ECS 235B, Lecture 26

March 13, 2019

State Machine Model: 2-Bit Machine

Levels *High*, *Low*, meet 4 properties:

1. For every input i_k, state σ_j, there is an element $c_m \in C^*$ such that $T^*(c_m, \sigma_j) = \sigma_n$, where $\sigma_n \neq \sigma_j$

T^* is total function, inputs and commands always move system to a different state

Property 2

2. There is an equivalence relation $\equiv$ such that:
   a. If system in state σ_i and HIGH sequence of inputs causes transition from σ_i to σ_j, then $\sigma_i \equiv \sigma_j$
      - 2 states equivalent if either reachable from the other state using only HIGH commands
   b. If $\sigma_i \equiv \sigma_j$ and LOW sequence of inputs i_1, …, i_n causes system in state σ_i to transition to σ_i', then there is a state σ_j' such that $\sigma_i' \equiv \sigma_j'$ and inputs i_1, …, i_n cause system in state σ_j to transition to σ_j'
      - States resulting from giving same LOW commands to the two equivalent original states have same LOW projection

$\equiv$ holds if LOW projections of both states are same

- If 2 states equivalent, HIGH commands do not affect LOW projections

Property 3

- Let $\sigma_i \equiv \sigma_j$. If sequence of HIGH outputs o_1, …, o_n indicate system in state σ_i transitioned to state σ_i', then for some state σ_j' with $\sigma_j' \equiv \sigma_i'$, sequence of HIGH outputs o_1', …, o_m' indicates system in σ_j transitioned to σ_j'
  - HIGH outputs do not indicate changes in LOW projection of states

Property 4

- Let $\sigma_i \equiv \sigma_j$, let *c*, *d* be HIGH output sequences, *e* a LOW output. If output sequence *ced* indicates system in state σ_i transitions to σ_i', then there are HIGH output sequences *c*' and *d*' and state σ_j' such that *c*'*ed*' indicates system in state σ_j transitions to state σ_j'
  - Intermingled LOW, HIGH outputs cause changes in LOW state reflecting LOW outputs only

Restrictiveness

- System is *restrictive* if it meets the preceding 4 properties

Composition

- Intuition: by 3 and 4, HIGH output followed by LOW output has same effect as the LOW input, so composition of restrictive systems should be restrictive

Composite System

- System M_1's outputs are acceptable as M_2's inputs
- μ_{1i}, μ_{2i} states of M_1, M_2
- States of composite system pairs of M_1, M_2 states (μ_{1i}, μ_{2i})
- *e* event causing transition
- *e* causes transition from state (μ_{1a}, μ_{2a}) to state (μ_{1b}, μ_{2b}) if any of 3 conditions hold

Conditions

1. M_1 in state μ_{1a} and *e* occurs, M_1 transitions to μ_{1b}; *e* not an event for M_2; and $\mu_{2a} = \mu_{2b}$
2. M_2 in state μ_{2a} and *e* occurs, M_2 transitions to μ_{2b}; *e* not an event for M_1; and $\mu_{1a} = \mu_{1b}$
3. M_1 in state μ_{1a} and *e* occurs, M_1 transitions to μ_{1b}; M_2 in state μ_{2a} and *e* occurs, M_2 transitions to μ_{2b}; *e* is input to one machine, and output from other

Intuition

- Event causing transition in composite system causes transition in at least 1 of the components
- If transition occurs in exactly 1 component, event must not cause transition in other component when not connected to the composite system

Equivalence for Composite

- Equivalence relation for composite system

$$(\sigma_a, \sigma_b) \equiv_C (\sigma_c, \sigma_d) \text{ iff } \sigma_a \equiv \sigma_c \text{ and } \sigma_b \equiv \sigma_d$$

- Corresponds to equivalence relation in property 2 for component system

Theorem

The system resulting from the composition of two restrictive systems is itself restrictive

Side Channels

A *side channel* is set of characteristics of a system, from which adversary can deduce confidential information about system or a competition

- Consider information to be derived as HIGH
- Consider information obtained from set of characteristics as LOW
- Attack is to deduce HIGH values from LOW values only
- Implication: attack works on systems not deducibly secure

Types of Side Channel Attacks

- *Passive*: Only observe system; deduce results from observations
- *Active*: Disrupt system in some way, causing it to react; deduce results from measurements of disruption

Example: Passive Attack

- Fast modular exponentiation:

```
x := 1; atmp := a;
for i := 0 to k-1 do begin
   if z_i = 1 then
         x := (x * atmp) mod n;
   atmp := (atmp * atmp) mod n;
end;
result := x;
```

- If bit is 1, there are 2 multiplications; if it is 0, only one
- Extra multiplication takes time
- Can determine bits of the confidential exponent by measuring computation time

Example: Active Attack

Background

- Derive information from characteristics of memory accesses in chip
- Intel x86 caches
  - Each core has 2 levels, L1 and KL2
  - Chip itself has third cache (L3 or LLC)
  - These are hierarchical: miss in L1 goes to L2, miss in L2 goes to L3, miss in L3 goes to memory
  - Caches are inclusive (so L3 has copies of data in L2 and L1)
- Processes share pages

Example: Active Attack

Phase 1

- Flush a set of bytes (called a *line*) from cache to clear it from all 3 caches
  - The disruption

Phase 2

- Wait until victim has chance to access that memory line

Phase 3

- Reload the line
  - If victim did this already, time is short as data comes from L3 cache
  - Otherwise time is longer as memory fetch is required

Example: Active Attack

What happened

- Used to trace execution of GnuPG on a physical machine
- Derived bits of a 2048 bit private key; max of 190 bits incorrect
- Repeated experiment on virtual machine
- Error rates increased
  - On one system, average error rate increased from 1.41 bits to 26.55 bits
  - On another system, average error rate increased from 25.12 bits to 66.12 bits

Model

Components

- *Primitive*: instantiation of computation
- *Device*: system doing the computation
- *Physical observable*: output being observed
- *Leakage function*: captures characteristics of side channel and mechanism to monitor the physical observables
- *Implementation function*: instantiation of both device, leakage function
- *Side channel adversary*: algorithm that queries implementation to get outputs from leakage function

Example

- First one (passive attack) divided leakage function into two parts
  - *Signal* was variations in output due to bit being derived
  - *Noise* was variations due to other factors (imprecisions in measurements, etc.)
- Second one (active attack) had leakage function acting in different ways
  - Physical machine: one chip used more advanced optimizations, thus more noise
  - Virtual machine: more variations due to extra computations running the virtual machines, hence more noise

Example: Electromagnetic Radiation

- CRT video display produces radiation that can be measured
- Using various equipment and a black and white TV, van Eck could reconstruct the images
  - Reconstructed pictures on video display units in buildings
- E-voting system with audio activated (as it would be for visually impaired voters) produced interference with sound from a nearby transistor radio
  - Testers believed changes in the sound due to the interference could be used to determine how voter was vioting

Key Points

- Composing secure policies does not always produce a secure policy
  - The policies must be restrictive
- Noninterference policies prevent HIGH inputs from affecting LOW outputs
  - Prevents “writes down” in broadest sense
- Nondeducibility policies prevent the inference of HIGH inputs from LOW outputs
  - Prevents “reads up” in broadest sense
- Side channel attacks exploit deducability