Systems and Internet Infrastructure Security

Network and Security Research Center
Department of Computer Science and Engineering
Pennsylvania State University, University Park PA

Advanced Systems Security: Virtual Machine Systems

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

Where are we?

- OS Security from Reference Monitor perspective
  - Mediation
    - LSM
  - Tamperproof
    - Linux and SELinux
  - Simple enough to verify
    - Correct code
    - Correct policy

Basis for OS Security

- Isolation
  - A protection domain defines a boundary of isolation
- Based on
  - Rings
  - Address spaces
  - Access control policy
- Do these work for modern OSes?

Virtual Machine Systems

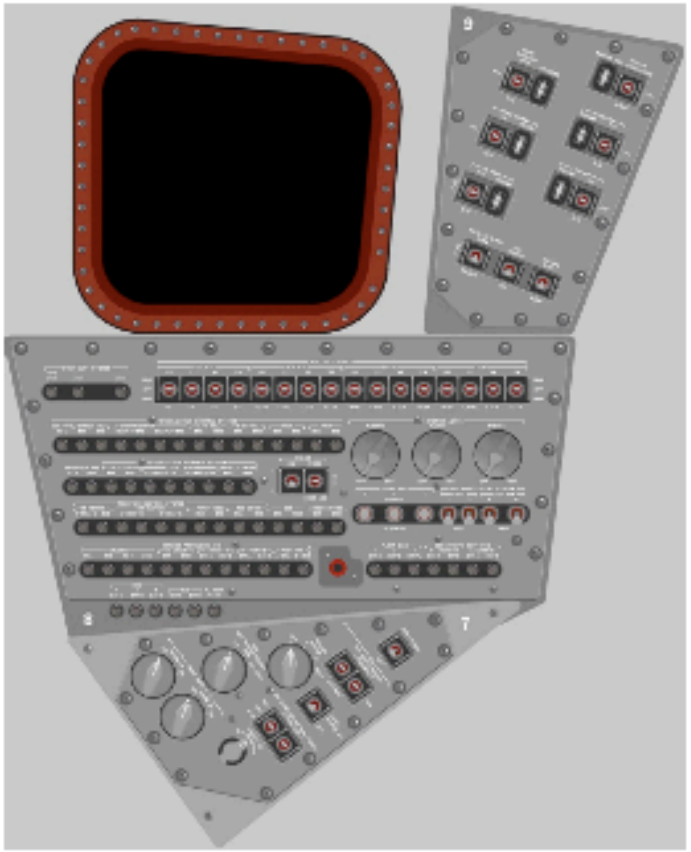
PENNSTATE 1855

- Protection domain is extended to operating systems on one physical platform
  - Invented for resource utilization
- But, also provide a potential security benefit due to default
  - ISOLATION
- How does VM isolation differ from OS isolation?

Virtual Machines

PENNSTATE

- Instead of using system software to enable sharing, use system software to enable ***isolation***
- Virtualization
  - "a technique for hiding the physical characteristics of computing resources from the way in which others systems, applications, and end users interact with those resources"
- Virtual Machines
  - Single physical resource can appear as multiple logical resources



Virtual Machine Types

PENNSTATE

- ***Type I***
  - Lowest layer of software is VMM
  - E.g., Xen, VAX VMM, etc.
- ***Type II***
  - Runs on a host operating system
  - E.g., VMWare, JVM, etc.

- Q: What are the trust model issues with Type II compared to Type I?

Hardware Virtualization

PENNSTATE 1855

- CPU virtualization
  - Instructions (still there)
  - Sensitive instructions must be privileged
- Memory virtualization
  - MMU (still there)
  - Nested/extended page tables
- I/O virtualization
  - IOMMU (new)
  - Chipset support for configuration and address translation

VM Systems and Ref Monitor

- How does a VM System improve ability to achieve reference monitor guarantees?

VM Systems and Ref Monitor

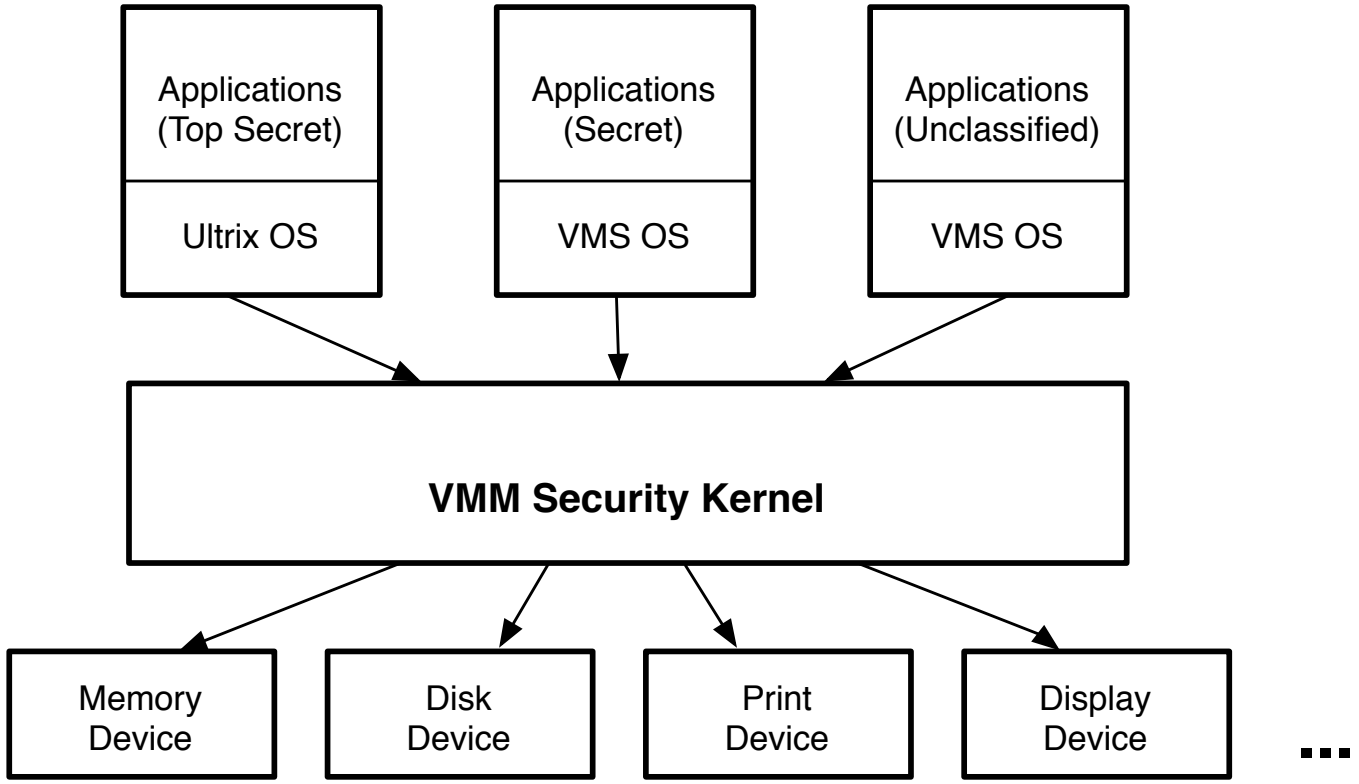
- How does a VM System improve ability to achieve reference monitor guarantees?
- Mediation
  - Mediation between VM interactions
- Tamperproof
  - Protection boundaries between VMs (OS)
- Simple Enough to Verify
  - Code that needs to be correct?
  - Policy

VAX VMM

PENNSTATE 1855

- A1-assured (formally assured) VMM system
- Carefully crafted VMM
- Mediation
  - VM (subject) and volume (object)
- Tamperproof
  - "Minimal" TCB – VMM only
- Simple enough to verify
  - Code assurance
  - Policy assurance: MLS policy, Biba policy, privileges

VAX VMM Architecture



VAX VMM Reference Monitor

PENNSTATE 1855

- Key design tasks
  - Virtualize processor
    - Make all sensitive instructions privileged
  - More rings
    - Need a new ring for the VMM
  - I/O emulation
  - Self-virtualizable
- What components constitute the VAX VMM reference monitor?

VAX VMM Policy

PENNSTATE 1855

- MLS
  - Control secrecy
- Biba
  - Control integrity
- Privileges
  - Exceptional accesses
  - Audited
  - There are more of these than meets the eye!
- How is the protection state modified?

VAX VMM Evaluation

PENNSTATE 1855

- **Mediation**: ensure all security-sensitive operations are mediated?
  - Virtualizing instructions, I/O emulation
  - VM-level operations? Privileges
- **Mediation**: mediate all resources?
  - VMM level
- **Mediation**: verify complete mediation?
  - A1-assured at VMM level

VAX VMM Evaluation

PENNSTATE 1855

- **Tamperproof**: protect VMM?
  - Similar to Multics (no gatekeepers, but some kind of filters); authentication in VMM; protection system ops in VMM
- **Tamperproof**: protect TCB?
  - All trusted code at ring 0; trusted path from VMs for admin;
- **Verification**: verify code?
  - A1-assured at VMM level
- **Verification**: verify policy?
  - MLS and Biba express goals and policy; Privileges are ad hoc

VAX VMM Challenges

PENNSTATE 1855

- Despite A1 assurance still several challenges in VAX VMM system
  - Device driver management; no network
  - Amount of assembler code
  - Covert channel countermeasures
  - Implications of 'privileges'
- Nonetheless, interesting mechanisms
  - Virtualization for security
  - Architecture of VMM system
  - Trusted path administration

Modern VM Systems

PENNSTATE 1855

- The development of a virtual machine monitor for x86 systems unleashed VMs on the masses
  - Why did this take so long?
- VMware, Xen, KVM, NetTop, …
  - Everyone is a virtual machine monitor now
- How do we implement a reference validation mechanism for these systems?
  - What granularity of control?

Isolation and Network

- Type I VM Systems assume that the VMM (and privileged VM) will isolate guest VMs
- Then, the problem is to control inter-VM communication
  - VMs talk to VMM (hypercalls, like system calls)
  - All other communication is via the network
- sHype adds reference monitor for controlling network access between VMs
- NetTop is built on VMware where only VMs of the same label may communicate via network

Control of VMM Resources

PENNSTATE 1855

- There are many virtual machine monitor resources that may be used to communicate
  - Memory, devices, IPC, VMs themselves, …
  - E.g., VMware permits VMCI – like IPC between VMs
- Xen Security Modules (XSM) adds reference validation on the Xen hypervisor's distribution of these resources
  - Less trust in privileged VMs, so finer-grained policy results
- Minimizing TCB versus simplicity

Xen as a Reference Monitor

PENNSTATE 1855

- Reference Monitor
  - XSM in Xen
  - Scope includes "dom0" VM
- Mediation
  - XSM to control VMM operations
  - SELinux in dom0; use network to communicate
- Tamperproof
  - Xen and Linux
- Verification (Xen)
  - Xen Code – 200K+ LOC – and Dom 0 Linux
  - Policy – SELinux style

Container Systems

PENNSTATE 1855

- A hybrid approach is developed in container systems
- Linux containers run multiple Linux systems (process hierarchies) on one Linux host operating system
  - Cgroups enables resource control without starting VMs
  - Also, each container gets its own namespaces for processes, mounts (filesystem), userids, and networks
    - Idea is to give each container an isolated view
- How do we configure access control for containers?

Container Systems

PENNSTATE 1855

- How do we configure access control for containers?
  - E.g., SELinux across and within containers…

Container Systems

PENNSTATE 1855

- How do we configure access control for containers?
  - Currently, the host system defines mandatory access control policies that govern every container
  - What are issues with that approach?

Dune

PENNSTATE 1855

- Goal: Safe access to hardware features from processes
- Normally, only the operating system can configure hardware features, such as page tables, ring protection, and TLBs
- However, applications may benefit from direct access to such hardware features
  - Modifying the kernel to provide such access in a sufficiently flexible way while maintaining security is a problem

Dune

- Approach: Dune uses virtualization hardware to provide a "process" rather than a "machine" abstraction
- Alternative: Instead of modifying the host kernel to achieve application-specific use of hardware features, an alternative is to deploy processes in a VM with a custom OS to do so
- However, launching a process in a VM can be complex because of sharing of OS abstractions, such as file descriptors between parent and child
  - Won't work if they are in different VMs

Dune – Process Abstraction

PENNSTATE 1855

- Process: Can enter "Dune mode" to access hardware features
  - Including privilege modes, virtual memory registers, page tables, and interrupt, exception, and system call vectors
  - Through use of virtualization hardware – Intel VT-x
    - VMX root and VMX non-root modes
      - VMX root – for VMM
      - VMM non-root – for virtualized operating systems, governed by VMM
  - Dune processes use VMCALL to invoke system calls – with help of library provided

Dune – System Architecture

- System: Dune mode is VMX non-root mode
  - Kernel is in VMX root mode like a VMM
  - Dune processes are in VMX non-root mode
  - Dune module intercepts VM exits, which are the only way to access the kernel – for syscalls and traps
- Other processes are unaffected

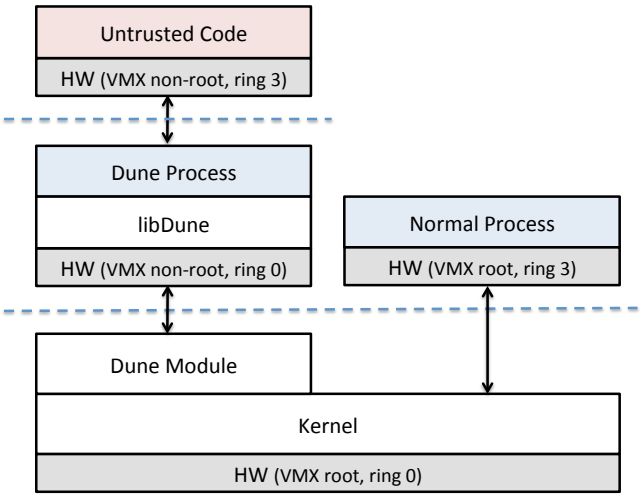


Figure 1: The Dune system architecture.

Dune – Memory Management

- Goal: manage page tables from user processes
  - But, just what programs want to manage – not all
  - Without allowing arbitrary access to memory
- Dune processes reference guest-virtual memory, so protected by extended page table – like process is a VM
  - Sync EPT to kernel PT

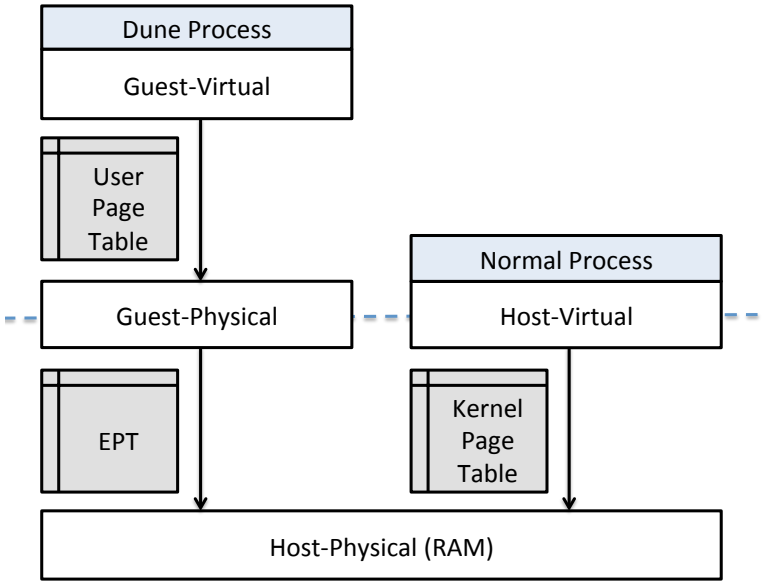


Figure 2: Virtual memory in Dune.

Container Security

PENNSTATE 1855

- Better or worse than VMs?
- Worse: Containers share the same OS
- Better: Containers only have one application
- Better: Containers can have limited attack surface by running it in a "jail"
- Worse?: Hypervisor can provide stronger isolation than an OS
  - However, Dune shows that such isolation is implemented by VT-x hardware, so same in OS and hypervisor

Conventional OS vs VM System

- Conventional OS
  - Broken easily and often
- VM system
  - Coarser control based on isolation
- If we trust the VM system and don't trust the OS, what can we do?

Deploy Critical Applications

PENNSTATE 1855

- Don't trust OS, but need its services
- Run programs directly on VMM
  - No services
- Run programs on a specialized, trusted system
  - Custom services must be written (yuk!)
- Reuse untrusted system services
  - Trusted system (custom, but potentially smaller) must enable secure use of such services

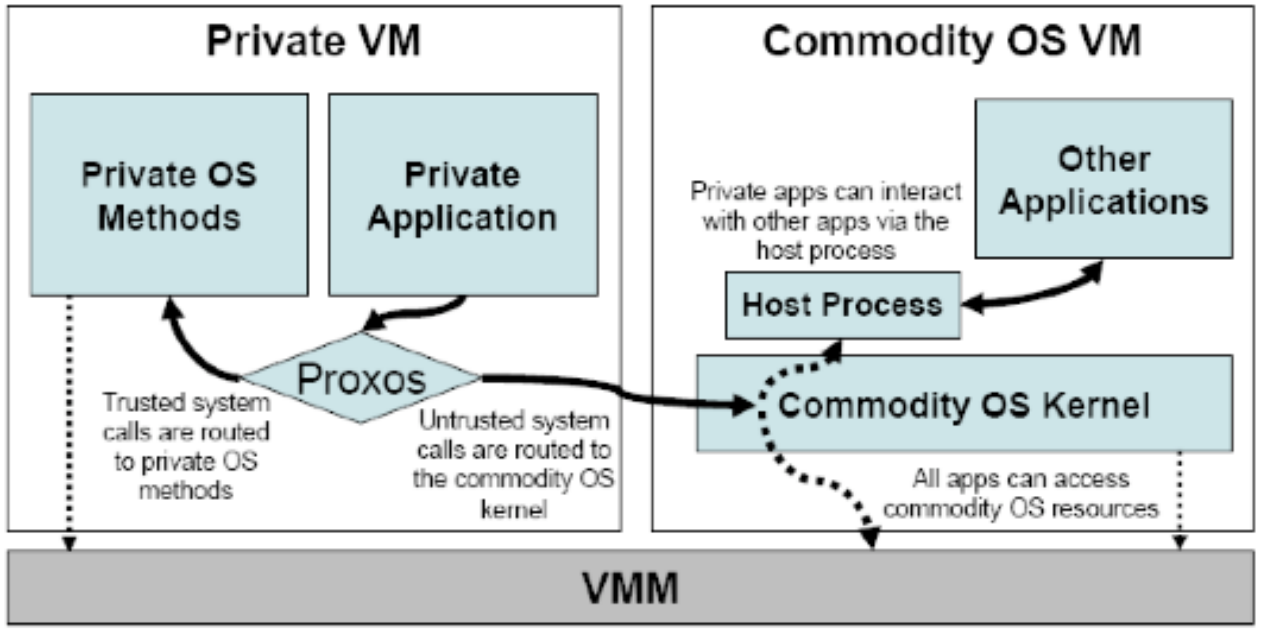
Splitting Interfaces

PENNSTATE 1855

Solution

- Separate application from other apps/kernel
  - Use separate VM for app with a Private OS separate from Commodity OS
- Provide interaction between apps/kernel in a secure way
  - Application developer decides what is sensitive and what is not
    - Separate sensitive part into VM on Private OS
    - Public part remains on Commodity OS
    - Interaction between apps also passes through kernel (eg. pipe(), mkfifo())
    - Sensitive part communicates through system calls with other apps
  - Use policy to decide if system calls are to be performed on commodity OS or private OS

Proxos Architecture



Proxos Guarantees

PENNSTATE 1855

- Assumption
  - VMM enforces separation
  - Application developer correctly specifies routing rules
- Guarantee
  - Confidentiality and integrity of sensitive private application data inspite of malicious commodity OS
    - VMM => No direct interference possible
    - Commodity OS can interfere with system calls routed to it, which are not security-sensitive
  - Availability not guaranteed

Proxos Routing Language

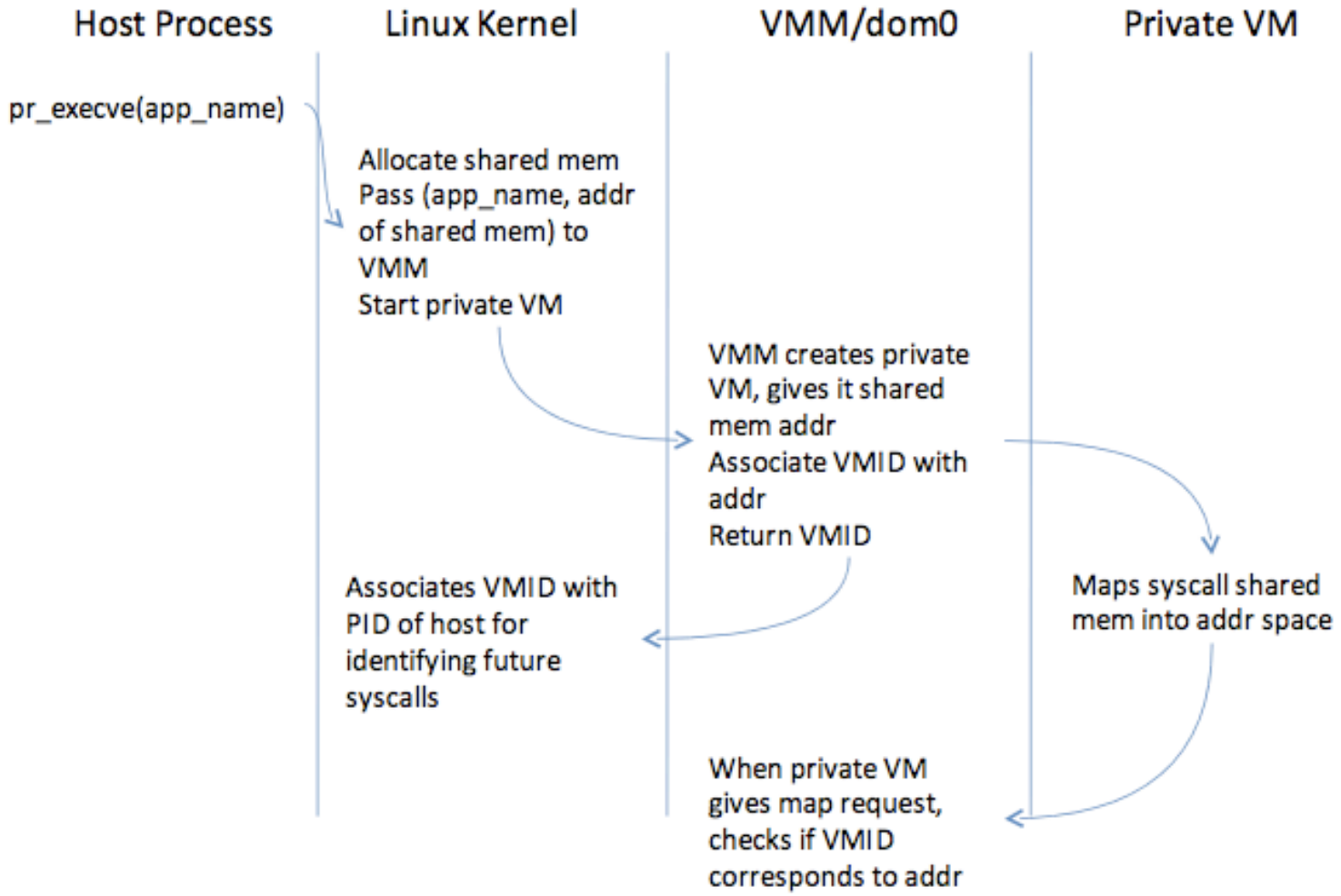
PENNSTATE 1855

- Needs to specify which system calls go where (arguments need be considered)
- Solution: Partition system calls by resources they access
  - Disk, Network, UI, Randomness, System Time, Memory
  - Randomness, System Time -> Always routed to VMM
  - Memory -> Always routed to private OS

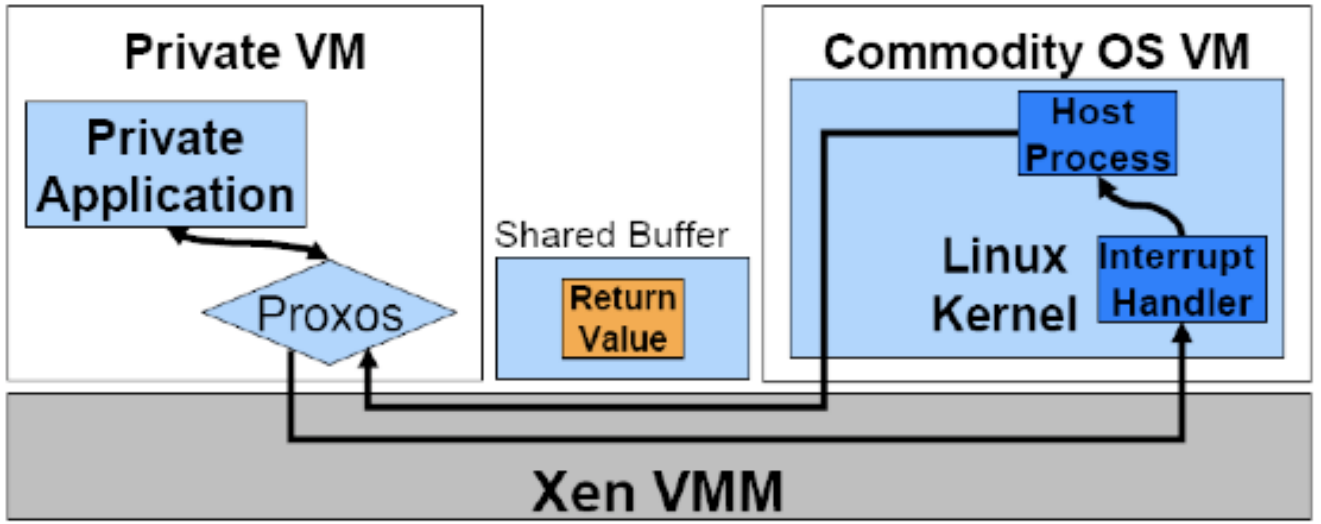
```
# Rules Section
# route accesses to /etc/secrets to private OS
DISK:("/etc/secrets", priv_fs)
# route accesses to UNIX domain socket bound
# to /tmp/socket and TCP socket bound to peer
# 192.100.0.4 port 1337 to private OS
NETWORK:("unix:/tmp/socket", priv_unix),
        ("tcp:192.100.0.4:1337", priv_tcp)
# route all accesses to stdin, stdout
# and stderr to private OS
UI: (*,priv_ui)

# Methods Section
# individual methods in the private OS
# that are bound to system calls
priv_fs = {
  .open = priv_open,
  .close = priv_close,
  .read = priv_read,
  .write = priv_write,
  .lseek = priv_lseek
}
```

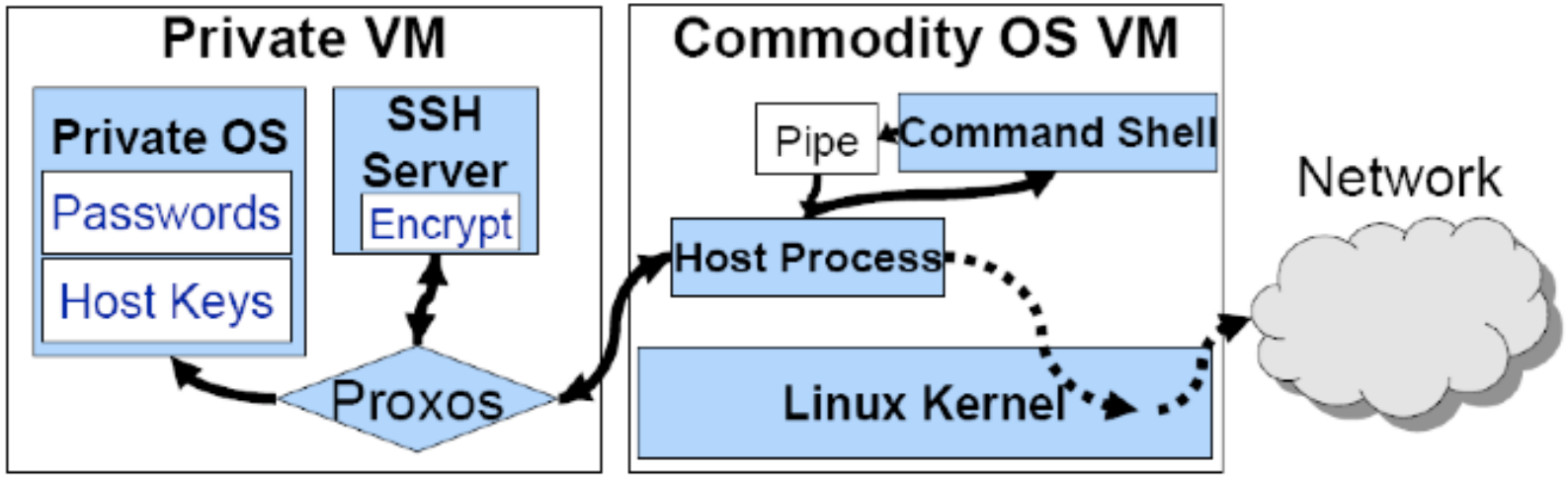
Proxos Implementation



Proxos Implementation



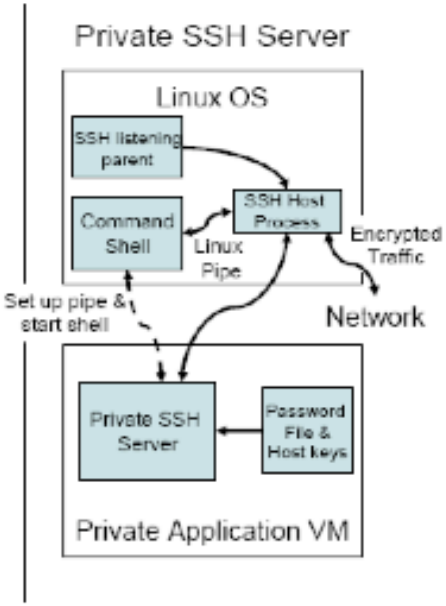
Proxos SSH Server



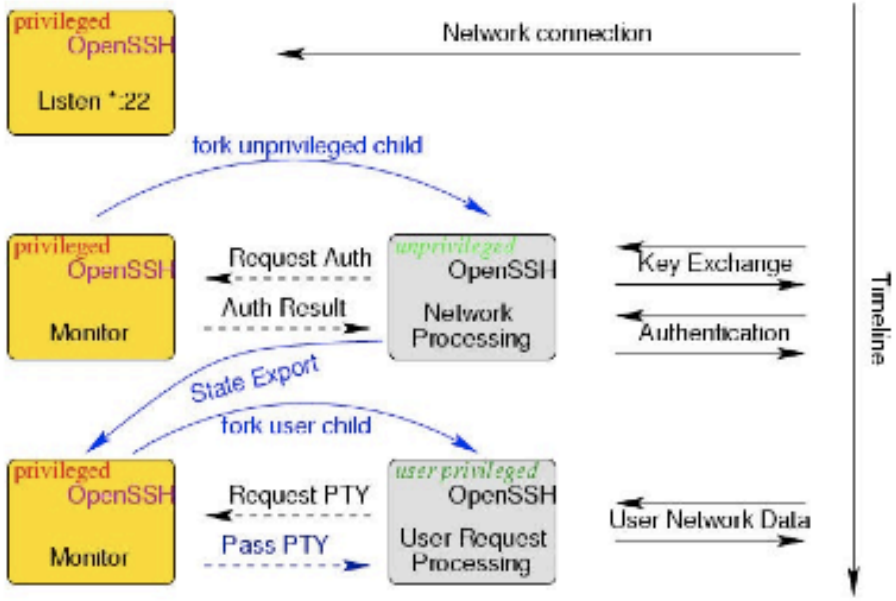
Compare to Privilege Separation

PENNSTATE 1855

Partitioning Interfaces to Resources



Partitioning Code (Provos et al)

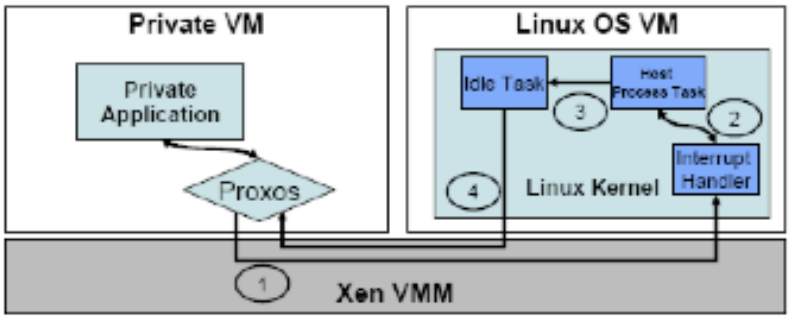


Implementation Effort

Application	Rules	LOC Modified	Total LOC	% Modification
Dillo	53	22	20,528	0.1%
SSH Server	35	108	27,000	0.4%
Apache & OpenSSL	28	667	135,916	0.5%
Glibc		218	1,775,440	0.01%

Performance

- System call forwarding overhead
  - Context Switch Cost: 14us.



Benchmark	Linux	Proxos	Overhead
NULL system call	0.37	12.88	12.51
fstat	0.57	14.28	13.71
stat	8.76	25.98	17.22
open & close	14.57	47.18	32.61
read	0.45	13.51	13.06
write	0.42	13.24	12.82

TrustShadow

PENNSTATE 1855

- The TrustShadow system employs the Proxos approach to deploy isolated applications that do not trust the Linux kernel
  - An application of Proxos to approximate SGX guarantees (next time)
- Isolated, unmodified applications are launched on the TrustShadow runtime system using the ARM TrustZone "Secure World"
- Runtime intercepts most system calls and forwards to them to the Linux kernel in the "Normal World"

Take Away

- VM Systems provide isolation
  - At OS granularity: some can be untrusted
- Moving towards container systems
  - Dune enables flexible use of hardware by "containers"
- Can we use VM isolation to prevent compromise of applications by malicious OS?
  - Proxos: use a "trusted" OS and redirect service requests
    - Applied in TrustShadow to isolate domains