CS 591: Introduction to Computer Security

Information Flow Epilog

James Hook
Last time:

- Information flow security
  - Denning and Denning as presented in Chapter 15
  - Flow Caml “nutshell” paper

- Compilation can be made aware of confidentiality levels
  - Levels must be identified
  - Levels can be tracked through computational effects: environment, state, control, exceptions, concurrency (Not shown in Flow Caml)
Does it work?

• Theoretical results
  – Volpano, Irvine and Smith (JCS ‘96) showed Soundness
    • “If an expression $e$ can be given a type $\tau$ in our system, then Simple Security says ... that only variables at level $\tau$ or lower in $e$ will have their contents read when $e$ is evaluated (no read up).... On the other hand, if a command $c$ can be given a type $\tau$ cmd then Confinement says ... that no variable below level $\tau$ is updated in $c$ (no write down).”
  – Using modern language theory the techniques in Flow Caml and similar systems can be proven sound
Does it work?

• In practice it is not broadly adopted
  – Technical issue is the complexity of managing policy
  – I suspect there are social issues as well ... the technical issues are not show stoppers
Recall

• Consider an example (in no particular language)

\[
\begin{align*}
H &= \text{readHighDatabase}() \\
L &= \text{readLowUserInput}() \\
\text{If } f(H,L) \\
& \quad \text{then printLow “Success”} \\
& \quad \text{else printLow “Fail”}
\end{align*}
\]

• Assume H is high and L is Low
But!!!

• Consider an example (in no particular language)

\[ H = \text{readHighDatabase(“passwd”)} \]
\[ L = \text{readLowUserInput()} \]

If checkPassword(H,L)
  then printLow “Success”
  else printLow “Fail”

• We do this every day!
Password checking paradox

• Why shouldn’t we allow someone to write the password program?
• Why should we?
Policy

- The password paradox is solved by explicit policy
- Similar issues arise with crypto algorithms
  - LoCypher = encrypt (HighClear, goodKey)
- Cf.
  - LoCypher = encrypt (HighClear, badKey)
FlowCaml and Policy

• FlowCaml solves the policy problem by dividing the program into two parts:
  – Flow caml portion (.fml), with all flows checked
  – Regular caml portion with an annotated interface

• The downgrading of encryption or password validation queries is not done within the flow-checked portion
Policy

• Zdancewic uses other techniques, including explicit downgrade assertions for confidentiality

• Basic philosophy: uniform enforcement with explicit escape mechanism
  – Focus analysis on the exceptions
Further reading