CSCI 455: Principles of Computer Security

Set 07:

Operating Systems. Concurrency Issues

Concurrency and Race Conditions

- Concurrency essential to modern computing
  - Multiple separate execution flows are able to run simultaneously (or appear to)
    - Two mechanisms: threads and processes
- Uncontrolled concurrency leads to non-determinism
  - Program has different results with same inputs
- Race condition
  - An unanticipated execution ordering of concurrent flows resulting in undesired behavior
  - A software defect and source of vulnerabilities
- ATM example
  - Two partners withdrawing money from a joint account

Race Conditions: Three Necessary Conditions

- Concurrency property
  - Must have at least two concurrently control flows
    - In ATM example, the two partners
- Shared object property
  - A shared race object must be accessed by both concurrent flows
    - In ATM example, shared object is the account
- Change state property
  - At least one control flow must alter state of race object
    - In ATM example, withdrawals change state
- Attacker can exploit race condition to coordinate actions of several people using ATMs simultaneously

Eliminating Race Conditions

- Caused by unregulated access to shared resources due to process scheduling
- Must identify critical sections in each program where it must execute alone (mutual exclusion)
- Synchronization primitives implement mutual exclusion
  - Locks, counting and binary (mutex) semaphores, pipes and named pipes, monitors and condition variables, rendezvous
- Critical sections need to be atomic
  - Special entry and exit to make mutual exclusion

Deadlock

- Deadlock occurs whenever a set of tasks are blocked waiting one another to finish
  - Leads to a denial of service vulnerability
- 4 necessary conditions for deadlocks to occur
  - Circular wait (cycle)
  - Mutual exclusion
  - No preemption
  - Hold & wait
- Can be exploited!