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 τ in our system, then Simple Security says … that only variables at level τ 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 τ *cmd* then Confinement says … that no variable below level τ 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)

```
H = readHighDatabase()

L = readLowUserInput()

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

- Assume H is high and L is Low

But!!!

- Consider an example (in no particular language)

```
H = readHighDatabase("passwd")

L = 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

- Dorothy E. Denning and Peter J. Denning, Certification of Programs for Secure Information Flow, http://www.seas.upenn.edu/~cis670/Spring2003/p504-denning.pdf
- Dennis Volpano, Geoffrey Smith, and Cynthia Irvine, A Sound Type System for Secure Flow Analysis, http://www.cs.fiu.edu/~smithg/papers/jcs96.pdf
- Steve Zdancewic, Lantian Zheng, Nathaniel Nystrom, and Andrew C. Myers, Secure Program Partitioning, http://www.cis.upenn.edu/~stevez/papers/ZZNM02.pdf
- Andrei Sabelfeld and Andrew C. Myers, Language-based Information-Flow Security, http://www.cs.cornell.edu/andru/papers/jsac/sm-jsac03.pdf
- Peng Li and Steve Zdancewic, Downgrading Policies and Relaxed Noninterference, http://www.cis.upenn.edu/~stevez/papers/LZ05a.pdf