Reference Monitors

Daniel Bosk

Department of Information and Communication Systems,
Mid Sweden University, SE-851 70 Sundsvall.

24th April 2017

- We now have authentication and authorization.
- But how do we enforce these access controls?
- This is where reference monitors come in.

Definition (Trusted Computing Base)

- The totality of protection mechanisms within a system which is responsible for enforcing a security policy.
- Consists of one or more components which together enforces the policy.
- Its ability to enforce a policy depends on proper configuration of its security mechanisms and those mechanisms themselves.

Definition (Reference Monitor)

Is an abstract concept referring to an abstract machine which mediates all subjects' accesses to objects.

Definition (Security Kernel)

- Constitutes hardware, firmware, software of a Trusted Computing Base (TCB) which implement the reference monitor concept.
- It must mediate all accesses, be protected from modification and be verifiable as correct.

Definition (Reference Monitor)

Is an abstract concept referring to an abstract machine which mediates all subjects' accesses to objects.

Definition (Security Kernel)

- Constitutes hardware, firmware, software of a Trusted Computing Base (TCB) which implement the reference monitor concept.
- It must mediate all accesses, be protected from modification and be verifiable as correct.

- The RM could be implemented in hardware using the microprocessor.
- It could be implemented in the OS kernel, e.g. access control in UNIX-like systems or Windows.
- It could be implemented in the services layer, e.g. database systems or Java Virtual Machine.
- Finally, it could be implemented in the application layer, i.e. security checks in the application code.

- One of the tasks of the OS is to prevent unauthorized access to different resources.
- What if the attacker could modify the OS?
- Hence we need protection for the OS, we need to maintain its integrity.

- Now we have the problem that a user must be able to use the OS.
- But the user shouldn't be able to misuse the OS.
- To help us achieve this we have
  - Modes of Operation, and
  - Controlled Invocation (Restricted Privilege).
- These can be applied on any layer, be it OS or application.

- We must be able to distinguish between what the OS executes for itself and what it executes on behalf of the user.
- A mode bit is used to indicate which mode a system is currently in.
- Usually we use only two modes, *user mode* and *kernel mode*.
- This way we can limit the possibility of execution.

- One problem we have now is to allow a user to invoke the privileged operations in the operating system.
- Clearly just flipping the mode bit wouldn't work, that way the user can do anything.
- So, we want to be able to flip the mode bit under certain circumstances only – and also flip it back before returning to the user.
- This is called *controlled invocation*.

- Placing mechanisms at the core will allow us higher level of assurance.
- Security mechanisms can be bypassed from the layer below.
- A more complex system gives less assurance.
- Mechanisms at the core can decrease overheads which decrease performance.

- Registers, such as
  - program counter,
  - stack pointer,
  - status register (state information).
- ALU which executes instructions.

- RAM
- ROM
- EPROM (erasable, programmable)
- WROM (write once)

- Volatile memory – fades, not vanishes.
- Non-volatile.

- Divide memory into logical units, good for security but more difficult.
- Divide memory into pages of equal length, efficient but more difficult for access control.

- Uses the interrupt vector to see at what address to start execution, where the interrupt handler is located.
- Can be pointed to some other code?

- Use a base and a limit register to limit the address space.

- The Motorola 68000 supported function codes for all addresses.
- This system included separation of user data, user code, kernel data, kernel code.