ECS 235B Module 55
Mitigating Covert Channels
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
Randomness

• Make noise dominate channel
  • Does not close it, but makes it useless

• Example: MLS database
  • Probability of transaction being aborted by user other than sender, receiver approaches 1
    • $q \to 1$
  • $I(A; X) \to 0$
  • How to do this: resolve conflicts by aborting increases $q$, or have participants abort transactions randomly
Problem: Loss of Efficiency

- Fixed allocation, constraining use
  - Wastes resources
- Increasing probability of aborts
  - Some transactions that will normally commit now fail, requiring more retries
- Policy: is the inefficiency preferable to the covert channel?
Example

• Goal: limit covert timing channels on VAX/VMM
• “Fuzzy time” reduces accuracy of system clocks by generating random clock ticks
  • Random interrupts take any desired distribution
  • System clock updates only after each timer interrupt
  • Kernel rounds time to nearest 0.1 sec before giving it to VM
    • Means it cannot be more accurate than timing of interrupts
Example

• I/O operations have random delays

• Kernel distinguishes 2 kinds of time:
  • *Event time* (when I/O event occurs)
  • *Notification time* (when VM told I/O event occurred)
    • Random delay between these prevents VM from figuring out when event actually occurred
    • Delay can be randomly distributed as desired (in security kernel, it’s 1–19ms)
  • Added enough noise to make covert timing channels hard to exploit
Improvement

• Modify scheduler to run processes in increasing order of security level
  • Now we’re worried about “reads up”, so ...

• Countermeasures needed only when transition from dominating VM to dominated VM
  • Add random intervals between quanta for these transitions
The Pump

• Tool for controlling communications path between High and Low
Details

• Communications buffer of length \( n \)
  • Means it can hold up to \( n \) messages
• Messages numbered
• Pump ACKs each message as it is moved from \( High \) (\( Low \)) buffer to communications buffer
• If pump crashes, communications buffer preserves messages
  • Processes using pump can recover from crash
Covert Channel

• Low fills communications buffer
  • Send messages to pump until no ACK
  • If High wants to send 1, it accepts 1 message from pump; if High wants to send 0, it does not
  • If Low gets ACK, message moved from Low buffer to communications buffer ⇒ High sent 1
  • If Low doesn’t get ACK, no message moved ⇒ High sent 0

• Meaning: if High can control rate at which pump passes messages to it, a covert timing channel
Performance vs. Capacity

• Assume Low process, pump can process messages more quickly than High process
• $L_i$ random variable: time from Low sending message to pump to Low receiving ACK
• $H_i$ random variable: average time for High to ACK each of last $n$ messages
Case 1: $E(L_i) > H_i$

- *High* can process messages more quickly than *Low* can get ACKs
- Contradicts above assumption
  - Pump must be delaying ACKs
  - *Low* waits for ACK whether or not communications buffer is full
- Covert channel closed
- Not optimal
  - Process may wait to send message even when there is room
Case 2: $E(L_i) < H_i$

- Low sending messages faster than High can remove them
- Covert channel open
- Optimal performance
Case 3: $E(L_i) = H_i$

- Pump, processes handle messages at same rate
- Covert channel open
  - Bandwidth decreased from optimal case (can’t send messages over covert channel as fast)
- Performance not optimal
Adding Noise

• Shown: adding noise to approximate case 3
  • Covert channel capacity reduced to $1/nr$ where $r$ time from Low sending message to pump to Low receiving ACK when communications buffer not full
  • Conclusion: use of pump substantially reduces capacity of covert channel between High, Low processes when compared to direct connection