



CS 392/ CS 681 - Computer Security

Nasir Memon – Polytechnic University

Module 11 – Information Flow and Confinement Problems



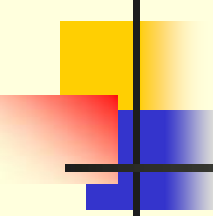
Course Logistics

- Homework 6 due Nov 16th.
- Read chapter 16 and 17 of your text (examples only, like chapters 5, 6 and 7).
- Finals?
- One HW will be dropped.
- We will have total of 8.



Information Flow Policies

- Information flow policies define the way information moves through the system. Deigned to preserve confidentiality and/or integrity.
- Example: BLP model describes a lattice based information flow policy.
- Access controls constrain rights of users but do not fully constrain information flow in a system.
- Compile time and run-time mechanisms needed for checking information flow.



Information Flow – Informal Definition

- What do we mean by information flow?
- Example $y := x$; What is the information flow here? What does knowledge about y tell about x before and after the statement?
- $y := x / z$; What about here?
- A command sequence c causes a flow of information from x to y if uncertainty about x given y decreases after the command sequence c is executed.
- Note: $\{ \text{tmp} := x; y := \text{tmp}; \}$ has information flowing from x to y but no information is flowing from tmp !
- Can be formalized with notion of entropy and conditional entropy.



Denning's requirements

- Information should be able to flow freely among members of a single class – reflexivity.
- Information flow should be transitive.
- BLP exhibits both characteristics.
- However, information flow policies are more general and need not be transitive and even if they are may not be modeled by a lattice.

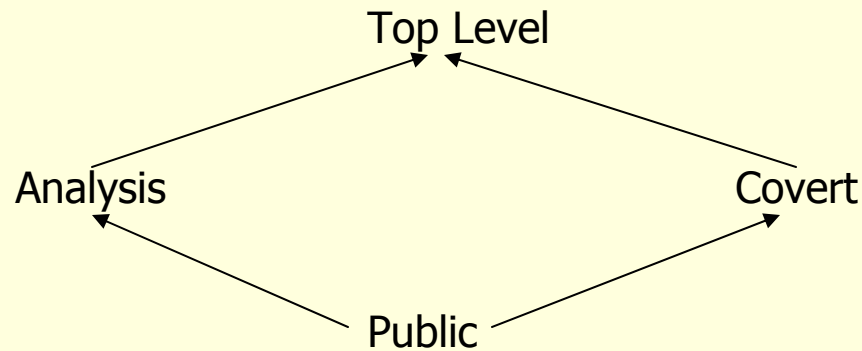


Confinement Flow Models

- Each entity, a , is assigned a confinement pair – $confine(a) = (a_L a_U)$ where a_L is the lowest classification of information allowed to flow out of a and a_U is the highest classification of information allowed to flow into a .
- Example: $Confine(x) = [Confidential, Confidential]$
 $Confine(y) = [Secret, Secret]$
 $Confine(z) = [Confidential, Topsecret]$

Possible flows? Secure or allowed flows? Transitive?

Non-Transitive Information Flow Policy



- Entities: Public Relation Officers (PRO)
Analysts (A) and Spymasters (S)
- Confine Flow: $\text{confine}(\text{PRO}) = [\text{public}, \text{analysis}]$
 $\text{confine}(\text{A}) = [\text{analysis}, \text{top-level}]$
 $\text{confine}(\text{S}) = [\text{covert}, \text{top-level}]$
- $S \leq A$ and $A \leq \text{PRO}$ does not imply $S \leq \text{Pro}$. System not transitive!



Compiler-Based Mechanisms

- Need some language construct to relate variables to security classes.

Example: x : integer class $\{ A B \}$ may mean that security classes A and B may flow into x .

- Assignment statements:

$x := y + z;$

for this to be a secure flow $\text{lub}\{y, z\} \leq x$.

Similarly one can extend to compound statements.



Compiler-Based Mechanisms

- Conditional Statements:

if $x + y < z$ then

$a := b$

else

$d := b * c - x;$

Requirement for secure flow is

1. $b \leq a;$
2. $\text{lub} \{b, c, x\} \leq d$
3. $\text{lub} \{x, y, z\} \leq \text{glb} \{a, d\}$



Execution Based Mechanisms

- Consider

if $x == 1$ then $y := a$ else $y := b$;

Information flows from x and a or x and b to y . But if $a \leq y$ only if some other variable z is 1 then compiler has no way of checking this. Need run time mechanisms.



The Confinement Problem

- The confinement problem is the problem of preventing a service from leaking information that the user of the service considers confidential.
- Example: In a client server application:
 - Server must ensure that resources it accesses on behalf of client are only those the client is authorized to access.
 - Server must ensure that it must not reveal clients data to any unauthorized entity.



Total Isolation

- A process that cannot store data cannot leak it.
- However, things are not that simple. Process can be observed and this may leak information.
- Total isolation – a process that cannot be observed and cannot communicate with other processes cannot leak information.
- Total isolation is hard to achieve with shared computer systems.



Isolation

- One can isolate a process by
 - Present it with an environment that appears to be a computer running only that process or processes to be isolated – virtual machine.
 - An environment is provided in which the process actions are analyzed to determine if they leak information – sandbox.



Virtual Machines

- A virtual machine is a program that simulates the hardware of a (possibly abstract) computer system.
- It runs on a virtual machine monitor that virtualizes the resources of the underlying system and presents to each virtual machine the illusion that it alone is using the hardware.
- One advantage of virtual machines is that existing operating systems need not be modified.



Sandboxes

- A sandbox is an environment in which the actions of a process are restricted according to a security policy.
- Enforcements may be restricted in two ways:
 - Sandbox can limit executable environment by, for example, adding extra security checking mechanisms to the libraries or kernels. Programs themselves do not need to be modified. Java sandbox for downloaded applets.
 - Modify programs to be executed. For example, add instructions to perform memory access checks.



Covert Channels

- A covert channel is a path of communication that was not designed to be used for communication.
- Example: Processes p and q that should not communicate share a file system. To send message P first creates a file named *send* which Q deletes on detection. Then P writes a succession of files named *0bit* or *1bit* either of which Q deletes when it sees. Each file creation conveys 1 bit of information. When P is done, it creates file named *end*.



Storage channel and Timing Channel

- A covert storage channel uses an attribute of the shared resource. A covert timing channel uses a temporal or ordering relationship among accesses to a shared resource.
- Examples:
 - Timing attack on RSA
 - Leak information by using or not using allotted time slice.
- Sometimes distinction between storage and timing channel is blurred – example in section 17.3

Detection of Covert Channel

- To detect covert channels one can examine what resources are being shared – Kemmerer Shared Resource Matrix Methodology.
- Example: Multi-level security model. Each file has 4 attributes. Two subjects – High and Low. Read succeeds if file exists and label permits access. Same for writing and deleting. Creation succeeds if no file with that name exists. File gets creator's label.

| | read | write | delete | create |
|----------------|------|-------|--------|--------|
| File existence | R | R | R, M | R, M |
| File owner | | | R | M |
| File label | R | R | R | M |
| File size | R | M | M | M |

Shared Resource Matrix

R – means attribute is read
M – means attribute is modified.



Checking for Covert Channels

- The following properties must hold for a storage channel to exist:
 1. Both sending and receiving process must have access to the same attribute of a shared object.
 2. The sending process must be able to modify the attribute of the shared object.
 3. The receiving process must be able to reference that attribute of the shared object.
 4. A mechanism for initiating both processes and properly sequencing their respective accesses to the shared resource must exist.
- Similar properties for timing channel can be listed – see text.



Mitigating Covert Channels

- Total isolation – declare all resources prior to execution which are then solely allocated to process and released when process terminates. Difficult to achieve in practice.
- Obscure the amount of resources a process uses.
 - By making usage uniform - For example, fixed time slice allotted whether process uses it or not.
 - By injecting randomness.
 - Both affect efficiency.
- Use a “pump” that controls communication path between Low process and High process. See text.