Securing Your Cloud With the Xen Hypervisor

Russell Pavlicek
Xen Project Evangelist
Citrix Systems

Who is the Old, Fat Geek Up Front?

- Xen Project Evangelist (I have a big mouth...)
- Employed by Citrix, focused entirely on Xen project
- History with open source begins in 1995
- Former columnist for Infoworld, Processor magazines
- Former panelist on The Linux Show, speaker at over 50 Open Source conferences
- Over 150 pieces published, one book on open source, plus blogs

Presentation Goals

- Introduce the subject of security in the Cloud
- Introduce you to the Xen Project Security Tools
- Discuss some key Xen Project security features
- Get you started in the right direction toward securing your Xen Project Hypervisor installation

Presentation Outline

- A few thoughts on the problem of securing the Cloud
- Overview of the Xen Project architecture
- Brief introduction to the principles of security analysis
- Examine some of the attack surfaces and the Xen Project features we can use to mitigate them:
  - Driver Domains
  - PVgrub
  - Stub Domains
  - Paravirtualization (PV) versus Hardware Virtualization (HVM)
  - FLASK example policy

Introduction: Xen Project, The Cloud, and Security

Introduction: Xen Project and Security

- Xen Project produces an enterprise-grade Type 1 hypervisor
- Built for the cloud before it was called cloud
- A number of advanced security features
  - Driver domains, stub domains, FLASK, and more
  - Most of them are not (or cannot) be turned on by default
  - Although they can be simple to use, sometimes they appear complicated

The Cloud Security Conundrum

- Cloud Security: The 800lb Gorilla in the room
  - Nothing generates more fear in specific, and FUD in general
  - Probably the single greatest barrier to Cloud adoption
    - Immediately behind it is the inability to get out of the 20th century IT mindset
      - Must get past the “Change is Bad” concept of 1980
      - Cloud is about embracing change at a rapid pace
  - The good news: the “Gorilla” is actually a “Red Herring”
    - We don't need to fear it – we just need to solve it

Cloud Security: New Visibility to an Old Problem

- Security has always been an IT issue
- Putting a truly secure system in the open does not reduce its security, it just increases the frequency of attack
- Unfortunately, system security behind the firewall has not always been comprehensive
- Having solutions in an external cloud forces us to solve the security issues we should have already solved

---

News Flash: ***Security Through Obscurity is Dead***

Use Security by Design, Not by Wishful Thinking

- Security by wishful thinking no longer works
  - Merely hoping that your firewall holds off the marauding hordes is **NOT** good enough
  - Addressing security in one area while ignoring others is **NOT** good enough
  - Saying, “We never had a problem before” is **NOT** good enough
- Comprehensive security starts with ***design***
  - It needs to be planned carefully and thought through
  - It needs to be implemented at multiple levels
  - It needs components which are themselves securable

Xen Project: Security by Design

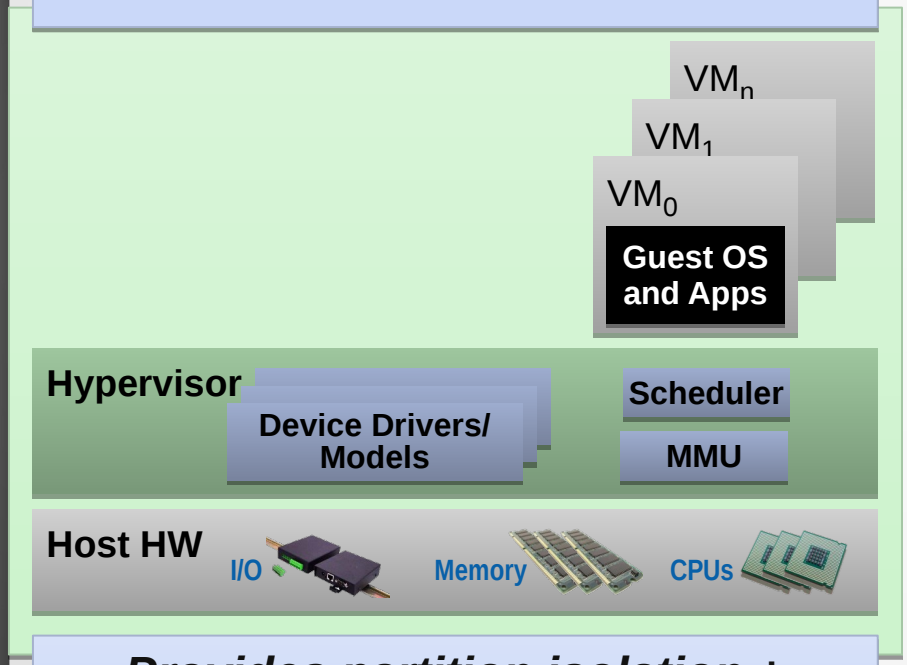
- Xen Project was designed for clouds before the term “cloud” was ever coined in the industry
  - Designers foresaw the day of “infrastructure for wide-area distributed computing” which we now call “the cloud”
  - *http://www.cl.cam.ac.uk/research//srg/netos/xeno/publications.html*
- Xen Project is designed to thwart attacks from many attack vectors, using different defensive techniques

Basic Architecture of the Xen Project Hypervisor

Hypervisor Architectures

Type 1: Bare metal Hypervisor

A pure Hypervisor that runs directly on the hardware and hosts Guest OS's.

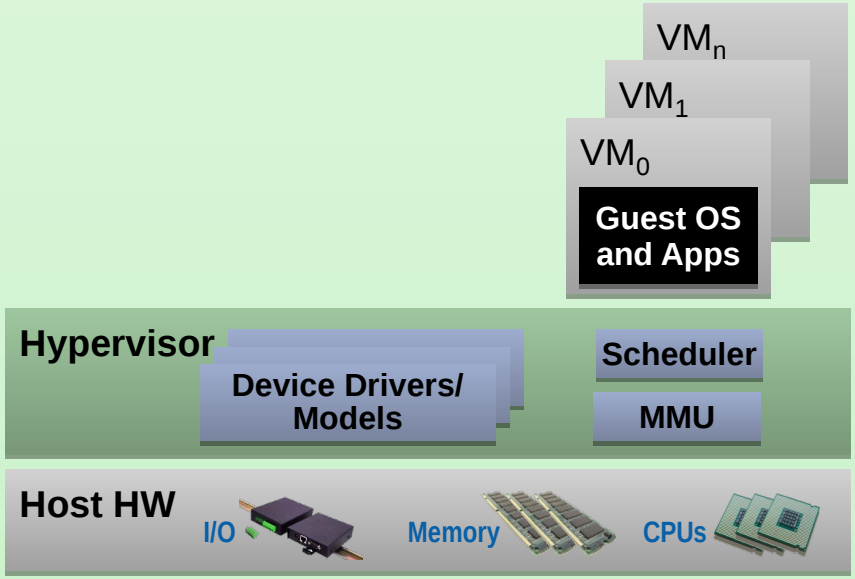


Provides partition isolation + reliability, higher security

Hypervisor Architectures

Type 1: Bare metal Hypervisor

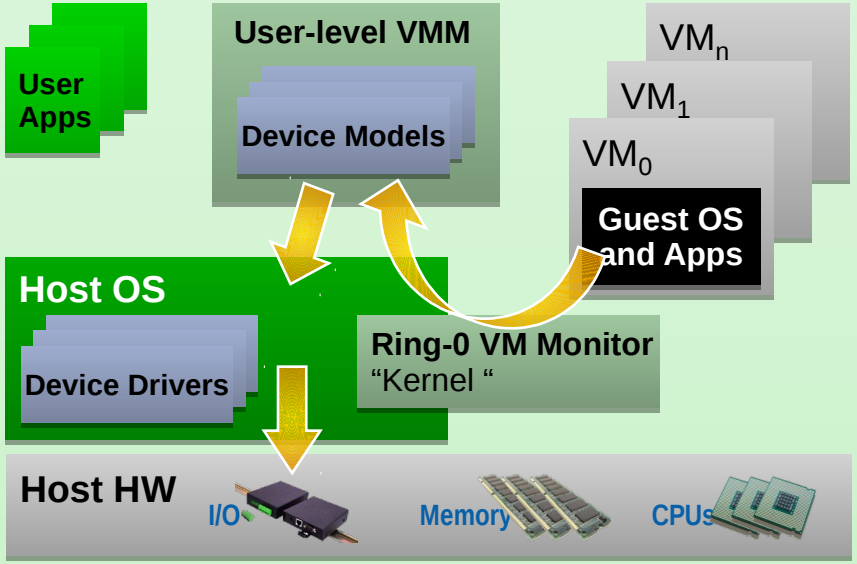
A pure Hypervisor that runs directly on the hardware and hosts Guest OS's.



Provides partition isolation + reliability, higher security

Type 2: OS 'Hosted'

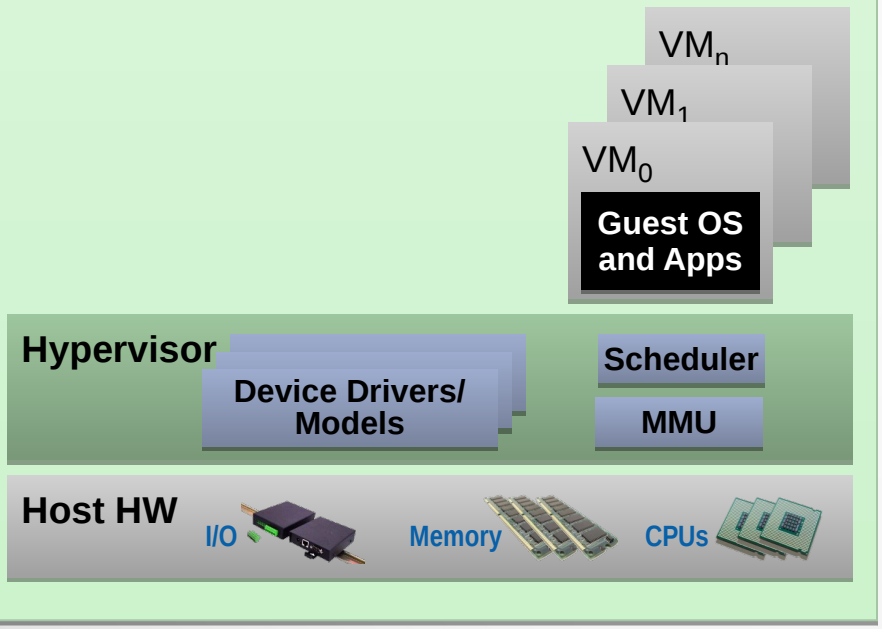
A Hypervisor that runs within a Host OS and hosts Guest OS's inside of it, using the host OS services to provide the virtual environment.



Low cost, no additional drivers Ease of use and installation

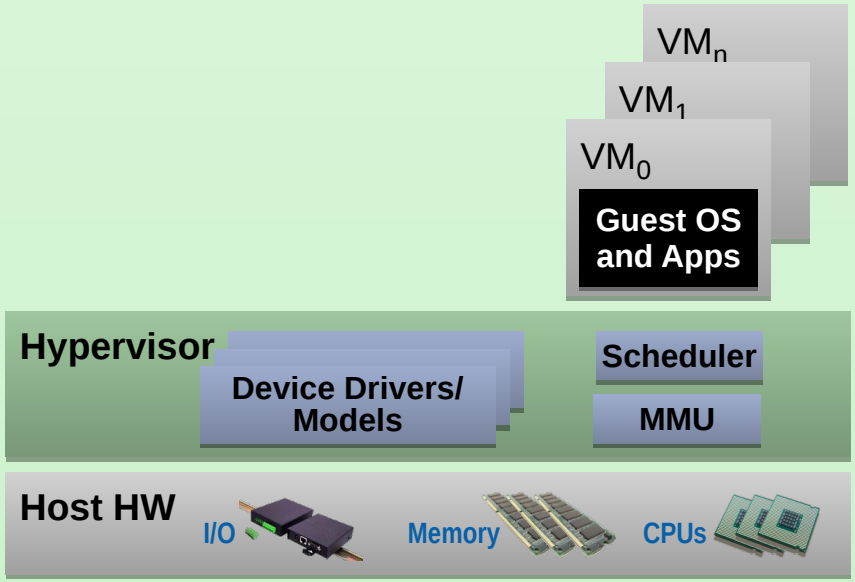
Xen Project: Type 1 with a Twist

Type 1: Bare metal Hypervisor

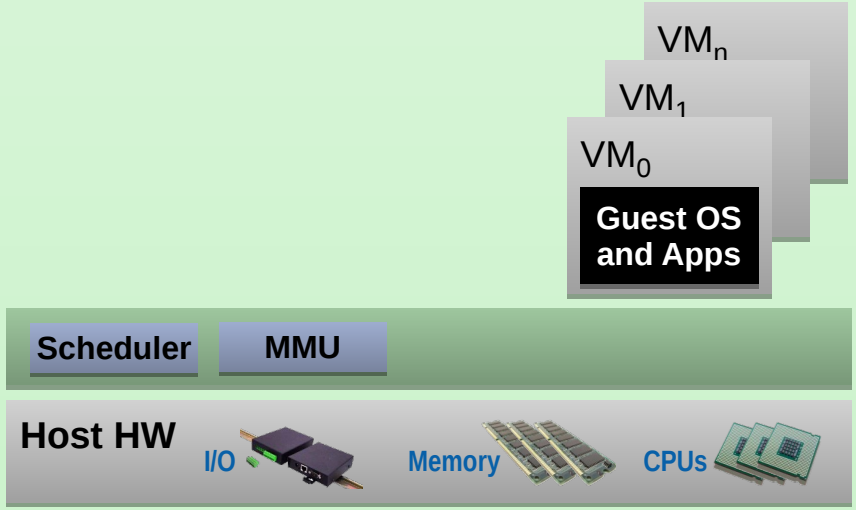


Xen Project: Type 1 with a Twist

Type 1: Bare metal Hypervisor

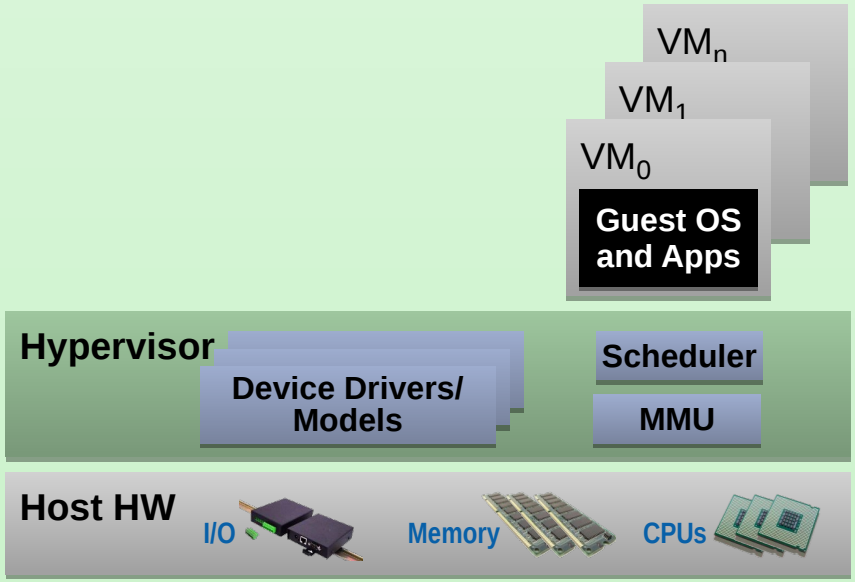


XEN Architecture

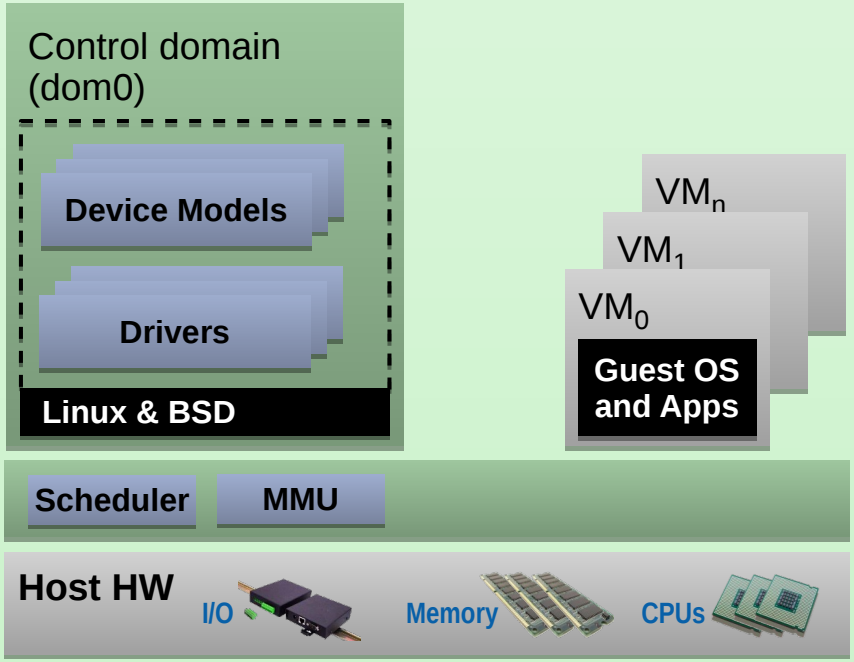


Xen Project: Type 1 with a Twist

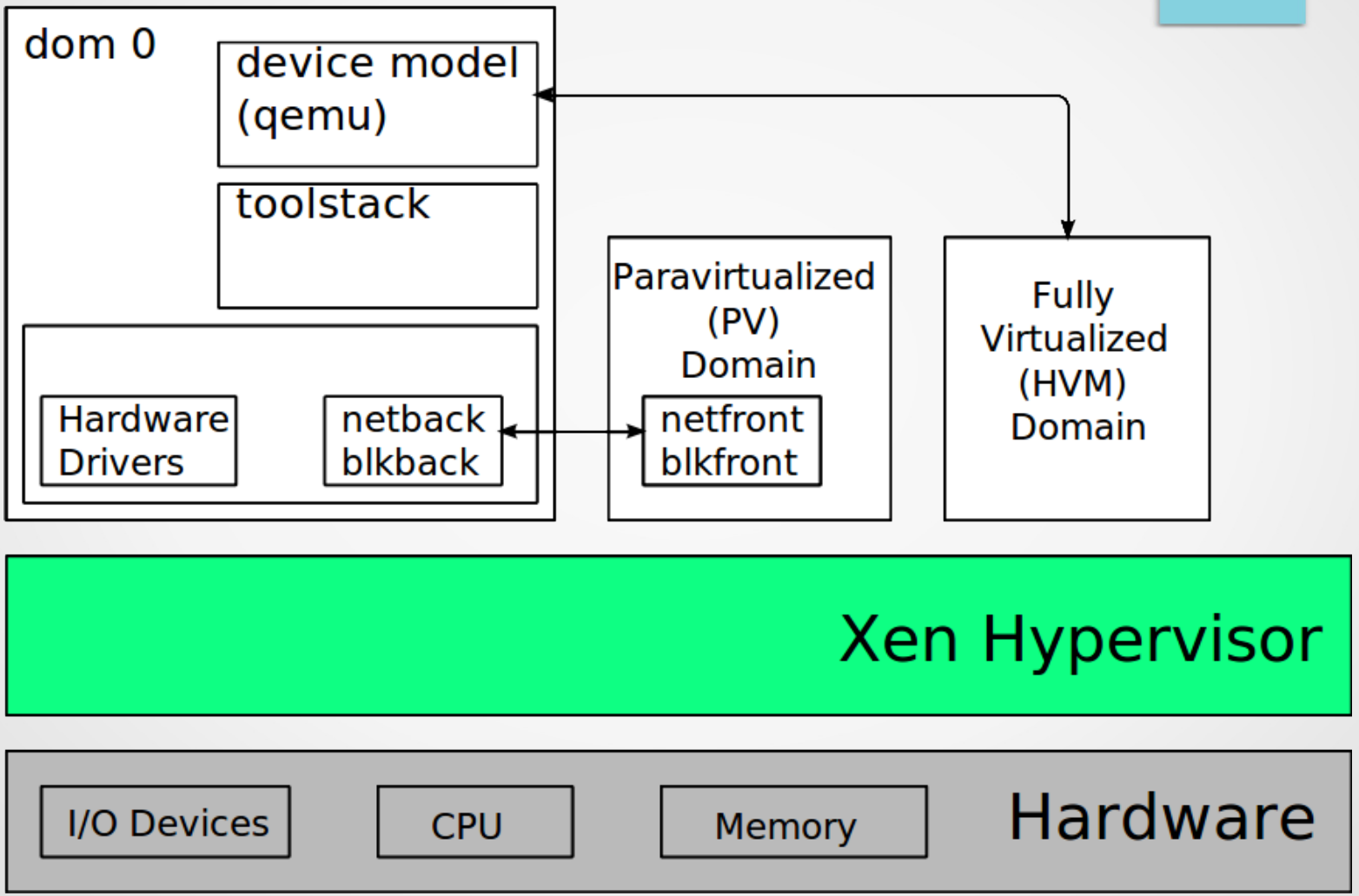
Type 1: Bare metal Hypervisor



XEN Architecture



Xen Project Architecture: Basic Parts



Security Thinking: An Approach

An Approach to Security Thinking

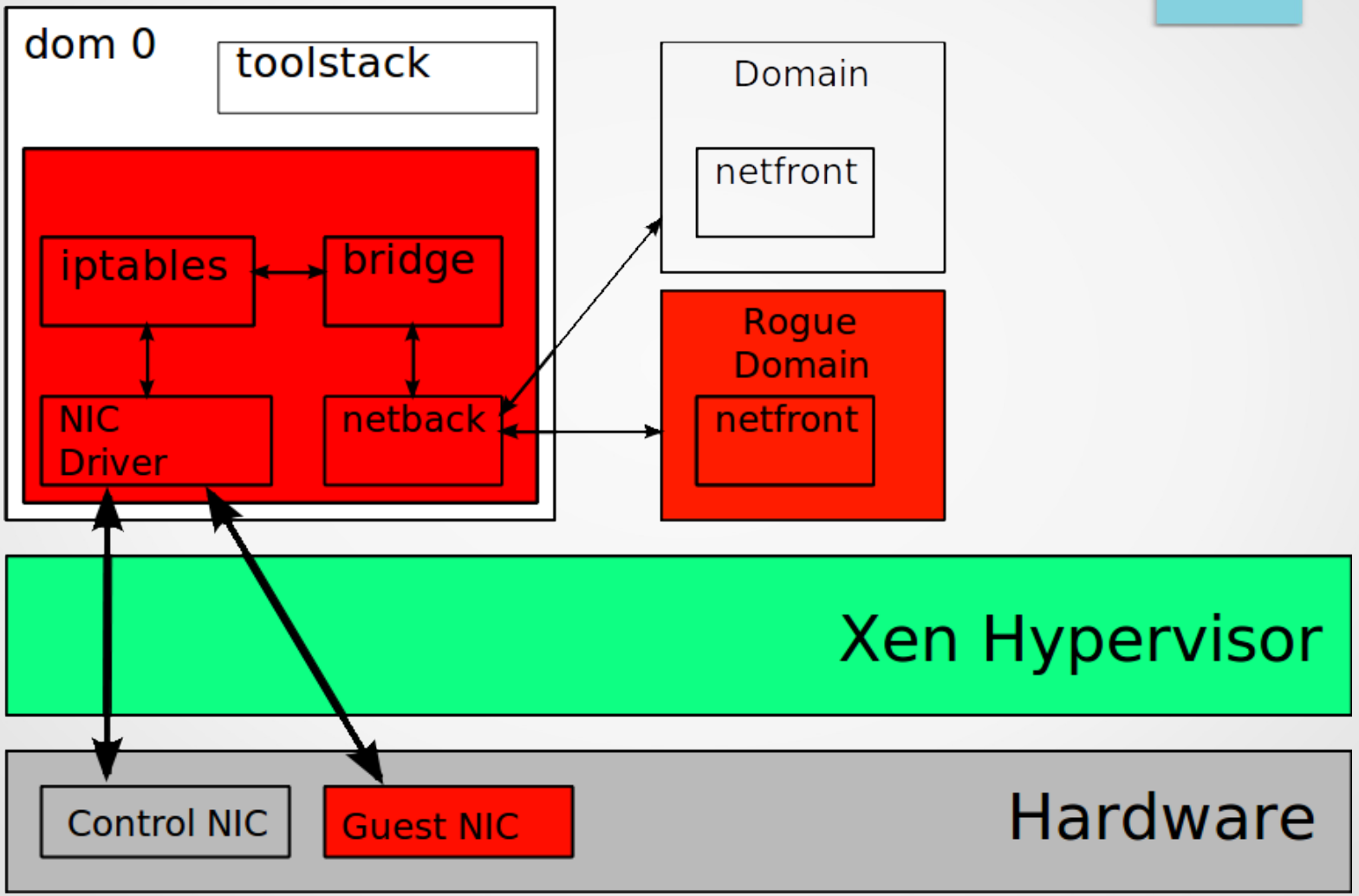
- Threat models:
  - Attacker can access network
  - Attacker controls one Guest VM
- Security considerations to evaluate:
  - How much code is accessible?
  - What is the interface like? (e.g., pointers vs scalars)
  - Defense-in-depth: how many rings of defense surround you?
- Then combine security tactics to secure the installation
  - There is no single "magic bullet"
  - Individual tactics reduce danger; combined tactics go farther

Example System For This Discussion

- Hardware setup
  - Two networks: one Control network, one Guest network
  - IOMMU with interrupt remapping (AMD or Intel VT-d v2) to allow for full hardware virtualization (HVM)
- Default configuration
  - Network drivers in the Control Domain (aka "Domain 0" or just "Dom0")
  - Paravirtualized (PV) guests using PyGrub (grub-like boot utility within context of Guest Domain)
  - Hardware Virtualized (HVM) guests using Qemu (as the device model) running in the Control Domain

Attacking the Network Interface

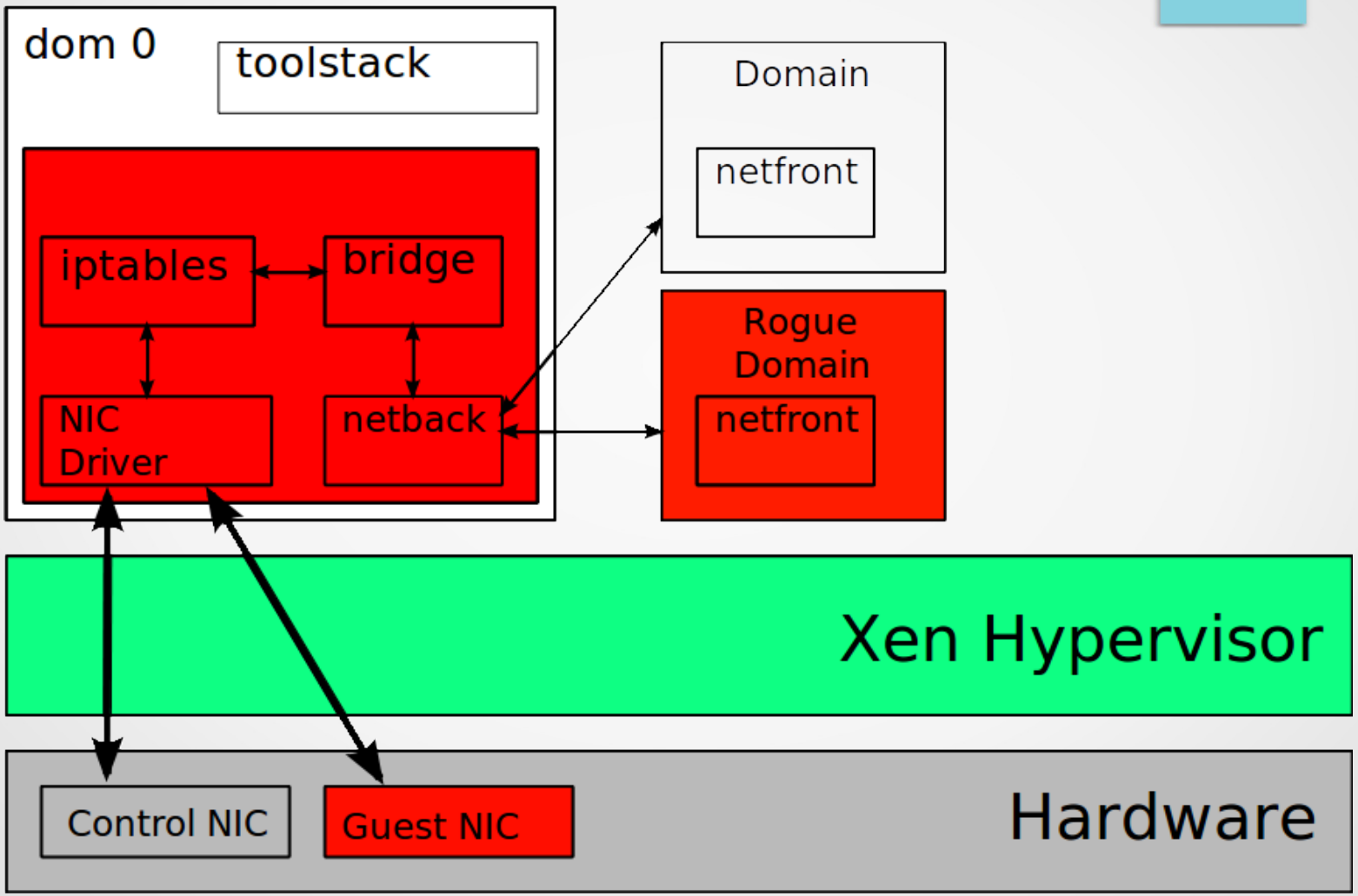
Attack Surface: Network Path



Attack Surface: Network Path

- Where might an exploit focus?
  - Bugs in hardware driver
  - Bugs in bridging / filtering
  - Bugs in netback (via the ring protocol)
    - Netback and Netfront are part of the Paravirtualization mode
- Note the exploits
  - The main exploits exist already, even in hardware
  - The netback surface is very small, but needs to be acknowledged
  - When these are attacked in hardware, you have deep trouble
  - You actually have better defense in the VM than in hardware

Result: Network Path Compromised



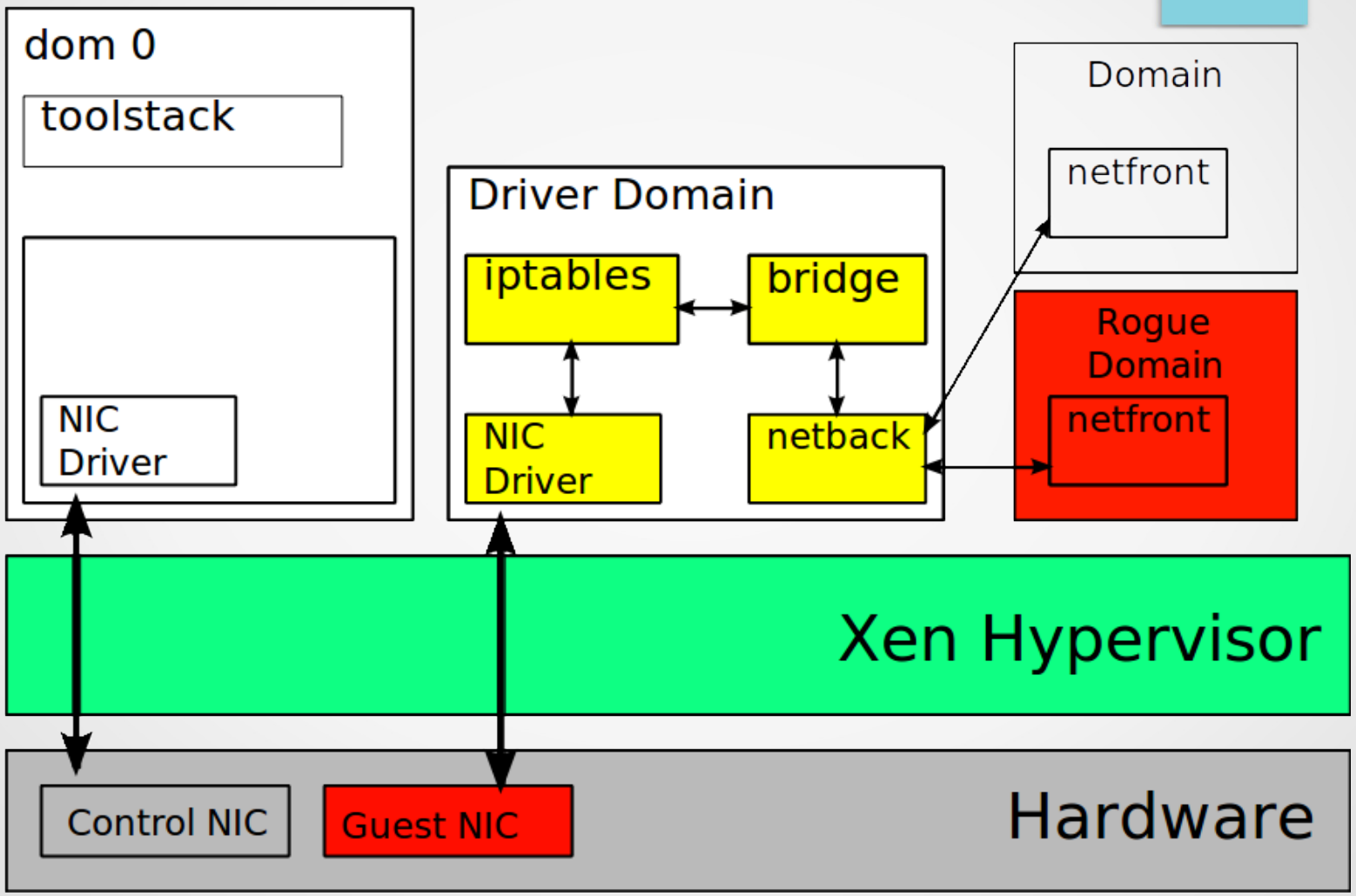
Result: Network Path Compromised

Vulnerability Analysis:

What could a successful exploit yield?

- Control of Domain 0 kernel
- **This could lead to the control of the whole system**

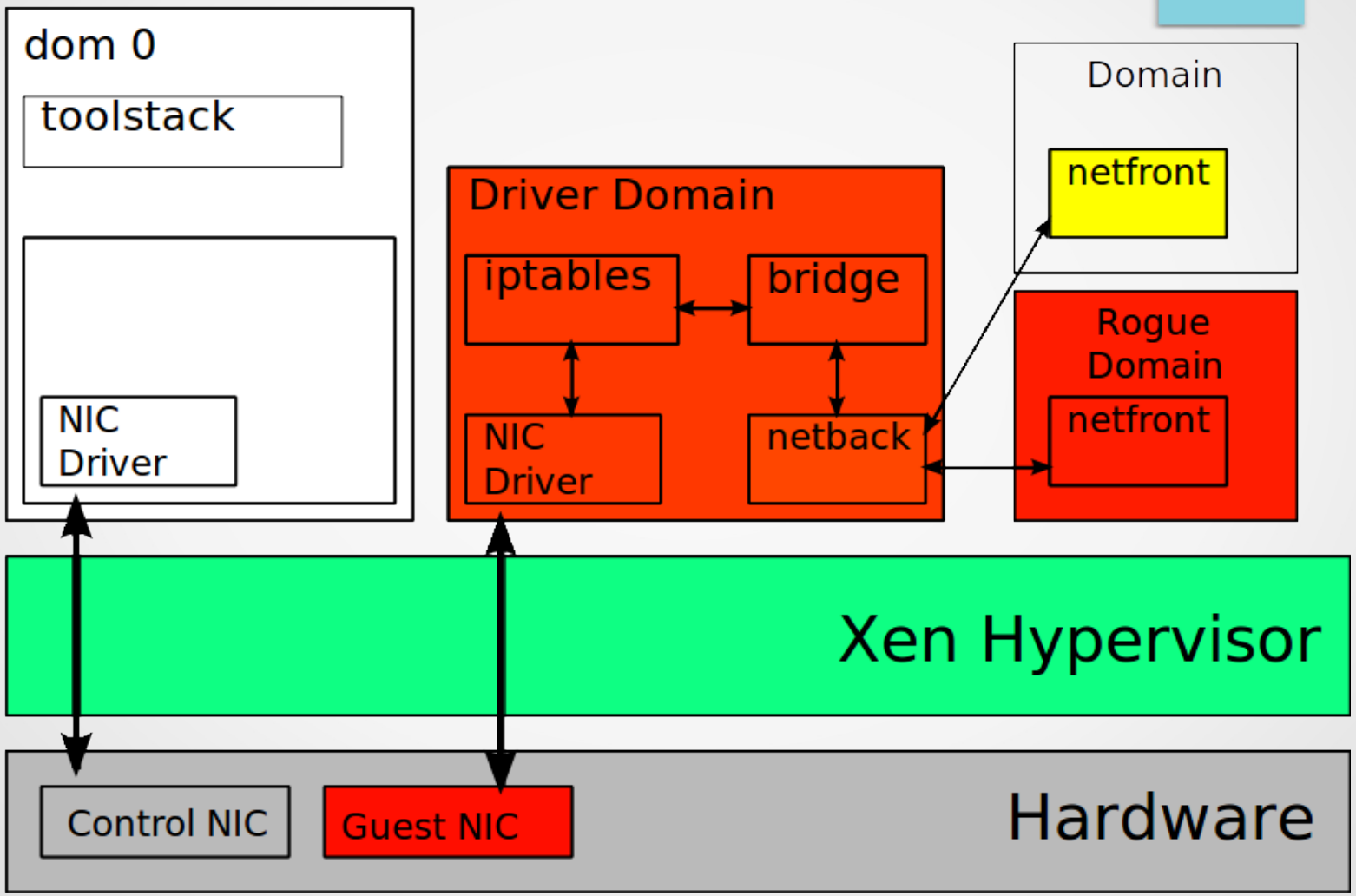
Security Feature: Driver Domains



Security Feature: Driver Domains

- What is a Driver Domain?
  - Unprivileged VM which drives hardware
  - It provides driver access to guest VMs
  - Very limited scope; not a full operating system
  - Does not have the access or capability of a full VM

Result: Driver Domain Compromised



Result: Driver Domain Compromised

- Now a successful exploit could yield:
  - Control of the Driver Domain
    (Paravirtualization hypercall interface)
    - But the Driver Domain is limited: no shell, no utilities
  - Control of that guest's network traffic
    - But in the cloud, most orchestrators detect network traffic failure
    - The problem is not allowed to stand very long
  - Control of the network interface card (NIC)
  - An opportunity to attack the netfront of other guest VMs
    - But to take advantage of this platform, you need to launch another attack
    - Compound attacks are complex, and they take time which you may not have

Basic How To: Driver Domains

- Create a VM with appropriate drivers
  - Use any distribution suitable as a Control Domain
- Install the Xen Project hotplug scripts
  - Just installing the Xen Project tools in the VM is usually good enough
- Give the VM access to the physical NIC with PCI passthrough
- Configure the network topology in the Driver Domain
  - Just like you would for the Control Domain

Basic How To: Driver Domains

- Configure the guest Virtual Network Interface (vif) to use the new domain ID
  - Add “backend=domnet” to vif declaration

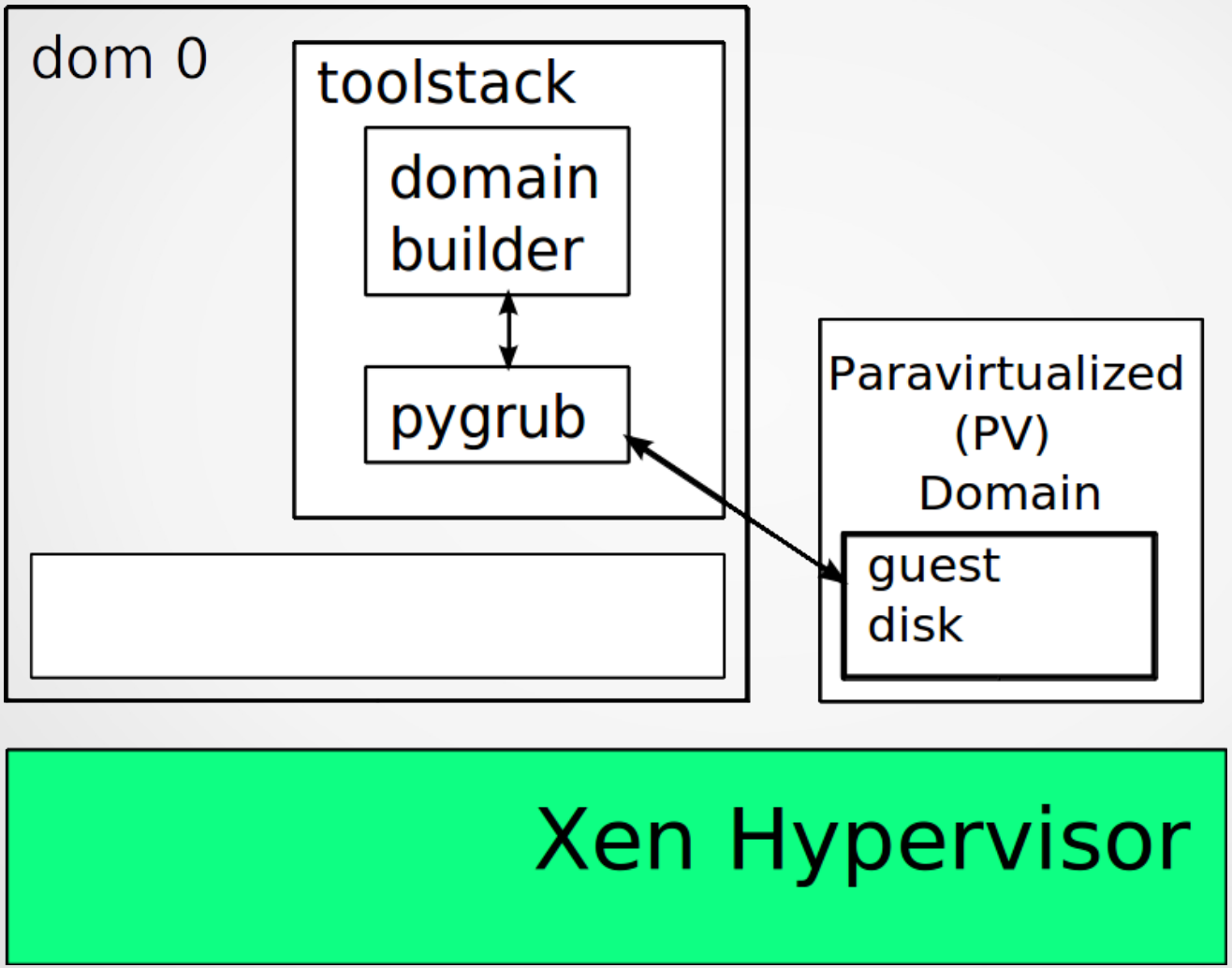
vif = ['type=pv, bridge=xenbr0, backend=domnet']

Detailed Info

http://wiki.xenproject.org/wiki/Driver_Domain

Attacking the PyGrub Boot Loader

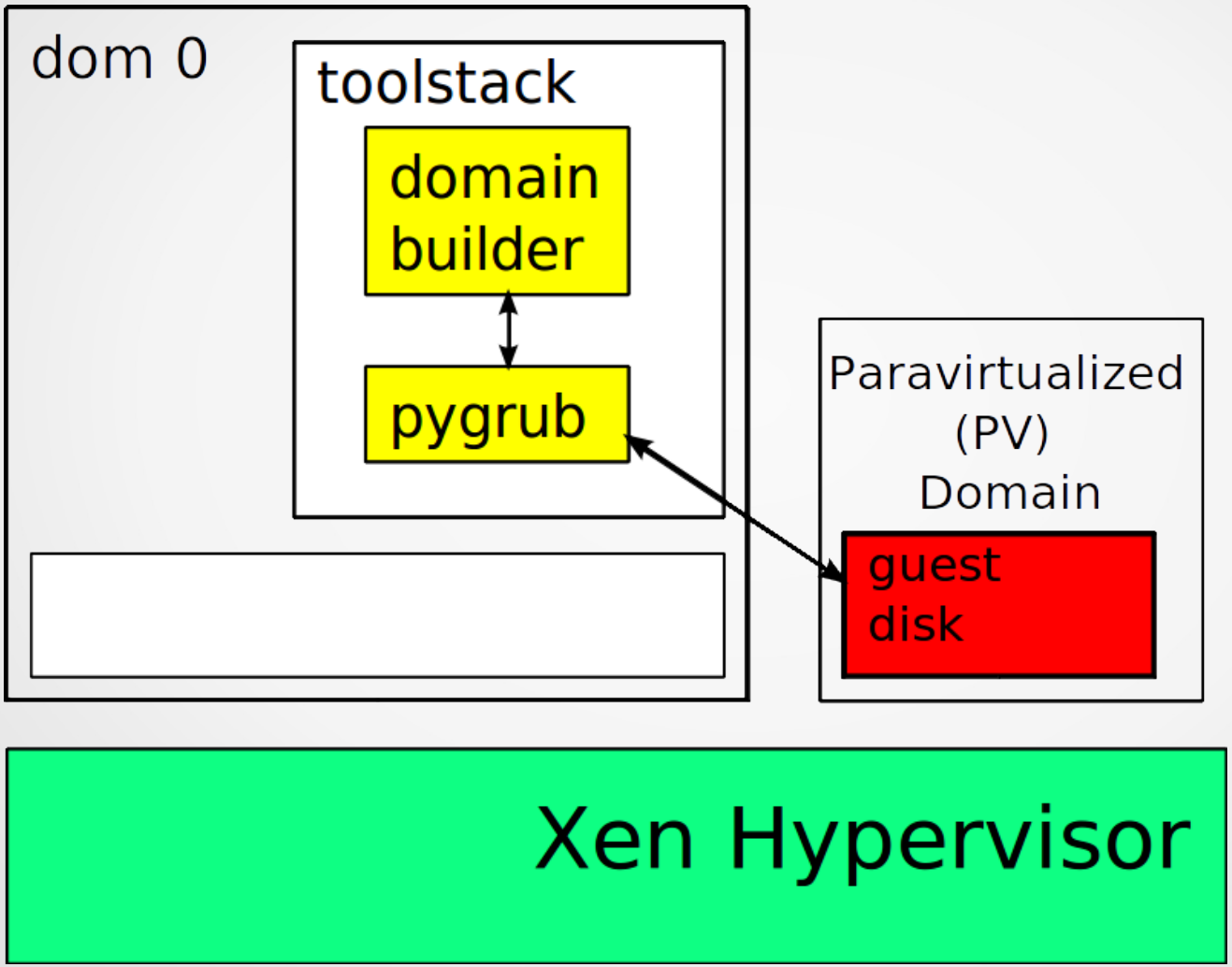
Attack Surface: PyGrub



Attack Surface: PyGrub

- What is PyGrub?
  - “grub” implementation for Paravirtualized guests
  - A Python program running in Control Domain
- What does it do?
  - It reads the guest VM's filesystem
  - It parses grub.conf
  - It displays a boot menu to the user
  - It passes the selected kernel image to domain builder

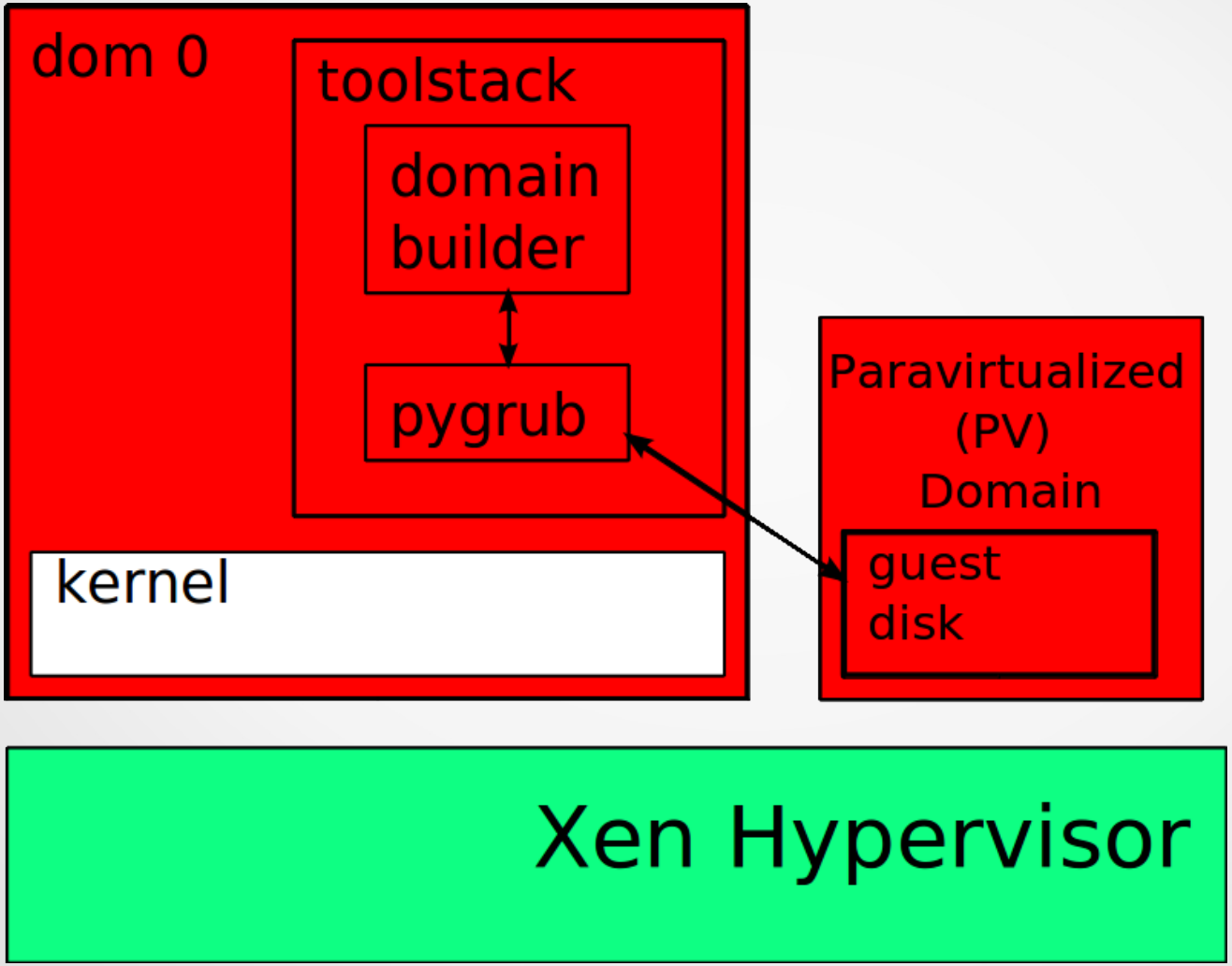
Attack Surface: PyGrub



Attack Surface: PyGrub

- Where might an exploit focus?
  - Bugs in file system parser
  - Bugs in menu parser
  - Bugs in domain builder
- Again, note the exploits
  - Forms of these exist in hardware as well
  - But hardware doesn't have as many options to combat the situation

Result: PyGrub Compromised



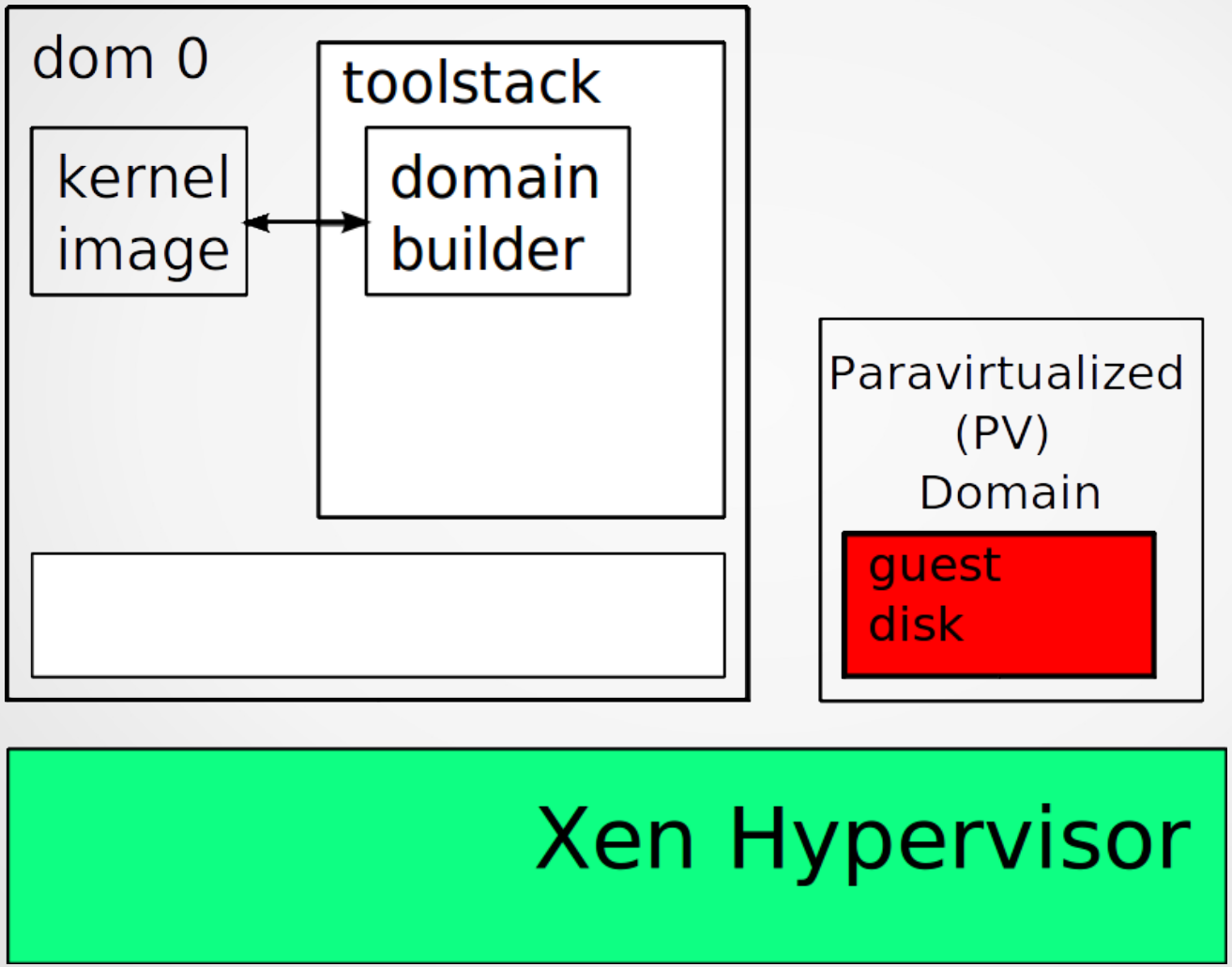
Result: PyGrub Compromised

Vulnerability Analysis:

What could a successful exploit yield?

- Control of Domain 0 user space
- **This could lead to the control of the whole system**

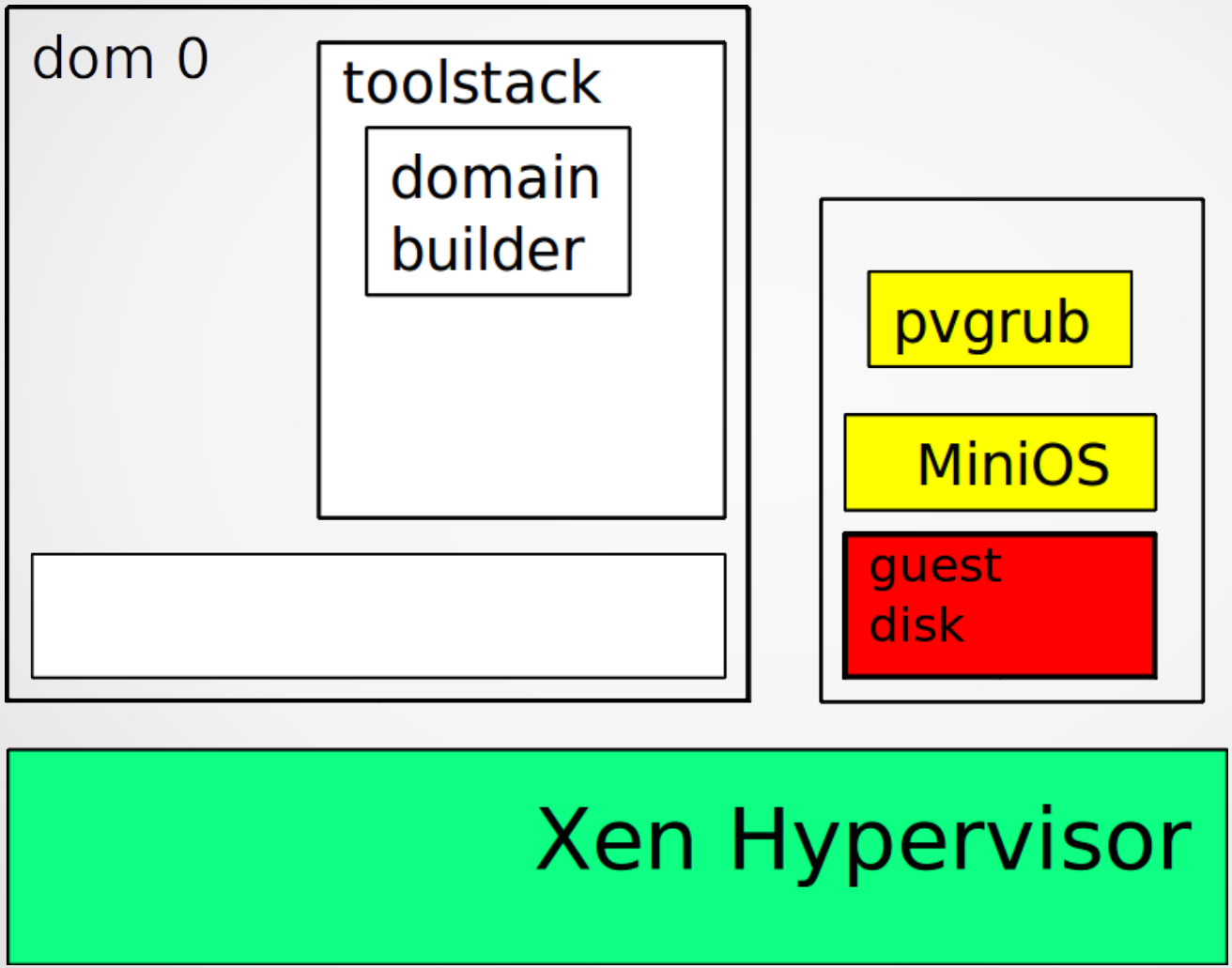
Security Feature: Fixed Kernels



Security Feature: Fixed Kernels

- What is a fixed kernel?
  - Passing a known-good kernel from Control Domain
    - No longer allows a user to choose the kernel
    - Best practice for anything in production
  - Removes attacker avenue to domain builder
- Disadvantages
  - Host administrator must keep up with kernel updates
  - Guest admin can't pass kernel parameters or custom kernels

Security Feature: PVgrub



Security Feature: PVgrub

- What is PVgrub?
  - MiniOS plus the Paravirtualized port of “grub” running in a guest context
  - Paravirtualized equivalent of Hardware Virtualized combination of BIOS plus grub

Result: PVgrub Compromised

Vulnerability Analysis:

Now a successful exploit could yield:

Control of the attacked Guest Domain alone

Control Domain is no longer at risk

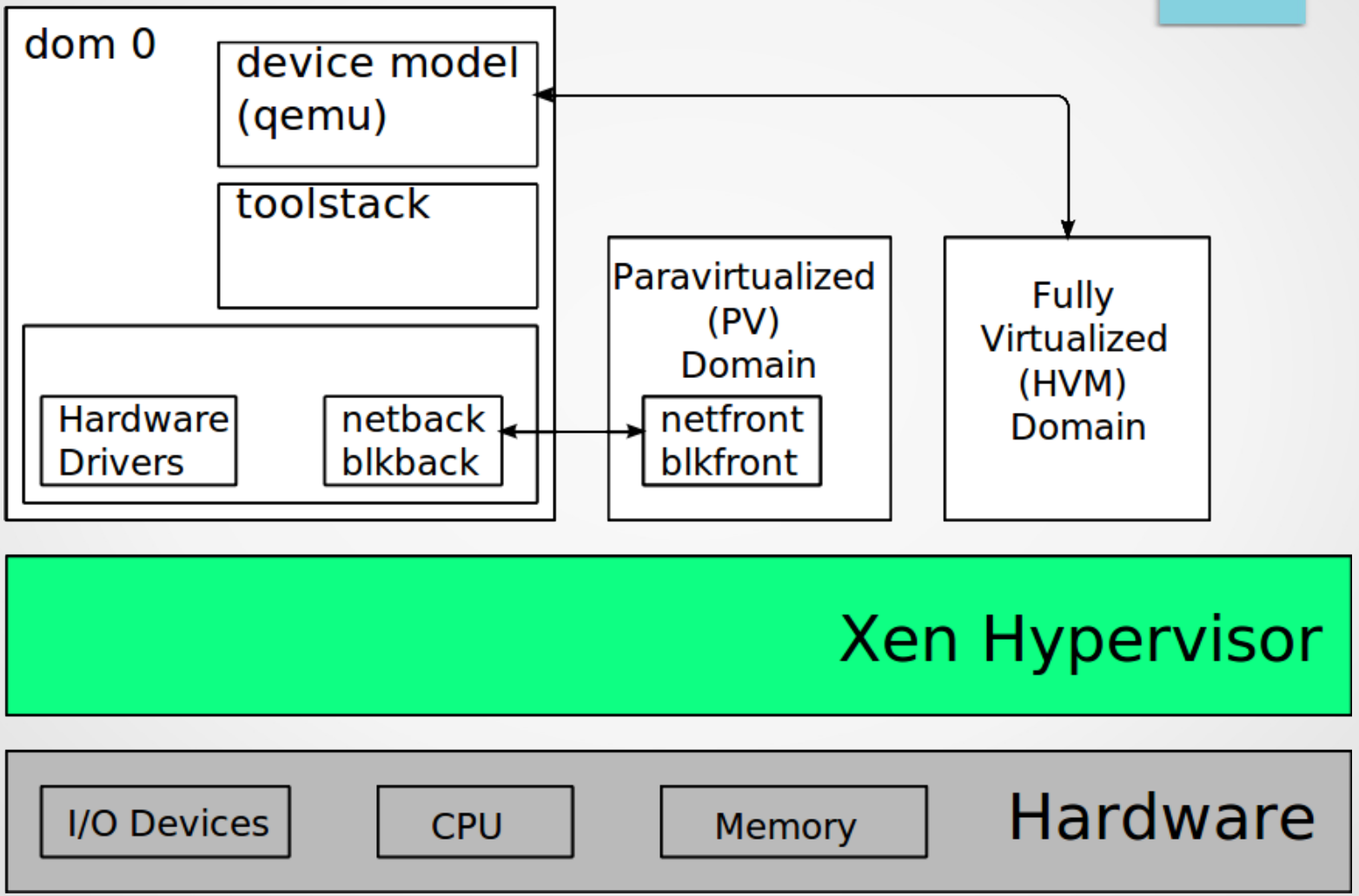
Basic HowTo: PVgrub

- Make sure that you have the PVgrub image
  - “pvgrub-$ARCH.gz”
  - Normally lives in “/usr/lib/xen/boot”
  - Debian, SLES: Currently need to build for yourself
  - Included in Fedora Xen Project packages
- Use appropriate PVgrub as bootloader in guest configuration:
  - kernel="/usr/lib/xen/boot/pvgrub-x86_32.gz"

See http://wiki.xenproject.org/wiki/Pvgrub

Attacking the Qemu Device Model

Attack Surface: Device Model (Qemu)



Attack Surface: Device Model (Qemu)

- What is Qemu?
  - In other contexts, a virtualization provider
  - In the Xen Project context, a provider of needed device models
- Where might an exploit focus?
  - Bugs in NIC emulator parsing packets
  - Bugs in emulation of virtual devices

Result: Device Model Compromised

Vulnerability Analysis:

What could a successful exploit yield?

Control Domain privileged user space

This could lead to the control of the whole system

Security Feature: Qemu Stub Domains

- What is a stub domain?
  - **Stub domain:** a small “service" domain running just one application
  - **Qemu stub domain:** run each Qemu in its own domain

Result: Stub Domain Compromised

Vulnerability Analysis:

Now a successful exploit could yield:

Control only of the stub domain VM
(which, if FLASK is employed, is a relatively small universe)

You need to devise another attack entirely to do anything more significant

Basic HowTo: Qemu Stub Domains

- Make sure that you have the *ioemu* image:
  - “ioemu-$ARCH.gz”
  - Normally lives in “/usr/lib/xen/boot”
  - SUSE SLES, currently need to build it yourself (SLES 12?)
  - Included in Fedora Xen Project packages
  - On Debian (and offshoots), you will need to build it yourself
- Specify stub domains in your guest configuration:

  device_model_stubdomain override = 1

Detailed Info

http://wiki.xenproject.org/wiki/Device_Model_Stub_Domains

Attacking the Hypervisor Itself

Attack Surface: The Hypervisor Itself

- Where might an exploit focus?
  - On Paravirtualized (PV) Guests:
    - PV Hypercalls
  - On full Hardware Virtualized (HVM) Guests:
    - HVM hypercalls (Subset of PV hypercalls)
    - Instruction emulation (MMIO, shadow pagetables)
    - Emulated platform devices: APIC, HPET, PIT
    - Nested virtualization
- Security practice: Use PV VMs whenever possible

Using the Xen Project Security Module

Security Feature: FLASK Policy

- What is FLASK?
  - Xen Security Module (XSM): Xen Project equivalent of LSM
  - FLASK: FLux Advanced Security Kernel
  - Framework for XSM developed by NSA
  - Xen Project equivalent of SELinux
  - Uses same concepts and tools as SELinux
  - Allows a policy to restrict hypercalls

Security Feature: FLASK Policy

- What can FLASK do?
  - Basic: Restricts hypercalls to those needed by a particular guest
  - Advanced: Allows more fine-grained granting of privileges
- FLASK example policy
  - This contains example roles for the Control Domain (dom0), User/Guest Domain(domU), stub domains, driver domains, etc.
  - Make sure you TEST the example policy in your environment BEFORE putting it into production!

NOTE: As an example policy, it is not as rigorously tested as other parts of Xen during release; make sure it is suitable for you

---

Basic HowTo: FLASK Example Policy

- Build Xen Project software with XSM enabled
- Build the example policy
- Add the appropriate label to guest config files:
  - “seclabel=[foo]”
  - “stubdom_label=[foo]”

Detailed Info

http://wiki.xenproject.org/wiki/Xen_Security_Modules_:_XSM-FLASK

ARM-Specific Security Features

Xen Project + ARM = A Perfect Match

ARM SOC

Device Tree describes …

GT	GIC v2	2 stage MMU

ARM Architecture Features for Virtualization

User mode : EL0

Kernel mode : EL1

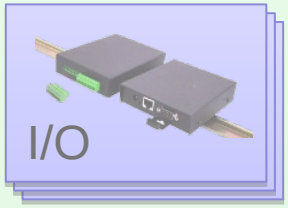
Hypercall Interface HVC

Hypervisor mode : EL2

Xen Project + ARM = A Perfect Match

ARM SOC

Device Tree describes …



ARM Architecture Features for Virtualization

EL0

EL1

HYC

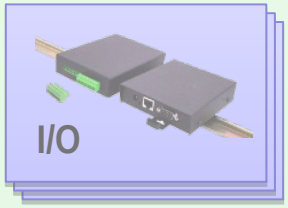
Xen Project Hypervisor

Xen Project + ARM = A Perfect Match

ARM SOC

ARM Architecture Features for Virtualization

Device Tree describes …



Any Xen Project Guest VM (including Dom0)

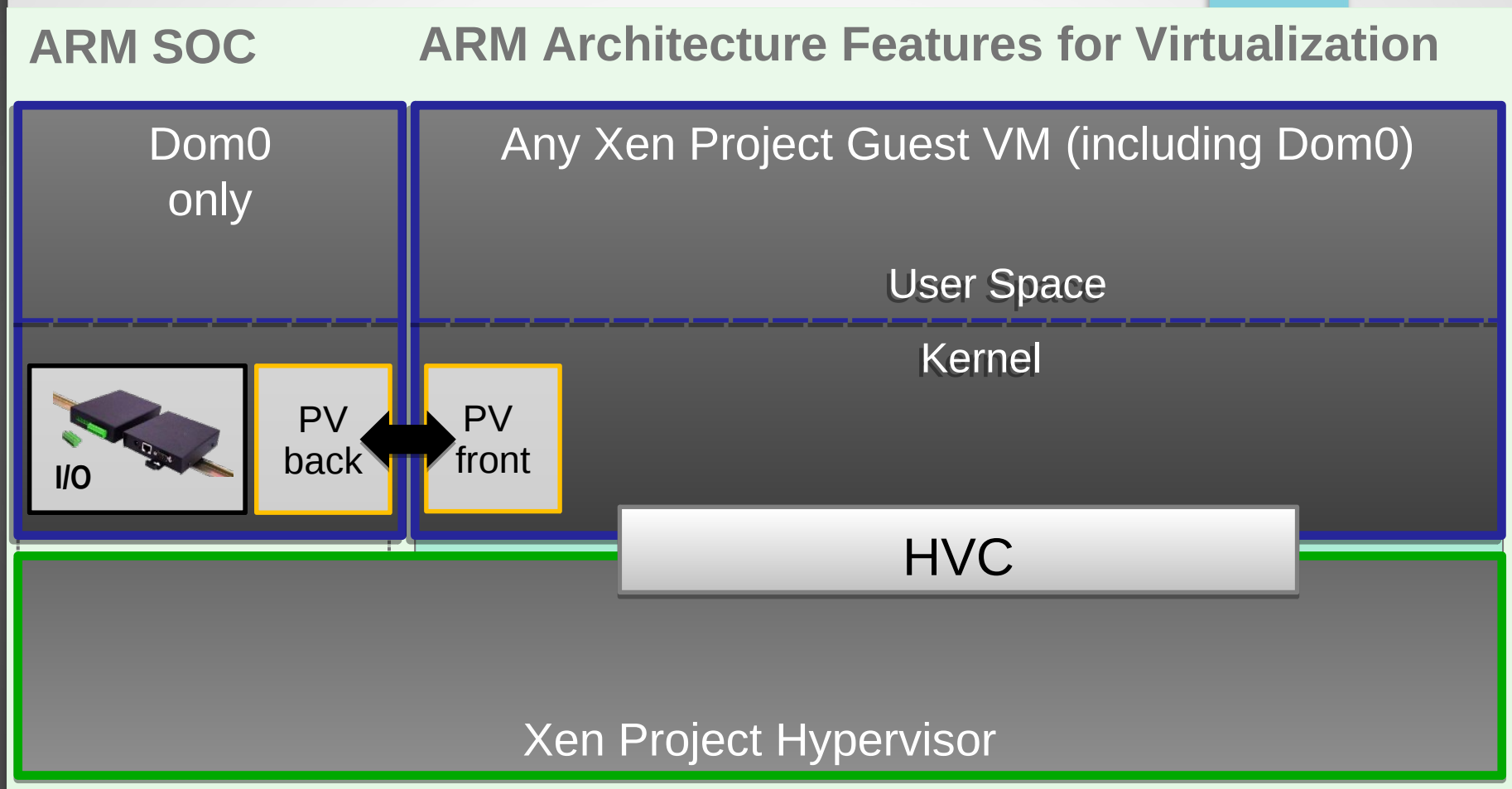
User Space

Kernel

HVC

Xen Project Hypervisor

Xen Project + ARM = A Perfect Match



ARM: Right Solution for Security

- Stays in ARM Hypervisor Mode
  - The ARM architecture has separate Hypervisor and Kernel modes
  - Because Xen Project's architecture maps so well to the ARM architecture, the hypervisor never has to use Kernel mode
  - Other hypervisors have to flip back and forth between modes
  - If a hypervisor has to enter Kernel mode, it loses the security of running in a privileged mode, isolated from the rest of the system
  - This is a non-issue with the Xen Project Hypervisor on ARM
- Does not need to use device emulation
  - No emulation means a smaller attack surface for bad guys

For More Information...

Detailed Info

http://wiki.xenproject.org/wiki/Securing_Xen

Thanks to George Dunlap for supplying much of the information presented here, and Stefano Stabellini for ARM information

Center of the Xen Project universe:
http://www.XenProject.org/

Contact me at russell.pavlicek@xenproject.org

Thank You!