ECS 235B Module 6
HRU Result
What Is “Secure”? 

• Adding a generic right $r$ where there was not one is “leaking” 
  • In what follows, a right leaks if it was not present \textit{initially} 
  • Alternately: not present \textit{in the previous state} (not discussed here) 

• If a system $S$, beginning in initial state $s_0$, cannot leak right $r$, it is \textit{safe with respect to the right $r$} 
  • Otherwise it is called \textit{unsafe with respect to the right $r$}
Safety Question

• Is there 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 $c_1, \ldots, c_k$ to leak the right.
  • Can omit delete, destroy (with some rewriting)
  • 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 $k \leq n(s+1)(o+1)+1$
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
Mapping

Current state is $k$

<table>
<thead>
<tr>
<th></th>
<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>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td>B</td>
<td></td>
<td>own</td>
<td></td>
</tr>
<tr>
<td>$s_3$</td>
<td></td>
<td>C $k$</td>
<td></td>
<td>own</td>
</tr>
<tr>
<td>$s_4$</td>
<td></td>
<td></td>
<td></td>
<td>D end</td>
</tr>
</tbody>
</table>
Mapping

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

\[
\text{command } c_{k,C}(s_3, s_4) \\
\text{if own in } A[s_3, s_4] \text{ and } k \text{ in } A[s_3, s_3] \\
\hspace{1em} \text{and } C \text{ in } A[s_3, s_3] \\
\text{then} \\
\hspace{1em} \text{delete } k \text{ from } A[s_3, s_3]; \\
\hspace{1em} \text{delete } C \text{ from } A[s_3, s_3]; \\
\hspace{1em} \text{enter } X \text{ into } A[s_3, s_3]; \\
\hspace{1em} \text{enter } k_1 \text{ into } A[s_4, s_4]; \\
\text{end}
\]
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,C}(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];
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];
create subject s_5;
enter own into A[s_5,s_5];
enter end into A[s_5,s_5];
enter k_2 into A[s_5,s_5];
end
```
Rest of Proof

- Protection system exactly simulates a TM
  - Exactly 1 \textit{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 \texttt{create} primitive; then safety question is complete in P-SPACE
• Delete \texttt{destroy, delete} primitives; then safety question is undecidable
  • Systems are monotonic
• Safety question for biconditional protection systems is decidable
• Safety question for monoconditional, monotonic protection systems is decidable
• Safety question for monoconditional protection systems with \texttt{create, enter, delete} (and no \texttt{destroy}) is decidable.