Lecture 12 - MAC Security
October 4, 2007
URL: http://www.cse.psu.edu/~tjaeger/cse543-f07/
Mandatory Access Control

• Is about *administration*

• Policy is defined and fixed for the system
  • Users cannot modify policy
  • More importantly, users’ processes cannot modify policy

• So, what should the policy be?
Security Goals

- Secrecy
  - *Do not leak data to unauthorized subjects*

- Integrity
  - *Do not depend on input from lower integrity subjects*
    - Invocation, inputs, files, etc.
MAC Systems

• Major Effort: *Multics*
  
  • Multiprocessing system -- developed many OS concepts
    
    • Including security
  
  • Begun in 1965
    
    • Development continued into the mid-70s
  
  • Used until 2000

• Initial partners: MIT, Bell Labs, GE/Honeywell

• Subsequent proprietary system, *SCOMP*, became the basis for secure operating systems design
Multics Goals

• Secrecy
  • Multilevel security

• Integrity
  • Rings of protection

• Reference Monitoring
  • Mediate segment access, ring crossing

• Resulting system is considered a high point in secure system design
Multilevel Security

• A multi-level security system tags all object and subject with security tags classifying them in terms of sensitivity/access level.
  – We formulate an access control policy based on these levels
  – We can also add other dimensions, called categories which horizontally partition the rights space (in a way similar to that as was done by roles)
Evaluating Policy

Access is allowed if

subject clearance level $\geq$ object sensitivity level and
object categories $\supseteq$ subject categories (read down)

Q: What would write-up be?

Hence,
Protection Rings

• Successively less-privileged “domains”

• Example: Multics (64 rings in theory, 8 in practice)

• Modern CPUs support 4 rings
  – Use 2 mainly: Kernel and user

• Intel x86 rings
  – Ring 0 has kernel
  – Ring 3 has application code
What Are Protection Rings?

- Coarse-grained, Hardware Protection Mechanism
- Boundary between Levels of Authority
  - Most privileged -- ring 0
  - Monotonically less privileged above
- Fundamental Purpose
  - Protect system integrity
    - Protect kernel from services
    - Protect services from applications
    - So on...
Intel Protection Ring Rules

• Each Memory Segment has a privilege level (ring number)

• The CPU has a Current Protection Level (CPL)
  – Level of the segment where instructions are being read

• Program can read/write in segments of lower level than CPL
  – kernel can read/write user space
  – user cannot read/write kernel
    • why not?
Protection Ring Rules

• Program cannot call code of higher privilege directly

  – Gate is a special memory address where lower-privilege code can call higher
    • Enables OS to control where applications call it (system calls)
Multics Interpretation

• Kernel resides in ring 0
• Process runs in a ring \( r \)
  – Access based on current ring
• Process accesses data (segment)
  – Each data segment has an *access bracket*: \( (a_1, a_2) \)
    • \( a_1 \leq a_2 \)
  – Describes read and write access to segment
    • \( r \) is the current ring
    • \( r \leq a_1 \): access permitted
    • \( a_1 < r \leq a_2 \): \( r \) and \( x \) permitted; \( w \) denied
    • \( a_2 < r \): all access denied
Multics Interpretation (con’t)

• Also different procedure segments
  – with *call brackets*: (c1, c2)
    • c1 <= c2
  – and access brackets (a1, a2)
  – Rights to execute code in a new procedure segment
    • r < a1: access permitted with ring-crossing fault
    • a1 <= r <= a2 = c1: access permitted and no fault
    • a2 < r <= c2: access permitted through a valid gate
    • c2 < r: access denied

• What’s it mean?
  – case 1: ring-crossing fault changes procedure’s ring
    • increases from r to a1
  – case 2: keep same ring number
  – case 3: gate checks args, decreases ring number

• Target code segment defines the new ring
Examples

• Process in ring 3 accesses data segment
  – access bracket: (2, 4)
  – What operations can be performed?

• Process in ring 5 accesses same data segment
  – What operations can be performed?

• Process in ring 5 accesses procedure segment
  – access bracket (2, 4)
  – call bracket (4, 6)
  – Can call be made?
  – How do we determine the new ring?
  – Can new procedure segment access the data segment above?
Multics Segments

- **Named segments** are protected by access control lists and MLS protections
  - Hierarchically arranged
  - Precursor to hierarchical file systems

- **Memory segment** access is controlled by hardware monitor
  - Multics hardware retrieves *segment descriptor word*
    - Like a file descriptor
  - Based on rights in the SDW determines whether can access segment

- **Master mode** (like root) can override protections

- Access a directory or SDW on each instruction!
Multics Vulnerability Analysis

• Detailed security analysis covering
  – Hardware
  – Software
  – Procedural features (administration)

• Good news
  – Design for security
  – System language prevents buffer overflows
    • Defined buffer sizes
  – Hardware features prevent buffer overflows
    • Addressing off segment is an error
    • Stack grows up
  – System is much smaller than current UNIX systems

• Vulnerability analysis found flaws that were fixed
  – Multics attained a B2 evaluation (MAC system)
Vulnerabilities Found

• Not mentioned in this paper

• Hardware
  – Indirect addressing -- incomplete mediation
    • Check direct, but not indirect address
  – Mistaken modification introduced the error

• Software
  – Ring protection (done in software)
    • Argument validation was flawed
    • Certain type of pointer was handled incorrectly
  – Master mode transfer
    • For performance, run master mode program (signaler) in user ring
    • Development assumed trusted input to signaler -- bad combo

• Procedural
  – Trap door insertion goes undetected
Proprietary product from Honeywell (owners of Multics)

*Security kernel: minimize TCB*

- **Custom Hardware**
  - Scomp, 4 rings
  - Complete mediation of memory access by bus mediation
  - Even by devices -- consider DMA

- **Operating System**
  - Scomp Trusted Operating Program (STOP)
  - Essential services only: build memory descriptors, schedule, ...

- **Application Programming Interface**
  - Scomp Kernel Interface Package (SKIP)
  - Minimal, basic kernel utilities (filesystem, processes, concurrency)

- **Designed to be general purpose**
  - But used for very limited operations
  - Guards: Ensure communication contains no secrets
Dime-a-Dozen

• Everyone started building secure operating environments

• Some from scratch
  – GEMSOS (security kernel)
  – PSOS (design only)
  – Adept-50 (High water mark)
  – KSOS (emulate UNIX interface)

• Many based on the OS’s of the day
  – KVM/370 (VM/370)
  – UCLA Secure UNIX (UNIX)
  – DEC OS (VAX/VMS)

• None particularly took hold
  – GEMSOS is still in business (Aesec)