March 5, 2014

- Covert channels
- Detection
- Mitigation
Noisy vs. Noiseless

• Noiseless: covert channel uses resource available only to sender, receiver
• Noisy: covert channel uses resource available to others as well as to sender, receiver
  – Idea is that others can contribute extraneous information that receiver must filter out to “read” sender’s communication
Key Properties

• *Existence*: the covert channel can be used to send/receive information

• *Bandwidth*: the rate at which information can be sent along the channel

• Goal of analysis: establish these properties for each channel
  – If you can eliminate the channel, great!
  – If not, reduce bandwidth as much as possible
Step #1: Detection

• Manner in which resource is shared controls who can send, receive using that resource
  – Shared Resource Matrix Methodology
  – Information flow analysis
  – Covert flow trees
SRMM

• Shared Resource Matrix Methodology
• Goal: identify shared channels, how they are shared
• Steps:
  – Identify all shared resources, their visible attributes [rows]
  – Determine operations that reference (read), modify (write) resource [columns]
  – Contents of matrix show how operation accesses the resource
Example

- Multilevel security model
- File attributes:
  - existence, owner, label, size
- File manipulation operations:
  - read, write, delete, create
  - create succeeds if file does not exist; gets creator as owner, creator’s label
  - others require file exists, appropriate labels
- Subjects:
  - High, Low
### Shared Resource Matrix

<table>
<thead>
<tr>
<th></th>
<th>read</th>
<th>write</th>
<th>delete</th>
<th>create</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>existence</strong></td>
<td>R</td>
<td>R</td>
<td>R, M</td>
<td>R, M</td>
</tr>
<tr>
<td><strong>owner</strong></td>
<td></td>
<td></td>
<td>R</td>
<td>M</td>
</tr>
<tr>
<td><strong>label</strong></td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>M</td>
</tr>
<tr>
<td><strong>size</strong></td>
<td>R</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>
Covert Storage Channel

- Properties that must hold for covert storage channel:
  1. Sending, receiving processes have access to same *attribute* of shared object;
  2. Sender can modify that attribute;
  3. Receiver can reference that attribute; and
  4. Mechanism for starting processes, properly sequencing their accesses to resource
Example

- Consider attributes with both R, M in rows
- Let High be sender, Low receiver
- create operation both references, modifies existence attribute
  - Low can use this due to semantics of create
- Need to arrange for proper sequencing accesses to existence attribute of file (shared resource)
Use of Channel

- 3 files: ready, done, 1bit
- Low creates ready at High level
- High checks that file exists
  - If so, to send 1, it creates 1bit; to send 0, skip
  - Delete ready, create done at High level
- Low tries to create done at High level
  - On failure, High is done
  - Low tries to create 1bit at level High
- Low deletes done, creates ready at High level
Covert Timing Channel

• Properties that must hold for covert timing channel:
  1. Sending, receiving processes have access to same attribute of shared object;
  2. Sender, receiver have access to a time reference (wall clock, timer, event ordering, ...);
  3. Sender can control timing of detection of change to that attribute by receiver; and
  4. Mechanism for starting processes, properly sequencing their accesses to resource
Example

- Revisit variant of KVM/370 channel
  - Sender, receiver can access ordering of requests by disk arm scheduler (attribute)
  - Sender, receiver have access to the ordering of the requests (time reference)
  - High can control ordering of requests of Low process by issuing cylinder numbers to position arm appropriately (timing of detection of change)
  - So whether channel can be exploited depends on whether there is a mechanism to (1) start sender, receiver and (2) sequence requests as desired
Uses of SRM Methodology

- Applicable at many stages of software life cycle model
  - Flexibility is its strength
- Used to analyze Secure Ada Target
  - Participants manually constructed SRM from flow analysis of SAT model
  - Took transitive closure
  - Found 2 covert channels
    - One used assigned level attribute, another assigned type attribute
Summary

• Methodology comprehensive but incomplete
  – How to identify shared resources?
  – What operations access them and how?

• Incompleteness a benefit
  – Allows use at different stages of software engineering life cycle

• Incompleteness a problem
  – Makes use of methodology sensitive to particular stage of software development
Measuring Capacity

• Intuitively, difference between unmodulated, modulated channel
  – Normal uncertainty in channel is 8 bits
  – Attacker modulates channel to send information, reducing uncertainty to 5 bits
  – Covert channel capacity is 3 bits
    • Modulation in effect fixes those bits
Formally

• Inputs:
  – A input from Alice (sender)
  – V input from everyone else
  – X output of channel
• Capacity measures uncertainty in X given A
• In other terms: maximize
  \[ I(A; X) = H(X) - H(X | A) \]
  with respect to A
Example

- If $A$, $V$ independent, $p = p(A=0)$, $q = p(V=0)$:
  - $p(A=0, V=0) = pq$
  - $p(A=1, V=0) = (1-p)q$
  - $p(A=0, V=1) = p(1-q)$
  - $p(A=1, V=1) = (1-p)(1-q)$

- So
  - $p(X=0) = p(A=0, V=0) + p(A=1, V=1) = pq + (1-p)(1-q)$
  - $p(X=1) = p(A=0, V=1) + p(A=1, V=0) = (1-p)q + p(1-q)$
More Example

• Also:
  – \( p(X=0|A=0) = q \)
  – \( p(X=0|A=1) = 1-q \)
  – \( p(X=1|A=0) = 1-q \)
  – \( p(X=1|A=1) = q \)

• So you can compute:
  – \( H(X) = -[(1-p)q + p(1-q)] \log [(1-p)q + p(1-q)] \)
  – \( H(X|A) = -q \log q - (1-q) \log (1-q) \)
  – \( I(A;X) = H(X) - H(X|A) \)
\[ I(A;X) \]

\[
I(A; X) = - [pq + (1 - p)(1 - q)] \lg [pq + (1 - p)(1 - q)] - \\
[(1 - p)q + p(1 - q)] \lg [(1 - p)q + p(1 - q)] + \\
q \lg q + (1 - q) \lg (1 - q)
\]

- Maximum when \( p = 0.5 \); then

\[
I(A;X) = 1 + q \lg q + (1- q) \lg (1-q) = 1 - H(V)
\]

- So, if \( V \) constant, \( q = 0 \), and \( I(A;X) = 1 \)

- Also, if \( q = p = 0.5 \), \( I(A;X) = 0 \)
Analyzing Capacity

- Assume a noisy channel
- Examine covert channel in MLS database that uses replication to ensure availability
  - 2-phase commit protocol ensures atomicity
  - *Coordinator* process manages global execution
  - *Participant* processes do everything else
How It Works

• Coordinator sends message to each participant asking whether to abort or commit transaction
  – If any says “abort”, coordinator stops

• Coordinator gathers replies
  – If all say “commit”, sends commit messages back to participants
  – If any says “abort”, sends abort messages back to participants
  – Each participant that sent commit waits for reply; on receipt, acts accordingly
Exceptions

- Protocol times out, causing party to act as if transaction aborted, when:
  - Coordinator doesn’t receive reply from participant
  - Participant who sends a commit doesn’t receive reply from coordinator
Covert Channel Here

- Two types of components
  - One at Low security level, other at High
- Low component begins 2-phase commit
  - Both High, Low components must cooperate in the 2-phase commit protocol
- High sends information to Low by selectively aborting transactions
  - Can send abort messages
  - Can just not do anything
Note

- If transaction *always* succeeded except when *High* component sending information, channel not noisy
  - Capacity would be 1 bit per trial
  - But channel noisy as transactions may abort for reasons *other* than the sending of information
Analysis

• **X** random variable: what *High* user wants to send
  - Assume abort is 1, commit is 0
  - $p = p(X = 0)$ probability *High* sends 0
• **A** random variable: what *Low* receives
  - For noiseless channel $X = A$
• $n + 2$ users
  - Sender, receiver, $n$ others
  - $q$ probability of transaction aborting at any of these $n$ users
Basic Probabilities

- Probabilities of receiving given sending:
  - $p(A=0 \mid X=0) = (1-q)^n$
  - $p(A=1 \mid X=0) = 1 - (1-q)^n$
  - $p(A=0 \mid X=1) = 0$
  - $p(A=1 \mid X=1) = 1$

- So probabilities of receiving values:
  - $p(A=0) = p(1-q)^n$
  - $p(A=1) = 1 - p(1-q)^n$
More Probabilities

- Given sending, what is receiving?
  - \( p(X=0 \mid A=0) = 1 \)
  - \( p(X=1 \mid A=0) = 0 \)
  - \( p(X=0 \mid A=1) = p[1-(1-q)^n] / [1-p(1-q)^n] \)
  - \( p(X=1 \mid A=1) = (1-p) / [1-p(1-q)^n] \)
Entropies

- \( H(X) = -p \log p - (1-p) \log (1-p) \)
- \( H(X | A) = -p[1-(1-q)^n] \log p - p[1-(1-q)^n] \log [1-(1-q)^n] + [1-p(1-q)^n] \log [1-p(1-q)^n] - (1-p) \log (1-p) \)
- \( I(A;X) = -p(1-q)^n \log p + p[1-(1-q)^n] \log [1-(1-q)^n] - [1-p(1-q)^n] \log [1-p(1-q)^n] \)
Capacity

• Maximize this with respect to $p$ (probability that High sends 0)
  – Notation: $m = (1–q)^n$, $M = (1–m)^{1–m}$
  – Maximum when $p = \frac{M}{Mm+1}$

• Capacity is:

\[
I(A;X) = Mm \log p + M(1–m) \log (1–m) + \log (Mm+1)
\]

\[
\frac{(Mm+1)}{(Mm+1)}
\]
Mitigation of Covert Channels

• Problem: these work by varying use of shared resources

• One solution
  – Require processes to say what resources they need before running
  – Provide access to them in a way that no other process can access them

• Cumbersome
  – Includes running (CPU covert channel)
  – Resources stay allocated for lifetime of process
Alternate Approach

• Obscure amount of resources being used
  – Receiver cannot distinguish between what the sender is using and what is added

• How? Two ways:
  – Devote uniform resources to each process
  – Inject randomness into allocation, use of resources
Uniformity

- Variation of isolation
  - Process can’t tell if second process using resource
- Example: KVM/370 covert channel via CPU usage
  - Give each VM a time slice of fixed duration
  - Do not allow VM to surrender its CPU time
    - Can no longer send 0 or 1 by modulating CPU usage