Interest in code injection / hacking

Black/White hat competition
1. Intro
   – OS vs. Kernel?
     • main component of most computer operating systems; bridge between applications and actual data processing done at hardware level
     • managing the system's resources (the communication between hardware and software components)
     • lowest-level abstraction layer for the resources (especially processors and I/O devices) that application software must control to perform its function
   – Monolithic OS (Micro vs. Monolithic Kernel)
   – Hypervisor
     • also called virtual machine manager (VMM)
     • one of many hardware virtualization techniques that allow multiple operating systems to run concurrently on a host
2
• Kernel mode vs. user mode
• Non-present in page table
• Guest OS
• Fingerprint

3: TLB

4: Nonce
  – **Nonce** may refer to:
  – Nonce, time being: the present occasion; "for the nonce"
  – Nonce word, a word used to meet a need that is not expected to recur
  – Cryptographic nonce, a number or bit string used only once, in security engineering
  – The Nonce, American rap duo
  – Nonce (slang), a sex offender
  – Nonce orders, an architectural term

• Golden List

5: SPEC Benchmarks
Cache Miss Read Policy

- **direct-mapped cache:**
  - the valid bit is checked
    - **if not valid**, the block is read in, possibly forwarding the requested word;
    - **if valid**, the current block is eliminated (with consideration for the dirty bit) then the new block is loaded.
Cache Miss Read Policy

• set-associative cache:
  – if an empty line exists load block there
  – If no lines are empty, select a line to eliminate from the cache by one of two methods:
    • Least Recently Used (LRU)
    • Random Replacement
Cache Miss Write Policy

• There are two methods:

• *write allocate* -- the block is loaded into the cache and updated (commonly associated with write-back caches).

• *write-no allocate* -- the data is written directly to main memory and commonly associated with write-through caches.
Thrashing

- Thrashing can occur in many situations and results from a system being so overloaded that it spends all its time moving resources around so that they can be used.

- In caches, thrashing results when the same few blocks are being moved in and out of a common cache location.
Performance Impact

• Cache Size
  – Hit rate vs. Hit Time

• Block Size
  – Hit rate vs. fewer cache lines & higher miss penalty

• Associatively
  – Larger E: reduce risk of thrashing
  – But slower hit time, more tag bits, higher miss penalty

• Write strategy
  – Write through: simpler, can use write buffer, read misses less expensive
  – Write back: fewer transfers (more important in lower part of hierarchy)