CS 458 — Module 3

1 Outline

- Allows different entities to access different resources in a shared manner.
- The OS needs to control this sharing and provide an interface for allowing this access.
- Identification and authentication are required for access control.
- Focus on memory protection for now.

2 Protection in General Purpose Operating Systems

2.1 History

- Worth looking at how OSes have evolved until now.
- They now allow multiple users to use the same hardware, and added sequential abilities to run multiple *executives*.
- The OS should make resources available to users if required and permitted to by some policy.
- OSes also protect users from each other; attacks, mistakes, malware, and resource overconsumption are things to protect against, even if it is a single-user OS.

2.2 Protection Basics

- Some protected objects:
  - Memory
  - Data
  - CPU
  - Programs
  - I/O devices
  - Networks
  - The OS itself
- We talk about memory protection first — a program should not be able to read the memory of another program unless permitted.
- This is important as a *lot* of things are stored in memory!
- One way to invoke security is separation — keep a user's objects separate from another's!
- We can do this:
  - Physically (easy but expensive and inefficient)
  - Temporally (execute different users' programs at different times)
  - Logically (users don't see other users)
  - Cryptographically (make users unable to see other user's data unencrypted)

- But sometimes, users do want to share resources.
- For example, library routines or files/DB records.
- OSes should allow for flexible sharing — but this creates new questions:
  - Which files to share? What part?
  - Which users?
  - Which users can share objects further?
  - What uses are permitted? R/W/E perms? Maybe only specific bits of information (ie: only aggregate info, not individual entries of a DB).
  - How long?
- Often, for memory protection, the OS can exploit hardware support (virtual memory from CS 350) for a cheap solution.
- Memory addresses used are virtual and the MMU will manage this and translate to physical addresses.
- the OS maintains the mapping tables the MMU will use and deals with raised exceptions from the MMU.
- What kinds of hardware support is used for memory protection?
- The simplest is a fence register — exception if the memory access below addresses is in the fence register.
  - This protects the OS from user programs.
  - This only works for a single-user OS.
- Another technique is base/bound register pairs.
  - This throws an exception if memory access is below/above an address in the base/bound register.
  - Each user program has different values.
  - When a ctx switch occurs, the OS is responsible for moving the current base/bond registers to the currently executing user program.
  - We can extend this to have 2 base-bound pairs, one for code, one for data.
  - This approach is more flexible.
- The tagged architecture takes this to the extreme; each memory word has one or more extra bits that identify access rights to the word.
  - This is very flexible, but it incurs a large overhead and is not portable at all.
- Segmentation is a common approach (remember CS 350?).
  - Each program has multiple address spaces, or segments.
  - We have different segments for code, data, stack, though this can be split up even further.
  - The virtual address contains of two parts — the segment name, and the offset within the segment.
  - The OS will keeps mappings from the segment name to its base physical address in the segment table; one is made for each process.
  - The OS can relocate/resize segments and share them between processes.
  - The segment table can also keep protection attributes.
  - However, because of the resizing/relocation capabilities, this means every access requires a bounds check.

- In paging, we divide the virtual and physical address spaces into pages and frames respectively. Frame size is equal to page size.
  - The virtual address consists of the page number and offset within page.
  - The number of bits for the offset is $\log_2(\text{page size})$.
  - The OS keeps the mapping from page number to its base physical address in the page table.
  - Similarly to segmentation, protection attributes are kept in the page table.
  - Typically used; advantages are:
    - Each address reference is checked for protection by hardware
    - Prevent users from accessing unpermitted pages
    - Less used pages can be moved to disk
    - Users can share access to a page
  - Disadvantages:
    - Internal fragmentation still is a problem
    - Assigning different levels of protection of different classes of data items is not feasible
- The x86 architecture supports both segmentation and paging, though paging is more common.
- Memory protection bits can indicate no access, R/W access, or RO access.
- Processors now also typically include a no execute bit, or NX bit. This forbids execution of instructions stored in a page, which would make the stack/heap non-executable.
- Note that this does not stop buffer overflow attacks; see module 2 (see return oriented programming, where we don't inject new code but change existing code to do what we want).