Covert Channel Analysis

John McHugh
Portland State University
mchugh@cs.pdx.edu

Covert Channels

- A covert channel is an information flow mechanism within a system that is based on the use of system resources not normally intended for communication between the users of the system.
  - Storage Channels use system variables and attributes (other than time) to signal information, e.g. – file status
  - Timing Channels vary the amount of time required to complete a particular task, e.g. – influencing the number of CPU cycles available to a given process in a given time frame.
  - The division is not a hard and fast one. Some channels can be characterized as either.

Storage Channels

a) The sending and receiving entities must have access to the same shared resource or attribute.

b) There must be some means by which the sending entity can force the shared resource or attribute to change.

c) There must be some means by which the receiving entity can detect the change.

d) There must be some mechanism for initiating the communication between the sending and receiving entities and for sequencing the events correctly. This mechanism could be a covert channel with a lower bandwidth.

If a-c are satisfied, one must find a scenario that satisfies d. If such a scenario exists, a storage channel exists.

Timing Channels

a) The sending and receiving entities must have access to the same shared resource or attribute.

b) The sending and receiving entities must have access to a time reference such as a real-time clock.

c) The sender must be capable of modulating the receiver's perception of time for detecting a change in the shared resource or attribute.

d) There must be some mechanism for initiating the channel and for sequencing the events transmitted over it.

Any time a processor or memory is shared, there is a shared attribute. A change in response time is detected by the receiving process by means of monitoring the clock or its surrogate.

Shared and Private Resources

- The identification of shared and private resources is a key portion of any covert Channel analysis.
  - If resources that are private are considered to be shared, the analysis is greatly complicated.
  - If resources that are assumed to be private are shared the analysis is incomplete.

  In the case of systems with informal specifications such as those defined in natural languages, it is often not clear which resources are intended to be shared.
- Even formal specifications do not solve the problem unless adequate conventions are imposed by specifiers.
- Often additional shared resources are introduced during the implementation of a system and do not show up in the specifications analyzed for covert channels.

Harmful Covert Channels

- Innocuous Cases
  - The channel parallels an Overt (or legal) Channel
  - The sender and receiver are the same (It's OK to talk to yourself)
- Dangerous Case
  - The sender and receiver are not permitted to communicate under the security policy of the system.

Coding and Signaling

- Since the value of a resource modulated in a Covert Channel is not the message being sent, the sender and receiver must agree on a suitable encoding of the message to be sent.
- If a limited vocabulary will suffice, a few bits will do a lot of damage.
- When a covert channel is identified, bandwidth analysis is important, but the analysis must take into account the nature of the information that might be compromised.
  - Work done at Bell Labs by Shannon is the root of such analysis.
- Since the typical Covert Channel scenario involves a Trojan Horse program at either end, the question of encoding and the nature of the information that might be compromised requires especial attention.

Manual Analysis of Specifications

- The basic problem is to identify the interactions between the user and the TCB as well as the effect that each interaction has on the internal state of the TCB.
  - DTLS or equivalent should identify the operations. Check to make sure that it is complete. Look at the implementation if possible.
  - Determine what the effect of each operation is on the data components of the TCB. Note that conditional operations pass information from the condition determining components to those that are assigned. Note attributes like status bits.
  - Determine what information is returned to the caller and where it comes from again noting conditional flows. Pay especial attention to error and abnormal returns.

SRM – àla Kemmerer

- The SRM presents
  - System resources and attributes
  - System operations
- Rows contain:
  - System resources and their attributes
- Columns contain:
  - System operations
- Each cell of the matrix indicates whether a given operation reads or modifies a system resource

SRM - Continued

- This provides a concise and comprehensive display of data and operational dependencies in the system under analysis.
- The SRM is suitable for both storage and timing channel analysis provided timing resources are identified
- It provides a means of gaining a global insight into system interdependencies
- The methodology can be applied to
  - Informal descriptions
  - Formal specifications
  - Code

Application

a) Identify and list shared resources and their attributes.
b) Identify and list primitive system operations.
c) Determine which primitives reference and/or modify which shared resources.
d) Represent the results of (c) as a matrix.
e) Make indirect operations visible by generating transitive closure of the matrix.
f) Identify the channels represented in the matrix.
g) Categorize the channels found:
   1 – The channel is a legal one
   2 – No useful information can be gained from this channel
   3 – The sending and receiving process are the same
   4 – It represents a genuine covert channel!

The Shared Resource Matrix (SRM)

- The results of the dependency analysis efforts can be presented in a variety of forms. Collectively, we will refer to these as Shared Resource Matrices (SRMs).
- The details presented in SRMs can go beyond those described in Kemmerer's paper.
- Consider the following fragment of code:

```
A := B;
C := if D then E
          else F;
```

The Basic SRM

In the style of [Kemmerer 83], this would lead to an SRM of the form:

```
A := B;
C := if D then E
          else F;
```

Operation	
Resource Attribute	OP1
A	M
B	R
C	M
D	R
E	R
F	R

Adding Precision to the SRM

- The basic SRM crowds too much information into too little space. It can be extended to make its representation of information flow more precise.
  - Factoring independent information flows that occur within the same operation (operation splitting)
  - Recognizing that conditional flows take only one branch at a time (Guard Expansion)
  - Making explicit flows between private and shared resources (User Flows)

Operation Splitting

Consideration of this SRM gives the impression that information flows from B to C and from D,E, and F to A.

Examination of the code shows that no such flows exist. We can fix this by separating the individual operations into separate columns.

```
A := B;
C := if D then E
          else F;
```

	Operation
Resource Attribute	OP1
A	M
B	R
C	M
D	R
E	R
F	R

Operation Splitting Results

```
A := B;
C := if D then E
          else F;
```

	Operation	
Resource Attribute	OP1	OP1
A	M	
B	R	
C		M
D		R
E		R
F		R

Guard Expansion

Consideration of this SRM still gives a false impression that information always flows from D to C and from E to C.

Examination of the code shows that one or the other occurs; not both at the same time. Again, we can split the cases.

```
A := B;
C := if D then E
         else F;
```

		Operation
Resource Attribute	**OP1**	**OP1**
A	**M**	
B	**R**	
C		**M**
D		**R**
E		**R**
F		**R**

Guard Expansion Results

```
A := B;
C := if D then E
         else F;
```

	Operation		
Resource Attribute		**OP1**	
	G1	**G2**	**G3**
A	**M**		
B	**R**		
C		**M**	**M**
D		**R**	**R**
E		**R**	
F			**R**

Legend

G1 = True

G2 = D

G3 = not D

Flows from Private to Shared Resources

Our example code fragment is not part of a system routine. Usually we deal with kernelized systems in which users communicate with the system and the outside world via calls on kernel routines. Data contained in the users space is considered to be a private resource. Information about it can be transferred to shared resources via:

- User Parameters or
- The act of making a kernel call

Flows from Shared to Private Resources

Information about shared resources is transferred to the users private resources as a result of returns from these calls. These can take the form of:

- Return values
- Modified User Parameters or other data
- Error signals and exceptions

Adding User Flows

We can indicate these flows, called *user flows* in the SRM. In this example, we assume that the call to the routine containing the fragment returns no distinct information to its caller.

```
A := B;
C := if D then E
          else F;
```

Legend
- **G1 = True**
- **G2 = D**
- **G3 = not D**

Resource Attribute	Operation		
		OP1	
	G1	G2	G3
A	M		
B	R		
C		M	M
D		R	R
E		R	
F			R
User In	R	R	R
User Out			

Searching for Covert Channels

- The tools that we will discuss later support a mechanized analysis of the SRM. This is based on an assignment of security levels to the internal components of the state.
- We want to develop an intuitive approach to the analysis.
  - Look at cases where a modify is done by one operation and a read by another.
  - Look at resources that are not polyinstantiated on a per subject or per security level
  - Try to find a scenario where a modification by one subject could be detected by another.
- It takes practice. Think nasty devious thoughts. Look at things upside down, inside out, and backwards. Forget what the system is supposed to do. Concentrate on what you can make it do. Look for combinations of actions. Only one weak link is needed

So You Found a Channel?

- Once you have found a real live covert channel, the question is what can it do. We will discuss this in more detail later, but there are two issues:
  - How much can it carry
  - How fast can it carry it
- The answer to the first question depends on the freedom that the user has in manipulating the channel. Figure Log2(choices) per execution of the scenario. For many channels, this may be a very small number of bits. It can be less than 1.
- The answer to the second question is a function of the system speed. How many complete scenarios per second can be executed?
- These answers are tempered by several factors. If other system activity reduces the probability of a clear signal being received, the capacity is reduced. On the other hand, if the sender and receiver have an efficient code (1 if by land, 2 if by sea), only a few bits (total) are needed.

Dependencies

A Dependency is a relationship that supports the ability to make a fuzzy inference.

- Strong dependencies allow near-certainty of inference
- Weak dependencies admit uncertainty in the inference but:
  - Statistical techniques can improve the certainty over time and redundancy
  - Global knowledge can reduce uncertainty

Dependency Examples

- `A : = B;`

  If we observe A, we can be sure that we know B exactly
- `A : = If  X  then  B  else  C  Fi;`

  If we observe A, we know B or C exactly. If we know X, we know whether its B or C

  If we observe A and know B, we may know C (unless it equals B)

  If we observe A and know B and C, we can determine X exactly

More Examples

- `A(I) : = B`

  If we know that no element of A equaled B prior to the assignment, and we know B, then we know I if we look at all the elements of A
- `A : = B + Rand (-1,1)`

  Let Rand be a random noise source uniformly distributed over [-1..1]. If we assume that B changes slowly and we observe A sufficiently often, we can determine B.

Specification Decomposition

- The objective of specification decomposition is to derive a set of dependencies for each system operation.
- Each dependency contains:
  - A single system resource that is the target of the information transfer
  - A boolean expression based on the values of system resources that is TRUE when and only when the transfer takes place
  - A list of system resources that are sources of the information transferred to the target

Operation Splitting

- We want one target per dependency. Some operations affect more than one resource.
  - Assigning a structure to a similar structure can be viewed as a set of parallel assignments, each with a distinct source and targets
  - Assignment of a single source to multiple targets can be viewed as multiple assignments
  - Independent assignments are no problem
  - Need to look for side effects (size changes, etc.) and split them out too.

Dependencies as Guarded Assignments

- As an intermediate step, specification decomposition can be viewed as producing a set of guarded assignment statements of the form.
  - IF (G) Then T : = S
    - G is the boolean guard expression
    - T is the target resource expression
    - S is a source expression
- All the guards taken together completely cover the input space of the operation.

Guards

- The conditions under which flows occur are determined by guards. Guards can arise in two ways:
  - Explicit statements in the specifications

```
If OK_TO (write, file, user)
   Then apnd (file, data)
```

  - Implicit conditions associated with transformations of specifications as when an array element is a target.

```
A[i] := x is A := A’ with ([I] : = x);
```

  - Which becomes

```
A[I#1] : =
   if(I#1 =I)Then X Else = A’[I#1]  Fi;
```

From Guarded Assignments to Dependencies

- Once the specification has been transformed into guarded assignments, we can determine the dependencies. Each dependency is a triple
  (T, {S}, G) where
  - T is the target of the dependency
  - {S} is the set of sources of the dependency
  - G is the guard expression of the dependency

Side Effects

- Operations on structures can have side effect. In the Gypsy Information Flow Tool, we introduced the concept of "Name Resources."
  - The array case `A(I) : = B` can be viewed as an assignment to both the element and the static name resource
    - `A(I) <- B, I`
    - `Name(A) <- I`
  - For dynamic structure, other resources may be involved. Consider the sequence operation `A : = B @ C`
    - `A <- B, C`
    - `Size (A) <- Size (B), Size (C)`

Example

`If A.LEV ≥ B.LEV Then A.CNTS(I) := B.CNTS(j)`

- Target Expression `A.CNTS(i)`
- Source Expression `B.CNTS(I)`
- Guard Expression `A.LEV ≥ B.LEV`

Covert Channels

Example contd.

- This gives rise to two dependencies.
  - Dependency 1 is the element modification
    - Target - `A.CNTS(I)`
    - Source Sources `B.CNTS(J), J, Name(B.CNTS)`
    - Guard Sources - `A.LEV, B.LEV`
    - Target Source - `I`
  - Dependency 2 is the Array modification
    - Target - `Name(A.CNTS)`
    - Source Sources `B.CNTS(J), J, Name(B.CNTS)`
    - Guard Sources - `A.LEV, B.LEV`
    - Target Source - `I`

Information Flow Techniques

- Theoretical Considerations
- Security Labeling of Resources
- Identification of Sources and Targets
- Structured Objects and Indirect Flows
- Information Flow Formulae
- Relation to Bell and LaPadula Security Models

Theoretical Considerations

- It is popular to view secure computer systems as a kernelized state machine.
  - A good match for most operating systems
  - Well-developed theory of state machines exists
  - Early specification systems (HDM, FDM) based on state machine model

Security Policies And Labeling

- Since covert channels refer to the flow of information contrary to some established policy, such a policy must be identified. The policy must characterize secure information flows in a manner which distinguishes them from insecure flows.
  - A set of security attributes (e.g. levels, categories, domains, etc.)
  - A comparison function for the attributes that determines when information can flow from an entity having one attribute value to one having another attributes value
  - A method of assigning security attributes to all system resources

Labeling

- Every system resource must be labeled.
  - This makes sense for things that normally hold information.
  - It is artificial for others.
    - What is the level of a label, a file name, a user-ID
    - Appropriate labeling is often the key to successful covert channel analysis.

A Security Policy Example

Consider our fragment of code:

```
A := B;
C := if D then E
          else F;
```

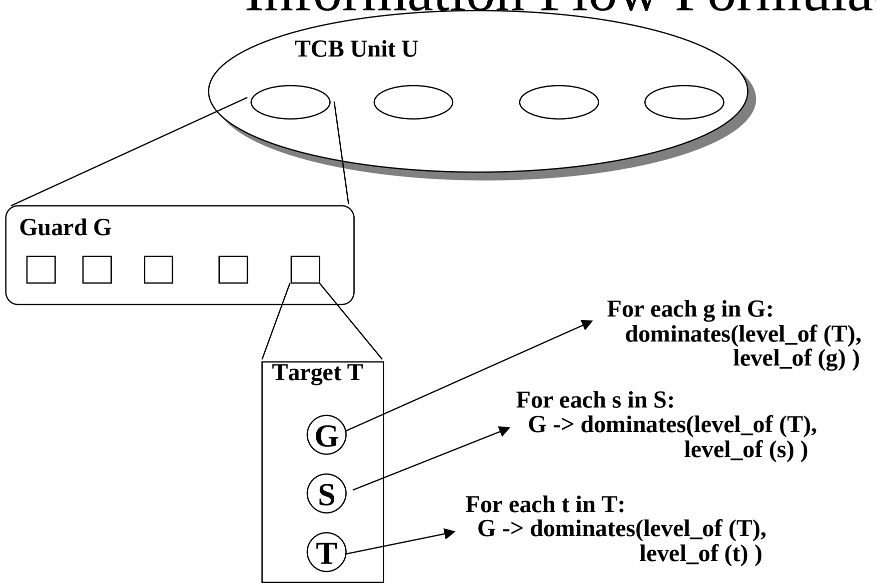
- Security Attributes:
  - L1, L2, & L3, where (L1 < L2 < L3)
- Security Relation:
  - GE (Greater Than or Equal To)
- Level Assignment:

  A: L3 D: L1
  B: L2 E: L2
  C: L2 F: L3

A General Analysis Paradigm

- Assume specifications describe a state machine. All system resources are represented by the state. All operations are represented by actions that alter the state and/or return information about the state.
- The specification is transformed by reducing the effects of each action to a series of guarded assignment statements operating on the primitive components of the state.
- Identify information flow targets and sources. For each target, determine its source. These results are available for a variety of forms of analysis
  - Shared Resource Matrices
  - Information Flow Formula

Information Flow Formulae



An Example of Information Flow Formulae

G1:	
$L3 \geq L2$	level_of(A) ≥ level_of(B)
G2:	
$L2 \geq L1$	level_of(C) ≥ level_of(D)
$D \rightarrow L2 \geq L2$	if guard is true, level_of(C) ≥ level_of(E)
G3:	
$L2 \geq L1$	level_of(C) ≥ level_of(D))
$\neg D \rightarrow L2 \geq L3$	if guard is false, level_of(C) ≥ level_of(F)

Remember: $L1 < L2 < L3$

Relationship Between SVCs and SRMs

- SRMs are a fairly direct representations of the results of the dependency analyses. The rows represents the canonical state leaves and the columns represent i/o with the user. The transitive closure represents indirect references to system resources.
- SVCs are putative theorems that are related to the non-transitive reads in the SRM; however, this is not a one to one correspondence.
- When looking at the most refined SRM representations of dependencies, one SVC would be generated for each non-transitive read indicated
  - In practice, many of these SVCs reduce to the same expression

Relationship to Bell and LaPadula

- Bell and LaPadula models consider only information containing "objects" as in need of protection.
  - Our model is similar, but views every system, resource as an "object" with the potential for containing information.
  - In addition, we are concerned with actual transfers of information, rather than permission for access.