Advanced Systems Security: Security Kernels

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

February 11, 2010
Multics circa 1976

- Final research report
  - Slower than desired
  - Bigger than desired
  - Expensive ($7M)
  - Low market share
  - UNIX winning mindshare

- Next generation systems?
Two Directions

• Focus on Generality and Performance
  ‣ Limited security
  ‣ Focus: UNIX
  ‣ Put us in our current state

• Focus on “verifiable” security
  ‣ Security kernels
  ‣ Lots of systems
    • KSOS, PSOS, Secure LAN, Secure Ada
      Target, various guard systems
MITRE Project

- Started in 1974
  - OS in 20 subroutines
    - Less than 1000 lines of code
  - System to manage physical resources
- What are the advantages of such an approach?
Security Kernel

- **Goals**
  - **(1)** Implement a specific security policy
  - **(2)** Define a verifiable protection behavior of the system as a whole
  - **(3)** Must be shown to be faithful to the security model’s design

- **Recommend a special issue**
  - IEEE Computer, 16(7), July 1983
Verification

- Became the focus of the approach
  - Verify that the implementation is faithful to the model
  - Which supports a specific security policy
- What are the formal limits of verification?
- What is it going to take in this case?
Secure Communications Processor (Scomp)

Figure 6.1: The Scomp system architecture consists of hardware security mechanisms, the Scomp Trusted Operating System (STOP), and the Scomp Kernel Interface Package (SKIP). The Scomp trusted computing base consists of code in rings 0 to 2, so the SKIP libraries are not trusted.
• Like Multics

• Access is control via **segments**
  ‣ Memory segments and I/O segments
  ‣ Files are defined at a higher level

• Security Goals
  ‣ Secrecy: **MLS**
  ‣ Integrity: **Ring brackets**
Scomp

- Unlike Multics
- Mediation on Segments
  - Although all access control and rings are implemented in hardware
- Formal verification
  - Verify that a formal model enforces the MLS policy
  - Trusted software outside the kernel is verified using a procedural specification
- Separate kernel from system API functions
  - In different rings (e.g., for file access)
Figure 6.2: The Scomp security protection module (SPM) mediates all accesses to I/O controllers and memory by mediating the I/O bus. The SPM also translates virtual addresses to physical segment addresses for authorization.
Scomp Drivers

• I/O Device Drivers in Scomp can be run in user-space
• Why can’t we do that in a normal OS?
• How can we do that in Scomp?
Scomp OS

- Whole thing is called Scomp Trusted Operating Program (STOP)
  - Lives on in BEA Systems XTS-400

- Security Kernel in ring 0
  - Provides “memory management, process scheduling, interrupt management, auditing, and reference monitoring functions”
  - In 10K lines of Pascal
  - Ring transitions controlled by 38 gates (APIs)
Trusted Software

- Officially part of STOP
  - But runs outside ring 0
- Software trusted to with system security goals
  - Like process loader
- System policy management and use
  - Such as authentication services
- 23 such processes, consisting of 11K lines of C code
  - All interaction requires a trusted path
Scomp Kernel Interface

- Like a system call interface for user processes
  - Trusted operations on user-level objects (e.g., files, processes, and I/O)
  - Still trusted not to violate MLS requirements
- Is accessible via a SKIP library
  - But that library runs in user space (ring 3)
Scomp Applications

• May also by MLS-Trusted Applications

• Mail Guard
  ‣ Makes sure that secrets are not leaked in communications to less secret subjects
  ‣ Mail guard obtains labeled communications
    • Has ad hoc filters to prevent leakage

• Why is Scomp appropriate to support such an application?
Scomp Evaluation

• **Complete Mediation:** Correct?
  ‣ In hardware
  ‣ In Trusted programs? In Mail guards?

• **Complete Mediation:** Comprehensive?
  ‣ At segment level
  ‣ For files? For mail data?

• **Complete Mediation:** Verified?
  ‣ Hardware; Trusted programs? Mail guards?
Scomp Evaluation

- **Tamperproof**: Reference Monitor?
  - In hardware, in kernel, in guard

- **Tamperproof**: TCB?
  - TCB is well-defined in rings, and protected by gates

- **Verify**: Code?
  - Performed verification on implementation using semi-automated methods
  - Led to assurance criteria and approach

- **Verify**: Policy?
  - MLS is security goal; Integrity is more difficult
Scomp Challenges

• Why don’t we all use Scomp-based systems now?
GEMSOS

- Similar system goals to Scomp
- Built for the ‘new’ x86 processor

*Figure 6.3: GEMSOS consists of a security kernel, gate library, and a layer of trusted software that is dependent on the deployed system. GEMSOS uses a software-based ring mechanism to simulate 8 protection rings.*
• Also, is still around
  ‣ Aesec corporation
• Fine-grained kernel design
• Eventually, UNIX (POSIX) emulation
• File system is inside the security kernel
  ‣ Kernel and trusted software depends on data layout in files
Take Away

• Security kernel design approach was designed to address security shortcomings of Multics
  ‣ In particularly, size and complexity

• Security kernel design approach
  ‣ Documented in a book by Morrie Gasser
  ‣ Led to the assurance approach in the Orange Book

• When people speak of how to build “secure OSes” they probably mean these systems

• So, we aren’t we building systems this way?

• What ideas/approaches can we take into current systems?