Information Flow Security (Chapter 15)

Information flow: the way information moves through a system

Typically want to preserve confidentiality or integrity of data
  • Confidentiality: prevent flow to a user not authorized to received it
  • Integrity: prevent flow to a process more trustworthy than the data

Information may flow in at least 2 ways:
Protecting information flow through programs

• Let $x$ and $y$ be variables in a program

• A command sequence $c$ causes a flow of information from $x$ to $y$ if, after execution of $c$, some information about the value of $x$ before $c$ was executed can be deduced from the value of $y$ after $c$ was executed.

Explicit information flows:

Implicit information flows:

If $x$ is a variable, then $x$ is the “information flow class” of $x$ (security label, integrity label, category [sec. label + codewords])
Compiler-based mechanisms check that information flows throughout a program are authorized.

A set of program statements is certified with respect to an information flow policy if the information flow within that set of statements does not violate the policy.

For a procedure, need to consider the information flow classes of the input parameters, the output parameters, and the input/output parameters.
Programs consist of several types of statements:

- Assignment statements
- Compound statements
- Conditional statements
- Iterative statements
- Goto statements
- Procedure calls
- Function calls
- Input/output statements

For each type of statement, information flow constraints can be defined:
Execution-based mechanisms

Goal is to prevent information flows at runtime that violate policy

Fairly straightforward for explicit flows:

More difficult when flows are implicit

Fenton (1974) proposed an abstract machine called the Data Mark Machine to study how implicit flows are handled at execution time

- The idea is to include the program counter (PC) and its info. flow class so that implicit flows can be treated as explicit flows (since branches are just assignments to the PC)

Note: in Fenton’s machine errors are not reported to the user (although they may be recorded in the log)
Protecting information flow through channels
  • Control of information flow at a system level

1. System-based mechanisms

  • E.g., Security Pipeline Interface (SPI):
    o A special processor inserted between a host and a destination

  • E.g., Secure Network Server Mail Guard:
    o A computer that sits between a classified SECRET network and a public network
2. Protocol-based mechanisms

E.g., passwords

Example protocols

- One-Time Password (OTP) e.g., S/Key (Lamport, 1981)

- Public Key based challenge-response authentication
Confinement (Chapter 16)

When a program executes, it interacts with its environment. The security policy allows some interactions and disallows others.

Confinement: preventing processes from taking disallowed actions.

Client/server architecture:
- Client sends request and data to server
- Server does some processing and returns result to client

Stored data can be a cause of leaks

Computation/processing/communication can be a cause of leaks

A process that cannot be observed and cannot communicate with other processes cannot leak information: “total isolation”
- Difficult to achieve and not very useful!
- Try to achieve “isolation” instead: some (controlled) observation and some (controlled) communication

Transitive confinement:
Two techniques for achieving isolation

1.
2.

1. Process is presented with an environment that appears to be a computer running only that process

Simulates the hardware and uses a special operating system (VMM) on which regular OSs can run

2. Environment is provided in which process actions are analyzed to determine if they leak information

Restricts the actions of a process according to a security policy. It may do this in at least two ways:
- Limiting the execution environment:
- Modifying the program:
Achieving confinement (isolation) can be difficult because physical or logical resources may be shared, even among “confined” processes

- Shared resources can lead to covert channels

A covert channel is a path of communication that was not designed to be used for communication

- E.g.:

Can consider covert storage channels and covert timing channels

- E.g.:

Can also consider

- Noiseless covert channels:

- Noisy covert channels:
Goal is to detect and mitigate covert channels

- “Covert flow tree” can be used to determine if a covert channel exists

- Obscuring the amount of resources that a process uses can make a covert channel unusable

Covert channels can be difficult to eliminate, but techniques to decrease the capacity of the covert channel can be effective