A Retrospective on the VAX VMM Security Kernel

Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, Andrew H. Mason and Clifford E. Kahn

Yogesh Raju Sreenivasan & Todd Arnold
Outline

• Background
• Related Work
• Design/Methodology
  – Virtualizing VAX
  – Security
  – Layered Design
  – Engineering Issues
  – Human Interface
• Conclusion
Background

• **Discretionary Access Controls**
  – Access rights at the discretion of the owner.
  – ACL & Capability based systems
  – Covert Channels
    • Storage Channels
    • Timing Channels

• **Mandatory Access Controls**
  – System defines access policy.
  – Lattice Security models
Background

- NCSC Evaluation Criteria (4 Major Categories)
- D-Minimal Protection
- C-Discretionary Protection
  - C1-Discretionary Security Protection
  - C2-Controlled Access Protection
- B-Mandatory Protection
  - B1-Labeled Security Protection
  - B2-Structured Protection
  - B3-Security Domains
- A-Verified Protection
  - A1-Verified Design
  - Beyond A1

http://en.wikipedia.org/wiki/TCSEC
Background

• **Virtual Machine**
  – “Efficient, isolated duplicate of the real machine” – Popek & Goldberg
  – A software program that emulates a hardware system.

• **Virtual Machine Monitor (or Hypervisor)**
  – Control program that implements virtual machines.
  – Multiplexes multiple virtual resources onto a single physical resource.
  – Properties of a VMM
    • Efficient
    • Control of Resources
    • Equivalence or Transparency
Background

Virtual Machine based approach for Security

- Provides high degree of isolation between users.
- All services and applications can be provided without extensive modifications those Operating Systems.
- More Reliable

Source: http://www.intel.com
Related Work

- IBM’s KVM/370
- sHype – Security Architecture of Xen Hypervisor
- Microsoft Virtual Server 2005
- VMWare ESX Server
Introduction

• VAX Security Kernel is a virtual-machine monitor.
• It provides an interface of the VAX Architecture and supports VMS & ULTRIX-32 in virtual machines.
• 5 major goals for VAX VMM Security Kernel
  – Meet A1 security requirements.
  – Run on commercial hardware.
  – Provide software compatibility for applications.
  – Provide acceptable performance.
  – Commercial software product.
Methodology

• Virtualizing the VAX

a) Sensitive Instructions:
   “An architecture will support virtual machines if the set of sensitive instructions is a subset of privileged instructions. “
   – Goldberg
   – *Sensitive Instructions: those that read or modify privileged system state*
   – *Privileged Instructions: those that trap when executed from a non-privileged mode.*
   – VAX architecture has some unprivileged but sensitive instructions (MOVPSL, PROBEx, REI)
   – Extensions to VAX Architecture
   – VM bit is added to the PSL
Methodology

• Virtualizing the VAX

b) Ring Compression:
   – 4 protection rings.
   – *User, Supervisor, Executive & Kernel*
   - Real ring numbers are concealed using
     1) *VM Bit in PSL*
     2) VMPSL
     3) *Modify all instructions that reveal ring numbers*
   - Change Memory protection of VM Pages.
   - No boundary between virtual kernel and executive mode!!
   - In both VMS & ULTRIX-32 Executive modes not used properly
Methodology

- **Virtualizing the VAX**
  
  c) **I/O Emulation:**
  
  a) In VAX I/O devices are programmed by reading and writing CSR
  b) VAX Security Kernel I/O is a specialized kernel call mechanism optimized for performance.
  c) VM stores I/O parameters in its I/O space
  d) Real I/O takes place on MTPR

  d) **Self-Virtualization:**
  
  a) “Ability of the VMM to run on its own VM”
  b) VAX VMM Support self-virtualization
  c) Useful for debugging.
Methodology

- **Subjects**
  a) Users & Virtual Machines
  b) Trusted path between User and Server (trusted kernel process)
     a) VMs are untrusted subjects (can also be treated as objects).

- **Objects**
  a) Real devices, Volumes & Primary Memory
  b) Disk volumes
     a) Exchangeable volumes and Security Kernel volumes
  c) Security Kernel files
     a) System Databases & logs
     b) Virtual Disks
Methodology

• **Access Classes**
  
a) VAX Kernel supports both secrecy (*Bell LaPadula*) and integrity (*Biba*) models.

b) Each kernel subject & kernel object has an access class (a secrecy class and an integrity class)

c) Read Access: $S_{AC} \text{ dom } O_{AC}$

d) Write Access: $O_{AC} \text{ dom } S_{AC}$

• **Privileges**
  
a) Analogous to Roles in RBAC

b) Restrict Access beyond DAC & MAC

c) User Privileges & Virtual-Machine privileges
Layered Design

- Levels of abstraction
- Based heavily on Multic’s
- Layers add specific functionality
Layered Design (cont.)

- Hardware-interrupt
- Low-level scheduler
- I/O Service
- Virtual Machine Physical Space Layer
- Virtual Machine Virtual Space Manager
Virtual Machines (VM)

Level 2 Virtual Processes (VP2)
- Dedicated VP2 - Server Processes
- Bindable VP2

Level 1 Virtual Processes (VP1)
- Dedicated VP1 - Device Drivers
- Bindable VP1
- Addressable VP1
Layered Design (cont.)

• Audit trail
• Files
• Volumes
• Virtual terminals
• Virtual Printers
Layered Design (cont.)

- Kernel Interface
- Virtual Vax
- Secure Server
- Virtual machine OS
- Users
Software Engineering Issues

• Used multiple languages
  – Pascal (16.5%)
  – PL/I (60.0%)
  – MACRO (23.5%)

• Memory Strategies
  – Sections of memory separated by no-access locations
  – Unused memory set to all zeroes
Software Engineering Issues (cont.)

- Defensive coding
  - Each layer protects against higher levels
  - Lower levels cannot call on higher levels
Human Interface

• Required to meet needs of commercial users
• Two command sets
  – Secure Server Commands
  – SECURE Commands
    • User SECURE
    • VM SECURE
• Secure Utilities
  – Reclassification
• Design issues
Assurance

- Code Design
- Testing
  - Layered Tests
  - KCALL
  - DTM
- Formal Methods
- Covert Channel Analyses
Production-Quality Kernels

- Required tools
  - Quality compiler and debugger
- Robustness
Conclusion

• VAX Security Kernel is a working production level kernel
• Effectively deals with covert-channels
• Successfully demonstrates work required to build A1 level kernel
• Obtaining performance is difficult, but not sufficient
• Discipline required for A1 certification improves overall software quality and reliability