A Retrospective on the VAX VMM Security Kernel

Paul A. Karger et al.

Presented by Haishan Liu
VAX Security Kernel
—DEC A1-Secure VMM system

- Designed for VAX architecture
  - VAX processor has to be modified to enable virtualization
- Designed to pass A1 security evaluation
- A technical success but not economically viable
Why Virtual Machines?

• To maintain compatibility with existing software

• To reduce software development and maintenance costs
Virtual Machines

- System virtual machines (hardware virtual machines)
  - allowing the sharing of the underlying physical machine resources between different virtual machines, each running its own operating system.

- Virtual Machine Monitor (hypervisor)
  - The software layer providing the virtualization
    - Type 1 or native VM: on bare hardware
    - Type 2 or hosted VM: on top of an operating system
VAX VMM Security Kernel

Configuration
Virtual Machine—Popek and Goldberg
virtualization requirement

**Theorem:** a VMM may be constructed if the set of *sensitive instructions* for that computer is a subset of the set of *privileged instructions*.

- **Sensitive instructions**
  - Those that either reveal or modify the privileged state of the processor
- **Privileged instructions**
  - Those that trap if the processor is in user mode and do not trap if it is in system mode.
Virtualizing VAX – sensitive instructions

- VAX architecture is not virtualizable
  - Containing sensitive instructions, but unprivileged

- Extensions have to be made
  - VM bit added to the processor status longword (PSL)
    - PSL—consisting of a set of processor state variables associated with each process
  - Sensitive instructions changed to trap based on the setting of the VM bit
Virtualizing VAX – ring compression

- Addition of a VM processor-status longword register (VMPSL)
  - Containing the virtual ring number

- Modification of all instructions that could reveal the real ring number
  - Trap to the VMM kernel
  - Obtain ring number from VMPSL
Virtualizing VAX – I/O emulation

- Native VAX
  - I/O by reading and writing CSRs (control and status registers) located in I/O space of physical memory.

- Virtual VAX
  - I/O by specialized kernel call mechanism
    1. VM stores I/O-related parameters
    2. VM issues Move to Privileged Register (MPTR) instruction
    3. MPTR traps to VMM kernel that then performs I/O

- Paravirtualization for I/O
- Hardware virtualization for non-I/O
Subjects

- User
  - communicates over the trusted path with a process called a *Server*
    - The Server is the user's direct interface to the kernel
    - The Server’s power is determined by the combination of the user and terminal’s access class and privileges
    - Users create sessions with VMs by issuing Server commands
- VM
  - runs VMOS
Objects

- Real devices
- Primary memory
- Disk volumes
- VAX security kernel volumes
  - VAX security kernel files
    - To hold System log and authorization file
    - To create Virtual disk volume
Access Class

- Secrecy/Integrity level
  - Hierarchical classification

- Secrecy/Integrity category
  - Non-hierarchical categories represent sensitivity

Secrecy class
- Level
- Category

Integrity class
- Level
- Category
Access Class

Access Class

Secrecy class
- Level
- Category

Integrity class
- Level
- Category

• \( \text{Access\_Class}(A) = \text{Access\_Class}(B) \) (A is equals to B) iff
  - \( \text{Secrecy\_level}(A) = \text{Secrecy\_level}(B) \)
  - \( \text{Secrecy\_Category}(A) = \text{Secrecy\_Category}(B) \)
  - \( \text{Integrity\_level}(A) = \text{Integrity\_level}(B) \)
  - \( \text{Integrity\_Category}(A) = \text{Integrity\_Category}(B) \)

• \( \text{Access\_Class}(A) \geq \text{Access\_Class}(B) \) (A dominates B) iff
  - \( \text{Secrecy\_level}(A) \geq \text{Secrecy\_level}(B) \)
  - \( \text{Secrecy\_Category}(A) \geq \text{Secrecy\_Category}(B) \)
  - \( \text{Integrity\_level}(A) \geq \text{Integrity\_level}(B) \)
  - \( \text{Integrity\_Category}(A) \geq \text{Integrity\_Category}(B) \)
Access Class

Determination of whether subject A may reference object B
- Access class of A
- Access class of B
- Type of the reference

E.g., A may read B, only if A dominates B.
Privileges

- System managers, security managers, and operators gain their powers by having *privileges*

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Powers</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASSIFY_DEVICE</td>
<td>Assign access classes to I/O devices and privileges to terminals</td>
</tr>
<tr>
<td>CLASSIFY_SUBJECT</td>
<td>Assign access classes and privileges to subjects; name levels and categories</td>
</tr>
<tr>
<td>CLASSIFY_VOLUME</td>
<td>Register and assign access classes to volumes</td>
</tr>
<tr>
<td>DELETE_AUDIT</td>
<td>Delete audit data</td>
</tr>
<tr>
<td>DOWNGRADE_SECRECY</td>
<td>DOWNGRADE_secrecy of text after human inspection</td>
</tr>
<tr>
<td>DOWNGRADE_SECRECY_NOINSPECT</td>
<td>DOWNGRADE_secrecy of data without inspection</td>
</tr>
<tr>
<td>ENABLE_DEBUGGER</td>
<td>Enable untrusted kernel debugger</td>
</tr>
<tr>
<td>OPERATE</td>
<td>Mount volumes, change printer paper, boot and shutdown system</td>
</tr>
<tr>
<td>REGISTER</td>
<td>Register and change non-security attributes of devices, virtual machines, and users</td>
</tr>
<tr>
<td>SET_AUDIT</td>
<td>Control audit log and real-time alarms</td>
</tr>
<tr>
<td>SET_COVERT_CHANNEL_DEFENSE</td>
<td>Enable or disable covert channel defenses</td>
</tr>
<tr>
<td>SET_FILE</td>
<td>Create, delete, or copy kernel files</td>
</tr>
<tr>
<td>SET_PASSWORD</td>
<td>Change users’ passwords and password parameters</td>
</tr>
<tr>
<td>UPGRADE_INTEGRITY</td>
<td>Upgrade integrity of text after human inspection</td>
</tr>
<tr>
<td>UPGRADE_INTEGRITY_NOINSPECT</td>
<td>Upgrade integrity of data without inspection</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Powers</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERATE</td>
<td>Dismount volumes; activate and deactivate other virtual machines; set login limits</td>
</tr>
<tr>
<td>SET_ACL</td>
<td>Change any object’s ACL, if access class permits</td>
</tr>
</tbody>
</table>
Layered Design

- reducing complexity and providing precise and understandable specifications
- Each layer implements some abstraction in part by making calls on lower layers
- No upward references
- called out in the NCSC requirements for B3 and Al systems
Software Engineering Issues

- Programming language choice
  - PL/I
    - Good data-typing support
    - Good character string manipulation support
    - Prior coding experience
  - PASCAL
    - Good V2.0 PASCAL compiler
    - Better data-type checking than PL/I
- Coding strategies
  - Avoidance of global pools, pre-allocation of resources, etc.
Human Interfaces

- Dilemma
  - Kernel must be small and verifiable
  - Market demands sophisticated user interface features

- Solution: two separate command sets
  - The Secure Server commands
    - Implemented entirely in trusted code (kernel)
  - The SECURE commands
    - Implemented in VMOS
  - Reduce the amount of trusted code and gain the well-developed command interface of mature commercial OS
Human Interface

- **Secure Commands**
  - Such as CONNECT, DISCONNECT, RESUME, etc.
  - Controlling terminal connections to virtual machines
  - Invoked via a trusted path to the Secure Server after user press the SECURE ATTENTION Key

- **SECURE Commands**
  - VM SECURE commands
    - Executed in the context of issuing VM
  - User SECURE commands
    - Executed in the Secure Server
    - Requiring the user to press the SECURE ATTENTION Key
    - the Secure Server displays the action of the command and prompts the user to approve or reject the operation.
Human Interface

```
$ SECURE DELETE TLS:STATUS.RPT
Press SECURE ATTENTION to complete execution of this command.
User presses SECURE ATTENTION to establish a trusted path.
Delete VAX Security Kernel file TLS:STATUS.RPT
Confirmation [Yes or No]: Y
VMM: File deleted
Resuming...
```

- User SECURE commands
  - Executed in the Secure Server
  - Requiring the user to press the SECURE ATTENTION Key
  - the Secure Server displays the action of the command and prompts the user to approve or reject the operation.
Human Interface

$ SECURE DELETE TLS:STATUS.RPT
Press SECURE ATTENTION to complete execution of this command.

User presses SECURE ATTENTION to establish a trusted path.

Delete VAX Security Kernel file TLS:STATUS.RPT
Confirmation [Yes or No]: Y

VMM: File deleted
Resuming...

- Difficulty with command confirmation
  - Introduction a form of asynchronous communication
  - Causing more complexity than a parser
  - “A menu interface combined with a mechanism for creating and checking precompiled scripts would have been simpler”
Assurance

- Rigorous protocol for design and code changes
  - Each layer has an owner
  - Code review results are publicized for the entire group to check
- Confined development environment
  - Developing and testing untrusted code on VMs
  - Hardware isolated from outside access
- Thorough test plan
Assurance

- Formal Methods
  - Formal top-level specification (FTLS)
  - Descriptive top-level specification (DTLS)
  - Formal Development Methodology (FDM)
    - Ina Jo
      - Security policy model
        - consisting of criteria and constraints and top-level specification (TLS)
      - FTLS
    - interactive theorem prover (ITP)
      - TLS obeys the security policy
      - FTLS maps to the TLS
Assurance

- Covert Channel Analysis and Countermeasures
  - Covert Channel: A path of communication not designed to be used for communication
    - *Covert storage channel* uses attribute of shared resource
      - E.g., Trojan allocates 0MB of memory to transmit 00, 64MB to transmit 01, 128MB to transmit 10, 192MB to transmit 11.
    - Optimization of disk arm movement
      - Initialization: Trojan seeks to cylinder 150
      - To send 1: Trojan seeks to cylinder 140
      - To send 0: Trojan seeks to cylinder 160
      - To receive: Spy requests seeking to cylinder 139 and 161, and then observes which finishes first.
  - *Covert timing channel* uses temporal or ordering relationship among accesses to shared resource
Assurance

- Covert Channel Analysis and Countermeasures
  - Storage channel
    - Mostly eliminated by preallocating resources
    - Optimization of disk arm movements
      - Complete countermeasure is possible
  - Timing channel
    - Using *fuzzy time* to make clocks less accurate
    - Fuzzy time can significantly reduce the channel bandwidth
Cancelation of VAX Security Kernel

- A technical success but not economically viable
  - U.S. export control drives up the development cost

- Failed to provide support to critical features
  - Ethernet
  - Windowing system

- High-performance A1-secure system alone is not sufficient for the marketplace
Lessons Learned

- VMM approach works
  - Achieving high security and good performance with legacy software compatibility
  - Allowing VAX/VMS and Unix co-existence
- A1-security is hard
  - Some of the A1 requirements can directly conflict with performance and usability goals