Topics in Systems and Program Security

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University

September 8, 2008
Where are systems “secure”?

- **Paraphrase**
  - If people just used Multics, we would be secure
  - UNIX and Windows are insecure

- **What is the basis for these statements?**
Secure OS Requirements

- Mandatory Protection System
- Implemented by a Reference Monitor
Evaluation Criteria

• **Mediation**: Does interface mediate correctly?
• **Mediation**: On all resources?
• **Mediation**: Verifably?
• **Tamperproof**: Is reference monitor protected?
• **Tamperproof**: Is system TCB protected?
• **Verifiable**: Is TCB code base correct?
• **Verifiable**: Does the protection system enforce the system’s security goals?
Multics

- 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 Security

- 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

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,

Bob: CONF., \{INTEL\})

Charlie: TS, \{CRYPTO, NUC, INTEL\})

Alice: (SEC., \{CRYPTO, NUC\})

DocA: (CONFIDENTIAL, \{INTEL\})

DocC: (UNCLASSIFIED, \{NUC\})

DocB: (SECRET, \{CRYPTO\})
MLS as MPS

- Why is MLS a **Mandatory Protection System**?

- **Protection State:**
  - Labels are fixed
  - Information flows defined and fixed

- **Labeling State:**
  - Subjects login at a label
  - Objects are labeled at creation (according to MLS rules)

- **Transition State:**
  - No transitions of secrecy (covert channels)
Integrity Brackets

- 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
Process Invocation

- 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)
Procedure Invocation Brackets

• 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
Why is MLS a Mandatory Protection System?

Protection State:
- Rings are fixed in a hierarchy
- Protection state can be modified by owner

Labeling State:
- Ring determined by code (administration)
- Objects can be written according to access bracket

Transition State:
- Thru call brackets (guarded by gates)
Multics Reference Monitor

- Mediation
  - Security-sensitive operations on segments
  - All objects are accessed via a named hierarchy of segments
    * Predates file system hierarchies

- Tamperproofing
  - Reference monitor is part of the kernel ring
  - Minimize dependency on software outside kernel

- Verifiability
  - Lots of code
  - MLS for secrecy and rings for integrity
Multics Reference Monitor

Figure 3.2: The Multics login process. The user's password is submitted to the Multics answering service which must check the password against the entries in the password segment. The Multics supervisor in the privileged protection ring 0 authorizes access to this segment and adds a SDW for it to the answering service's descriptor segment. The answering service cannot modify its own descriptor segment.
• Process uses SDW to access a segment
  ‣ Directory stores a mapping between segments and secrecy level
  ‣ Each segment has a ring bracket specification
    • Copied into SDW
  ‣ Each segment has an ACL
    • Authorized ops in RWE bots

Figure 3.3: Structure of the Multics segment descriptor word (SDW): in addition to the segment’s address and length, the SDW contains access indicators including ring brackets (i.e., R1, R2, R3), the process’s ACL for the segment (i.e., the rwe bits), and the number of gates for the segment.
Evaluation Criteria

• **Mediation**: Does interface mediate correctly?
  ‣ Mediates on segment access using segment name or SDW
  ‣ But some indirection via directory for MLS labels

• **Mediation**: On all resources?
  ‣ All objects are segments
  ‣ Servers may compose higher-level objects

• **Mediation**: Verifably?
  ‣ All operations use segments; ring-transitions are well-defined
  ‣ Some use complex formats
Evaluation Criteria

- **Tamperproof**: Is reference monitor protected?
  - In ring 0, with trusted code
  - “Master mode” code runs in ring 0
- **Tamperproof**: Is system TCB protected?
  - In lower rings (1-3), separated from untrusted code
  - Protection depends on gates, code integrity, ring bracket policy
- **Verifiable**: Is TCB code base correct?
  - Code cannot be verified (lots of it)
- **Verifiable**: Does the protection system enforce the system’s security goals?
  - MLS provides formal guarantees
  - Little covert channel analysis; integrity depends on all TCB
Vulnerability Analysis

• Even with the care taken, vulnerabilities were introduced
  ‣ Hardware: indirect addressing mode
  ‣ Software: master mode in unauthorized way

• Master mode
  ‣ Run code with ring 0 perms
  ‣ Originally: Only in ring 0
  ‣ Change: Handle a page fault without a transition
    • Must protect integrity of such an operation! Transition state…
UNIX and Windows

- They do not provide either a mandatory protection state or a reference monitor concept
  - Why not?
Evaluation Criteria

• **Mediation**: Does interface mediate correctly?
  ‣ Mediates on object references

• **Mediation**: On all resources?
  ‣ UNIX: just files; Windows: Active Directory objects
  ‣ Network objects; Many other system objects; Servers may compose higher-level objects

• **Mediation**: Verifiably?
  ‣ Inside the kernels
  ‣ Some use complex formats
Evaluation Criteria

- **Tamperproof**: Is reference monitor protected?
  - In supervisor, with trusted code
  - Can extend kernels arbitrarily; can modify kernel memory; can change p-state

- **Tamperproof**: Is system TCB protected?
  - Uses trusted subject (root, admin)
  - Protection depends on setuid policy, code integrity, permissions, libraries, …

- **Verifiable**: Is TCB code base correct?
  - Code cannot be verified (lots of it)

- **Verifiable**: Does the protection system enforce the system’s security goals?
  - No clue
Take Away

• Multics originated the development of a “secure operating system”
  ‣ Real attempts were made to achieve reference monitor guarantees and provide a mandatory protection system (e.g., MLS)

• However, it is not easy to satisfy reference monitor guarantees, even when you try
  ‣ Especially, if your system maintainers are not trying

• And if you are not trying to enforce RM guarantees
  ‣ You won’t have anything close (UNIX and Windows)