CS590U
Access Control: Theory and Practice

Lecture 9 (February 8)

Nondeducibility, Confinement & Covert Channels

A Model of Information

David Sutherland

System Model

- A system is described by an abstract state machine (similar to the noninterference paper)
  - a set of states
  - a set of possible initial states
  - a set of state transformations
- A possible execution sequence consists of
  - an initial state
  - a sequence of transformations applied to the system

Information

- Consider each possible execution sequence as a possible world.
  - the system is one world
- An information function is one that maps each possible world to a value
- Given a set W of all possible worlds, knowing no information, the current world w could be any one in W. Knowing that f1(w)=x, then one knows only those in W such that f1()=x is possible.

Information Flow From f1 and f2

- Given a set W of possible worlds and two functions f1 and f2, we say that information flows from f1 to f2 if and only if there exists some possible world w and some value z in the range of f2 such that
  - $\forall w'$ (f1(w)=f1(w') $\rightarrow$ f2(w')? z)

Proposition

- Proposition: Given W, f1, f2, information does not flow from f1 to f2 if and only if the function f1 ´ f2 is onto.
- Corollary: The information flow relation is symmetric
- Nondeducibility: a system is nondeducibility secure if information does from flow from high inputs to low outputs

Example: Stream Cipher

- Two high users & one low user
  - high user A generates a message
  - high user B generates a random string at a constant rate
  - the XOR of them (if A generates nothing, then 0 is used) is send to the low user
- This is nondeducibility secure
- This is NOT noninterference secure

Relationships Between Nondeducibility & Noninterference

- For deterministic systems with just one high user and one low user, a system is noninterference secure if and only if it is nondeducibility secure.
  - nondeducibility implies noninterference: no high input is also a possible world
  - noninterference implies nondeducbility: every possible world is equivalent to the one with no high-level input

Limitations of Nondeducibility & Noninterference

- Nondeducability may be too weak
  - Allows probabilistic reasoning
  - The stream cipher example is still nondeducibility secure even if high level user B generates 0 each time with 99% probability
- Noninterference may be too strong
  - as demonstrated by the stream cipher example

Comparisons of BLP & Noninterference

- In general, BLP is weaker than noninterference as it does not stop covert channels
- Noninterference is weaker than BLP in that it allows a low user to copy one high-level file to another high-level file
- In both cases, noninterference seems closer to intuition of security

A Note on the Confinement Problem

Butler Lampson

CACM October 1973

The Confinement Problem

- Confine a program's execution so that it cannot transmit information to any other program except its caller.
- Motivation:
  - a customer uses a service program and wants to ensure that the inputs are not leaked by the service program

Ways to leak information

0. The service has memory and can be called by its owner
1. The service writes to a permanent file that can be read by its owner
2. The service writes to a temporary file that can be read by its owner
3. The service sends a message to the owner’s process using interprocess communication

Ways to leak information

4. Information may be encoded in the bill rendered for the service, or payment for resources used by the service program
5. Using file lock as a shared boolean variable
6. By varying its ratio of computing to input/output or its paging rate, the service can transmit information to a concurrently running process

Confinement rules (from the paper)

- A confined program must be memoryless, i.e., it must not be able to preserve information within itself from one call to another
- **Total isolation**: A confined program shall make no calls on any other program
  - sufficient to ensure confinement
  - quite impractical as even system calls may be dangerous and thus need to be forbidden

Less Restrictive Case

- Trusted programs: programs trusted not to leak data or help any confined program that calls them leak data
- **Transitivity**: if a confined program calls another program which is not trusted, then the called program must also be confined.
- It is difficult to write a trustworthy operating system, as some information path are subtle and obscure.

Writing a Trustworthy Program

- A trustworthy program must guard against any possible leakage of data.
- In an operating system, the number of possible channels is large, but finite.
- It is necessary to enumerate all of them and to block each one.

Three Categories of Channels

- Storage: write/read files
- Legitimate: bill for the service program
- Covert: CPU/memory usage
- The following simple principle is sufficient to block all legitimate & covert channels:
  - Masking: A program is confined must allow its caller to determine all its inputs into legitimate and covert channels. We say that the channels are masked by the caller.

On Blocking Covert Channels

- Enforcement: The supervisor must ensure that a confined program's input to covert channels conforms to the caller's specifications.
  - this may require slowing the program down, generating spurious disk references, or whatever, but it is conceptually straightforward
  - The cost of enforcement may be high. A cheaper altrenative is to bound the capacity of the covert channels.

A Comment on the Confinement Problem

Steven B. Lipner

SOSP 1975

Key observations

- The confinement problem is similar in objective to MAC security
  - the common objective is to stop information flow
- Supposedly, *-property solves confinement problem for storage channels
  - Identifying all objects is difficult, but can be done

Closing “Covert Channels” is most difficult

- To close “timing channels”
  - each subject must be constrained to see a virtual time depending only on its activities
  - seems to solve the covert channel problem
  - unclear whether this is possible, because each user also has sense of time outside the system

Conclusion of this paper

- While the storage and legitimate channels of Lampson can be closed with a minimal impact on system efficiency, closing the covert channel seems to impose a direct and unreasonable performance penalty.
- Closing the covert channels seems at a minimum very difficult, and may very well be impossible in a system where physical resources are shared.

Other Discussions on Covert Channels

Covert Channels in MLS

- Covert storage channels: In BLP, if a file is considered to be an object, a low subject may be able to see file names of high, which can encode information.
  - low users can write high files; thus it reasonable to know names of high files
- Covert timing channels

- Covert channels are often noisy
- However, information theory and coding theory can be used to encode and decode information through noisy channels
- Military requires cryptographic components be implemented in hardware
  - to avoid trojan horse leaking keys through covert channels

The Resource Matrix Approach

- An approach to systematically identify covert channels
- Kemmerer: “Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels”, ACM TOCS.
  - Conference version in Oakland 1982.

Intuition

- Finding all resources that are shared between high and low users
  - covert channels reply on sharing of some resource that can be used in an unexpected way to transfer informaion

The Matrix

- Each system resource has a row
- Each lowest-level system operation that can be performed on resources is a column
- Each cell contains a subset of {R,M}
  - R means referencing the resource
  - M means modifying the resource

Criteria for Identifying Covert Channels

- E.g., a storage channel exists when a high user can change an attribute of a shared resource and a low user can detect the change
- E.g., the criteria for a timing channel includes a shared common attribute, a shared time reference, and a means for modulating changes to this attribute.

Polyinstantiation

- Suppose that a High user creates a file named agents, when a Low user tries to create the same file, it would fail, thus leaking information
  - may be solved using naming conventions
- The problem gets more difficult in databases: Suppose that a High user allocate classified cargo to a ship, then a low user may think the ship is empty and tries to allocate other cargos
  - one approach is to use a cover story

The Limit of Formal Security Models

Dorothy Denning

National Computer Systems Security Award Acceptance Speech

Quoted from Denning's Speech

- I learned my second lesson on the limits of models in the mid 80s while working at SRI. I was co-PI for a project to develop a model for a multilevel-secure database system based on views. Peter Neumann, Teresa Lunt, Roger Schell, Bill Shockley, and Mark Heckman were all working with me. Our model, which we called SeaView, grew progressively more complex as we attempted to address the real issues. By the time I left SRI in 1987, I was convinced that I would never want to use a system based on SeaView.

Quoted from Denning’s Speech

- Any hope of usability had been killed by a concept called polyinstantiation, which involved instantiating multiple data values within a single field of a record, all with different security classifications. Polyinstantiation was needed to satisfy the mathematical models of multilevel security, but it got uglier and uglier the deeper we went. I learned then that security models could lead to dreadful systems that nobody would ever use.

Quoted from Denning's Speech

- I left SRI in 1987 and went to Digital because I was tired of security and disillusioned by it. I wanted to work on user interfaces -- on ways of making systems *more* usable. But security was in my blood, and I never really gave it up. While there, I learned my third lesson, namely that building systems based on formal models was extraordinarily time consuming and costly. I saw this earlier, but it was really brought home to me at DEC.

Quoted from Denning’s Speech

- From my West coast office, I tracked Digital's largest security project -- a multi-million dollar effort on the East coast to develop the VAX Secure Virtual System, an A1 operating system. After years of work, the system was scheduled to ship in 1990 and enter formal evaluation, but in February of that year, the project was canceled instead. The projected volume of sales did not justify the projected costs of continuing development and enhancement.

End of Lecture 9

- Next lecture
  - Integrity, Biba, Clark-Wilson