,s'] = \emptyset \forall s \in S'$

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 $R'[s,o'] = \emptyset \forall s \in S$

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 $(S, O, R) \to_C (S', O', R')$ 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: $r' \in (o, o) \land r \in (s, o)$
    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

	s_1	s_2	s_3	s_4
s_1	$\{W\}$	$\{own\}$		
s_2		$\{X, q\}$	$\{own\}$	
s_3			$\{Y\}$	$\{own\}$
s_4				$\{Z, end\}$

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)

- $\delta(q, X) \to (p, Y, R)$ 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, $\delta(q, X) \rightarrow (p, Y, L)$

	s_1	s_2	s_3	s_4
s_1	$\{W\}$	$\{own\}$		
s_2		$\{X, q\}$	$\{own\}$	
s_3			$\{Y\}$	$\{own\}$
s_4				$\{Z, end\}$

	s_1	s_2	s_3	s_4
s_1	$\{W, p\}$	$\{own\}$		
s_2		$\{Y\}$	$\{own\}$	
s_3			$\{Y\}$	$\{own\}$
s_4				$\{Z, end\}$

- 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?