Chapter 3: Foundational Results

- Overview
- Safety Questions
- Turing Machine Mapping

Overview

- Safety Questions
- HRU Model
What Is “Secure”?

- Adding a generic right \( r \) where there was not one is “leaking”
- If a system \( S \), beginning in initial state \( s_0 \), cannot leak right \( r \), it is safe with respect to the right \( r \).
- What is a generic right?

  Generic rights correspond to a class of objects

Definitions

- **Definition 3-1.** When a generic right \( r \) is added to an element of the access control matrix not already containing \( r \), that right is said to be leaked.
- **Definition 3-2.** If a system can never leak the right \( r \), the system (including the initial state \( s_0 \)) is called safe with respect to the right \( r \). If the system can leak the right \( r \) (enter an unauthorized state), it is called unsafe with respect to the right \( r \).
Example

- A computer system allows the network administrator to read all network traffic. It disallows all other users from reading this traffic. The system is designed in such a way that the network administrator cannot communicate with other users. Is this system safe?

Yes, there is no way for the right $r$ of the network administrator over the network device to leak.

Safety Question

- Does there exist an algorithm for determining whether a protection system $S$ with initial state $s_0$ is safe with respect to a generic right $r$?
  - Here, “safe” = “secure” for an abstract model
Mono-Operational Commands

- Answer: yes
- Sketch of proof:
  Consider minimal sequence of commands $c_1, ..., c_k$ to leak the right $r$.
  - Can merge all creates into one
  Worst case: insert every right into every entry; with $s$ subjects and $o$ objects initially, and $n$ rights, upper bound is $k \leq n(s+1)(o+1)$

General Case

- Answer: no
- Sketch of proof:
  Reduce halting problem to safety problem
  Turing Machine review:
  - Infinite tape in one direction
  - States $K$, symbols $M$; distinguished blank $b$
  - Transition function $\delta(k, m) = (k', m', L)$ means in state $k$, symbol $m$ on tape location replaced by symbol $m'$, head moves to left one square, and enters state $k'$
  - Halting state is $q_f$; TM halts when it enters this state
Turing Machine

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

Mapping

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

Current state is $k$

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>A</td>
<td>own</td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td>B</td>
<td>own</td>
<td></td>
</tr>
<tr>
<td>$s_3$</td>
<td>C</td>
<td>$k$</td>
<td>own</td>
</tr>
<tr>
<td>$s_4$</td>
<td></td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>
After $\delta(k, C) = (k_1, X, R)$ where $k$ is the current state and $k_1$ the next state

---

**Command Mapping**

$\delta(k, C) = (k_1, X, R)$ at intermediate becomes

```plaintext
command $c_{k,C}(s_3, s_4)$
if own in $A[s_3, s_4]$ and $k$ in $A[s_3, s_3]$ and $C$ in $A[s_3, s_3]$
then
  delete $k$ from $A[s_3, s_3]$;
  delete $C$ from $A[s_3, s_3]$;
  enter $X$ into $A[s_3, s_3]$;
  enter $k_1$ into $A[s_4, s_4]$;
end
```
Mapping

After $\delta(k_1, D) = (k_2, Y, R)$ where $k_1$ is the current state and $k_2$ the next state

Command Mapping

$\delta(k_1, D) = (k_2, Y, R)$ at end becomes

```plaintext
command crightmost_{k_2}(s_4 s_5)
if end in A[s_4, s_4] and k_1 in A[s_4, s_4] and D in A[s_4, s_4]
then
   delete end from A[s_4, s_4];
   create subject s_5;
   enter own into A[s_4, s_5];
   enter end into A[s_5, s_5];
   delete k_1 from A[s_4, s_4];
   delete D from A[s_4, s_4];
   enter Y into A[s_4, s_4];
   enter end into A[s_4, s_5];
end
```
Rest of Proof

- Protection system exactly simulates a Turing Machine
  - Exactly 1 end right in ACM
  - 1 right in entries corresponds to state
  - Thus, at most 1 applicable command
- If TM enters state $q_f$, then right has leaked
- If safety question decidable, then represent TM as above and determine if $q_f$ leaks
  - Implies halting problem decidable
- Conclusion: safety question undecidable

Other Results

- Set of unsafe systems is recursively enumerable
- Delete create primitive; then safety question is complete in $P$-SPACE
- Safety question for mono-conditional, monotonic protection systems is decidable
- Safety question for mono-conditional protection systems with create, enter, delete (and no destroy) is decidable.
Key Points

• Safety problem undecidable
• Limiting scope of systems can make problem decidable