Looking to the Future

decreasing trust in dom0, and *client virtualization on ARM*

Kyle J. Temkin
Assured Information Security, Inc.
OpenXT Summit, June 7th 2016

Who am I?

devastating capability, revolutionary advantage

Kyle J. Temkin

- Security Researcher at AIS, Inc.
- Open source contributor, maintainer
- Low-level hardware guy:
  - Hardware design for security
  - Hypervisor/VMM-based security
  - Kernel/driver work

Feel free to get in touch:

- kyle@ktemkin.com
- @ktemkin on twitter, ktemkin on freenode

Talk format

devastating capability, revolutionary advantage

This talk will have a slightly unusual format.

This is really a pair of ~10 minute "lightning talks":

- **Reducing Trust in "dom0"**
  in which we re-explore disaggregate models made easier by modern Xen
- **Client Virtualization for ARM**
  in which we explore the challenges of creating OpenXT-like functionality on ARM

Important note:
I don't claim to be a definitive source, especially in terms of past disaggregation work. My goal is to encourage discussion– time is allotted for it tomorrow!

Reducing Trust in "dom0"

establishing a trusted foundation for *OpenXT*

Kyle J. Temkin
Assured Information Security, Inc.
OpenXT Summit, June 7th 2016

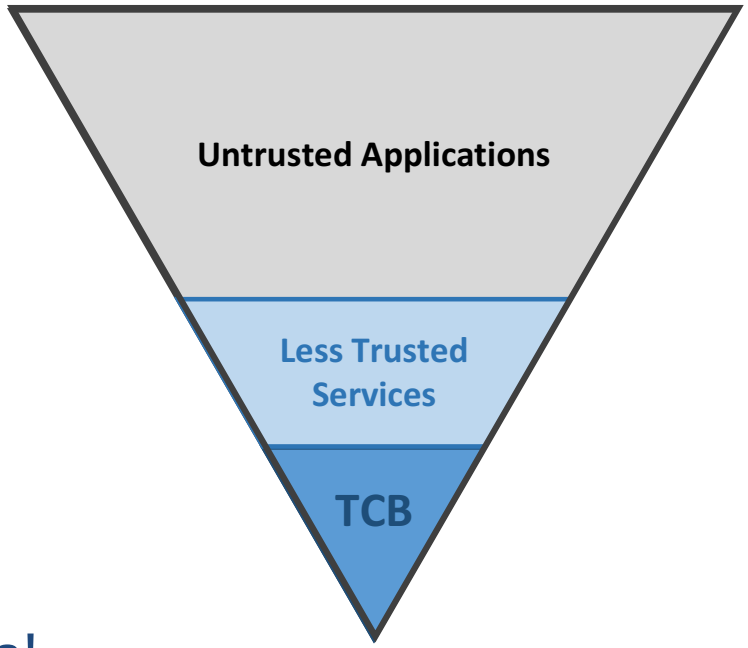
Background: Trusted Computing Base

Limitation:

It's not possible to audit and trust all components of a complex system. Instead, we trust a minimal selection of hardware and software—our trusted computing base—and build complex systems on top of it.

General Goals:

- TCB should be kept as small as possible.
- Platform services should have the least privilege possible.



Compromise of the TCB typically leads to compromise of the entire platform and its data!

OpenXT: Trusted Computing Base

devastating capability, revolutionary advantage

Unfortunately, pragmatism often leads to large TCBs.

- Together, the *Xen hypervisor*, *dom0 kernel*, and OpenXT toolstack comprise more than 1M SLoC.
  - A meaningful audit of this much security-critical code is intractable.

Existing mitigation technologies (Flask/XSM, SELinux) aren't enough:

- Compromising a privileged component yields all of its privileges.
  - dom0 is currently a trusted domain; the OpenXT *Flask/XSM* policy provides it enough privilege that it can easily compromise every VM on the system
  - The OpenXT toolstack requires similar privileges to do its job: as it acts as the loader for every non-dom0 VM on the system, it is currently granted arbitrary memory access privileges (`map_foreign`).
- dom0 kernel vulnerabilities **do exist**, and they're often **not obvious**!

Example Vulnerability:
Double Fetches in the dom0 kernel

devastating capability, revolutionary advantage

Description:

special case of TOCTTOU in which shared memory is inappropriately considered static, despite changes being possible from the "other side"

Example:

```
//Synthetic example
if(vm_authorized_for_command(shared_memory_buffer)) { //1
    time_consuming_prepatory_task();
    execute_command(shared_memory_buffer);            //2
} else {
    pr_warn("Access denied!\n");
}
```

Double fetch bugs have been around for a while; and were recently leveraged against Xen in Felix Wilhelm (@_fel1x)'s *XenPwn* talk.

XSA-155:
Double Fetches in e.g. *pciback* (dom0 kernel)

Felix Wilhelm was able to exploit a subtle double fetch in *pciback* to gain significant control over the dom0 kernel.

- This kind of attack could be leveraged from an NDVM against *pciback in dom0*.
  - This hurts one of our nicer properties: ideally, a compromised NDVM wouldn't be able to impact other platform components.
- This kind of attack is made easier by a privileged domain that provides services directly to a guest.
- **This doesn't have to be the case: we can design systems to be resilient!**

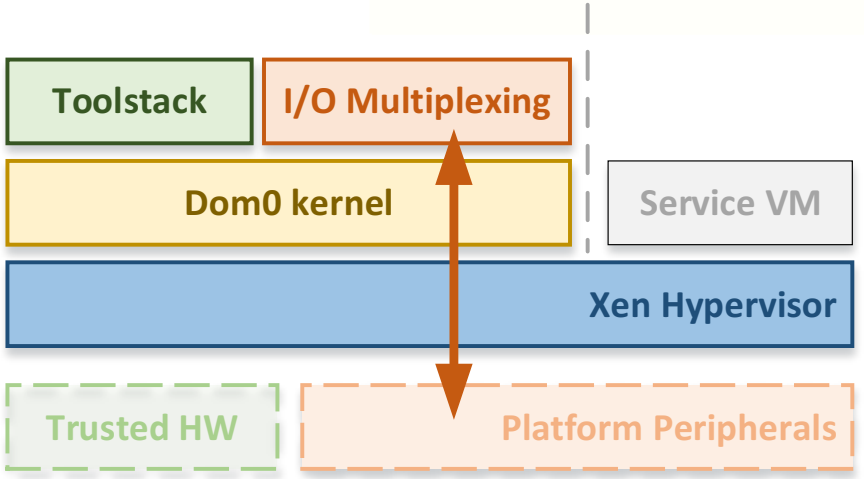
```
switch (op->cmd) {
    case XEN_PCI_OP_conf_read:
            op->err = xen_pcibk_config_read(dev,
                      op->offset, op->size, &op->value);
            break;
    case XEN_PCI_OP_conf_write:
            //...
    case XEN_PCI_OP_enable_msi:
            //...
    case XEN_PCI_OP_disable_msi:
            //...
    case XEN_PCI_OP_enable_msix:
              //...
    case XEN_PCI_OP_disable_msix:
            //...
    default:
            op->err = XEN_PCI_ERR_not_implemented;
            break;
}
```

The non-obvious vulnerable section in pciback.
Source: XenPwn, Felix Wilhelm, Infiltrate 2016

Core Issue: 'polymath' dom0

In this case, the current OpenXT architecture enables the attacker:

- dom0 has too many responsibilities:
  - Hosts drivers for interfacing with most platform hardware.
  - Provides services (e.g. PV backends) to guest domains.
  - Manages and controls all other domains, and thus requires significant privileges.



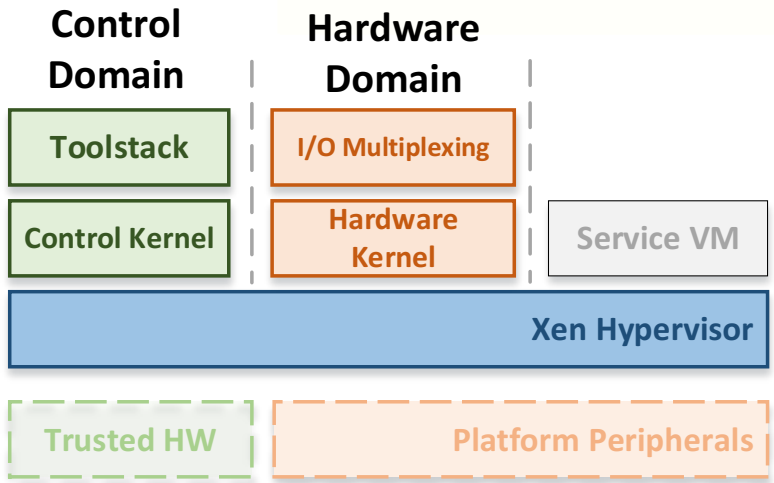
Core idea: Components with significant privileges shouldn't aggregate responsibilities; and shouldn't share attack surfaces with untrusted guests, or untrusted world-facing hardware.

First-step solution: disaggregation

devastating capability, revolutionary advantage

Disaggregation distributes responsibilities between **multiple dedicated domains**.

- OpenXT already separates network hardware into its own NDVM; and PCI passthrough allows similar treatment for other devices (USB, storage, display*).
- As of Xen 4.5, Xen finally separates the concept of the *initial domain* (dom0) from the idea of a *hardware domain* and a *control domain*. This allows us some significant new freedoms!

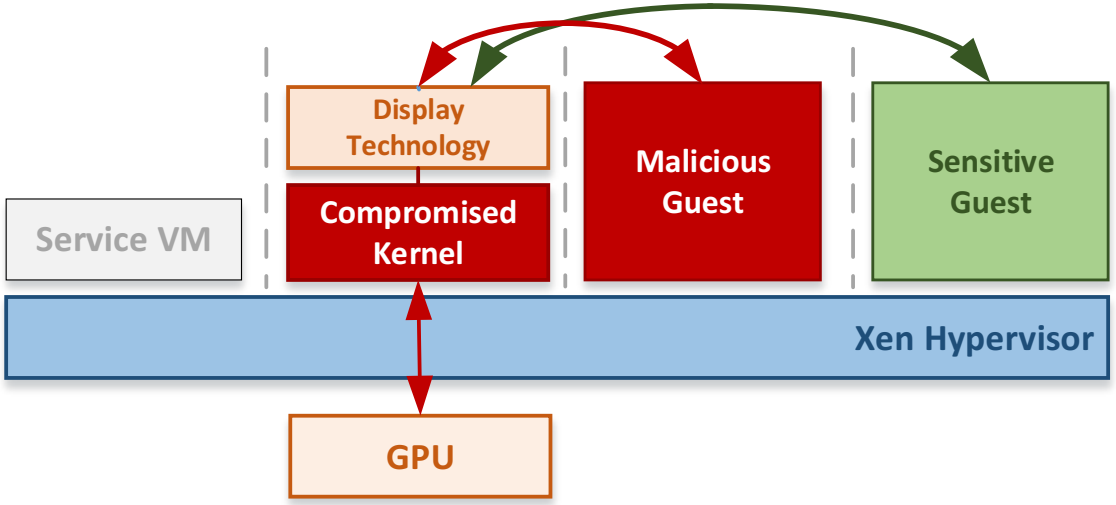


- Software components (e.g. the toolstack, the xenstore) can be further broken into multiple domains, allowing further compartmentalization and increase FLASK/XSM granularity.

Warning: Disaggregation compartmentalizes, reduces *shared* attack surface, and allows for finer-grained permissions, but doesn't automatically reduce the size of the TCB.

Disaggregation: limitations

Disaggregation can significantly limit the scope of an attack, but it doesn't automatically solve existing problems.



- Vulnerable components will likely remain vulnerable even in a disaggregate model—limited scope attacks are still attacks!
- Deprivileging components doesn't remove them from the TCB: sensitive data may still be passing through them!

“Advanced Techniques”

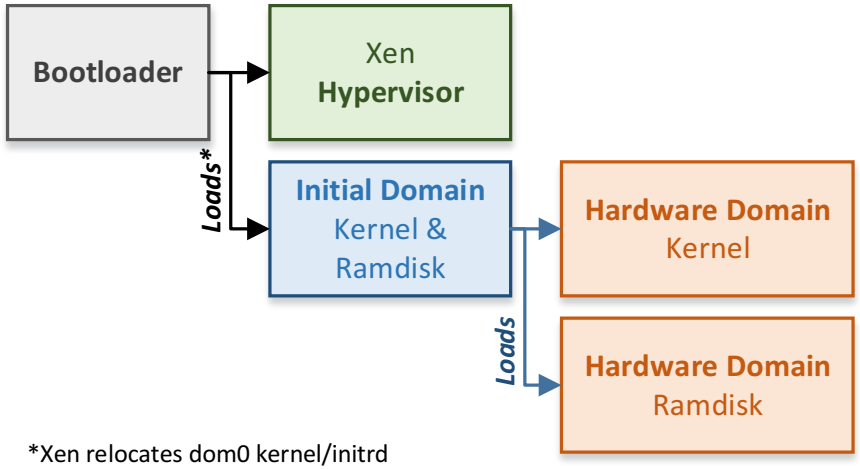
devastating capability, revolutionary advantage

Techniques from other areas of trusted computing can significantly augment disaggregation:

- **Preventing or limiting persistence can further limit the scope of an exploit.** Virtual machines that exist only for the duration of a request, or which are restarted or refreshed periodically (as in the Xoar/Bear architectures) significantly limit the scope of a compromise. These benefits are compounded when service VMs self-diversify.
- **Replacing complex “full-distribution” systems with unikernels can significantly reduce the size of the trusted computing base.** This technique comes with a trade-off: replacing general-purpose software with purpose-built equivalents can significantly reduce attack surface, but often significantly reduces real-world testing.

Daniel De Graaf's HWDOM Patchset…

Xen's **initial domain** (dom0) is unique in that it is constructed **by Xen**.
By default, it is granted all hardware resources and significant privilege.



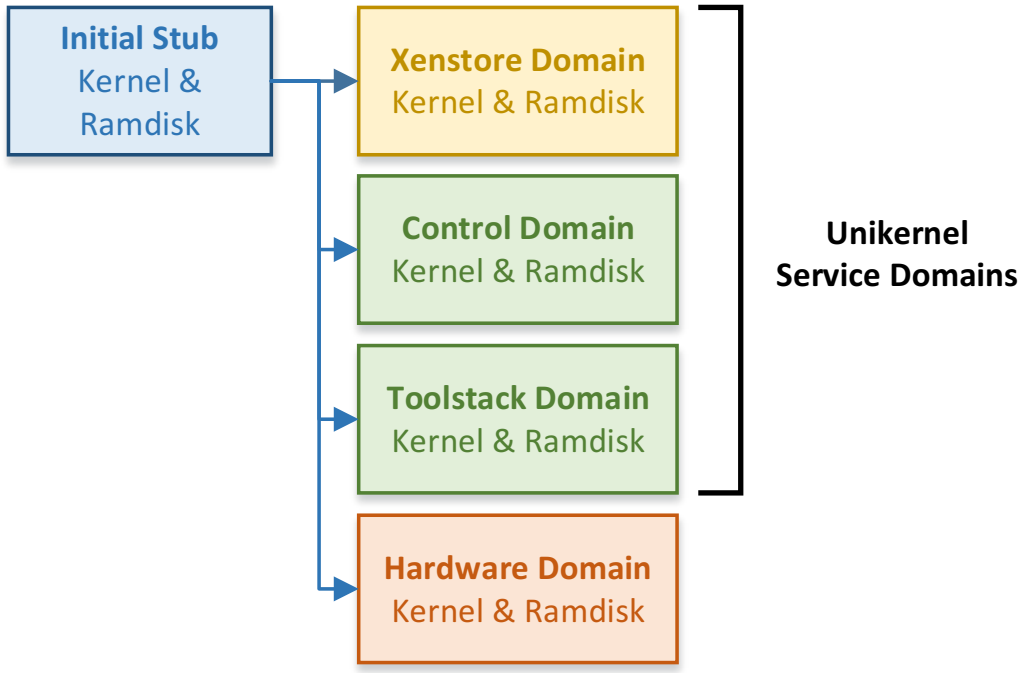
*Xen relocates dom0 kernel/initrd

Included in Xen 4.5, a **NSA patchset** allows deferred construction of the hardware domain, with the intention of enabling disaggreate models.

In this model, a privileged domain can be constructed by Xen without implied hardware access; this domain can later construct the hardware domain.

… and associated domain builder.

Included with the patch set was a **reference domain builder**, which serves to bootstrap a disaggregated system.

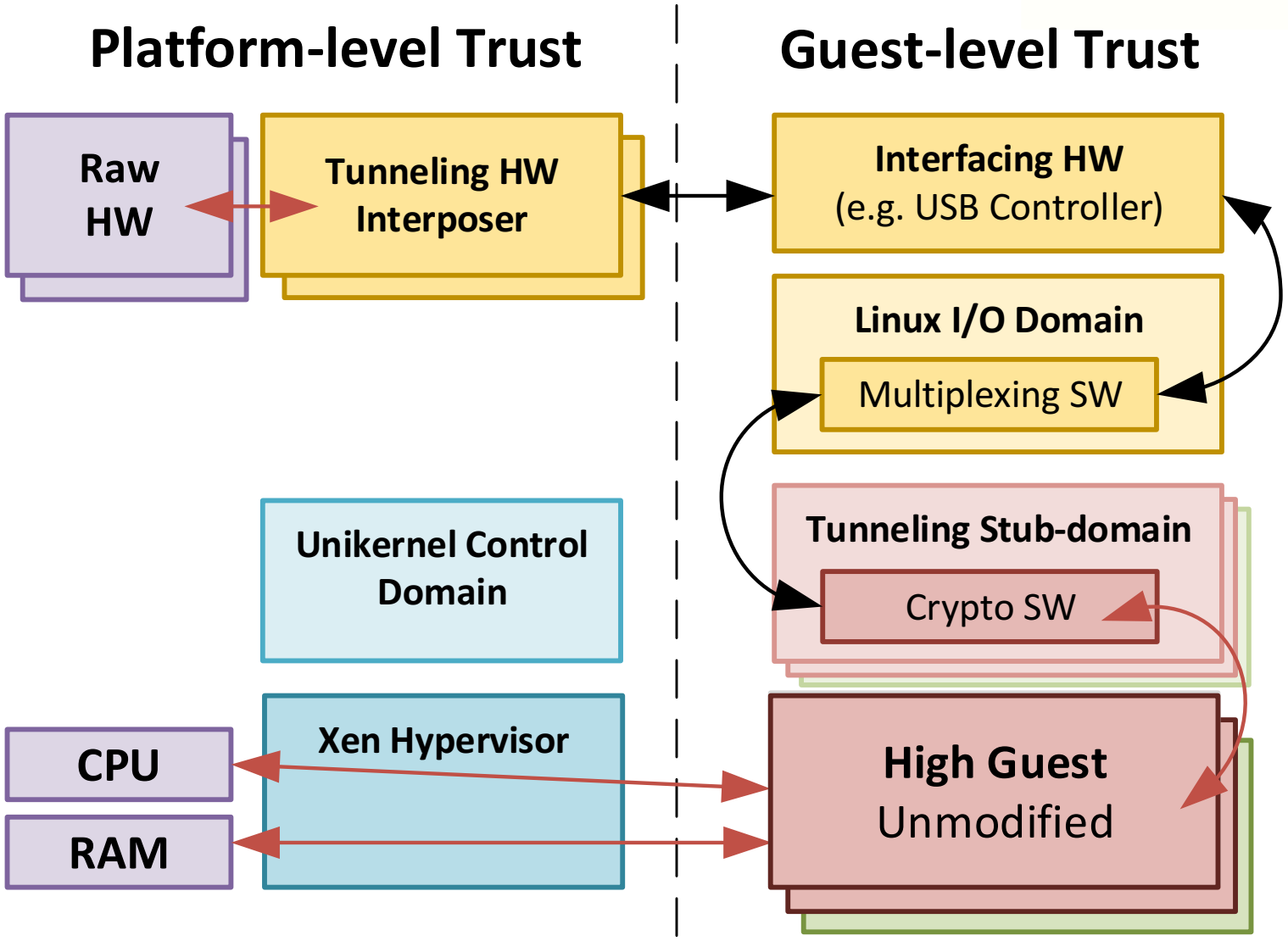


This domain builder provides an example of several **unikernel service domains**, providing purpose-built environments for e.g. the XenStore.

SOLID & Non-Traditional Tunneling

ais

devastating capability, revolutionary advantage



Question Break

Client Virtualization on ARM

towards virtualization for cell phones, tablets, and embedded

Kyle J. Temkin
Assured Information Security, Inc.
OpenXT Summit, June 7th 2016

153 Brooks Road, Rome, NY | 315.336.3306 | http://ainfosec.com

Client Virtualization for Mobile Devices

devastating capability, revolutionary advantage

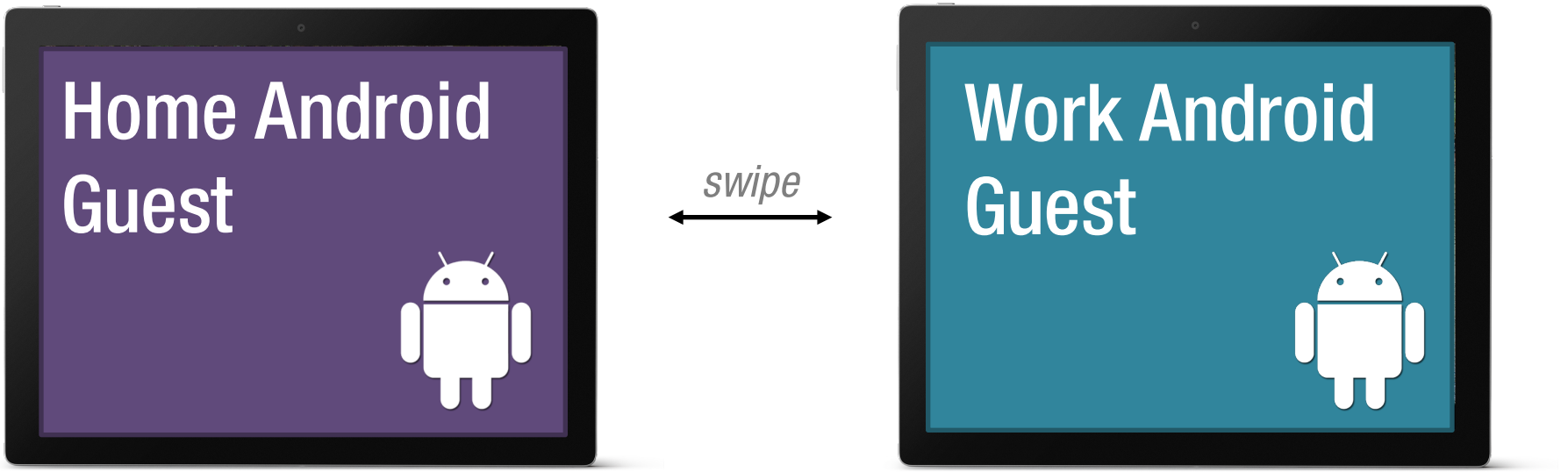
The rise of mobile processors with virtualization extensions presents new opportunities for client virtualization.

- If anything, the benefits of client virtualization (consolidation, isolation) are more important on mobile.
  - While it's often practical to own multiple computers, carrying multiple phones or tablets quickly reduces portability.
- In the private sector, organizations often have difficulty separating personal and 'official' data and applications.
  - especially difficult for BYOD ('Bring your own device') IT
- In the government sector, mobile devices can provide portable access solutions.

Secure Phones/Tablets: Ideal Case

devastating capability, revolutionary advantage

An ideal **client virtualization solution** would provide the an experience consistent with a baremetal mobile device, but will allow VM switching.

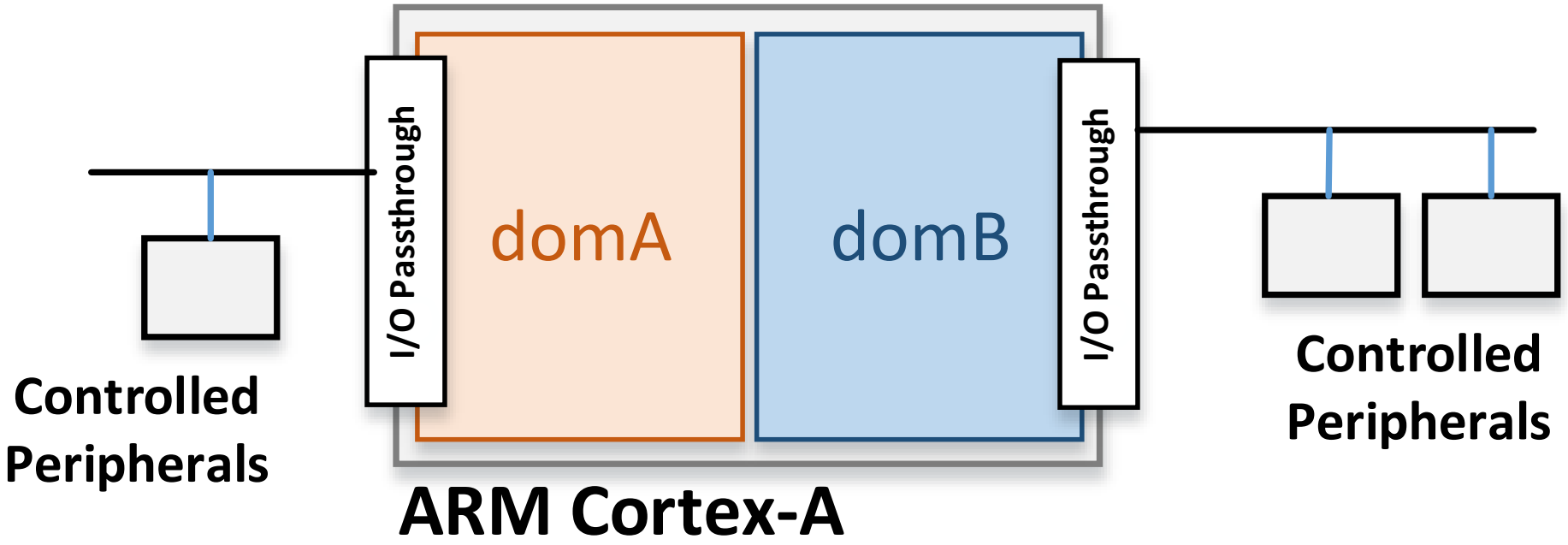


Pictured: *Google Pixel C*

Several companies have phone/tablet hypervisor platforms (e.g. VMware), but no commercial or open-source offering is currently available.

Embedded Hypervisors

Embedded hypervisors can be used to support legacy applications in non-user-facing systems, allowing similar consolidation.



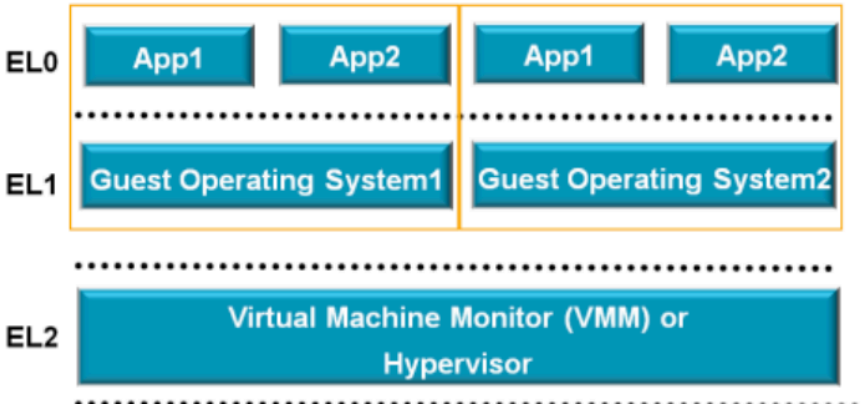
Embedded hypervisors straddle the line between client and server virtualization: while not user-facing, they often support peripheral HW.

ARM Virtualization Extensions

Modern ARMv7-A (32-bit) and ARMv8-A (64-bit) processors support virtualization extensions, which provide VT-x-like hardware support.

ARM's virtualization extensions:

- Provide similar capabilities to Intel's offerings, though with some significantly differing mecahnisms.
- Lack the near-ubiquitous support characteristic of Intel's virtualization tech.
  - Only select cores support the virtualization extensions, and most commercial phones and tablets prohibit you from using them!



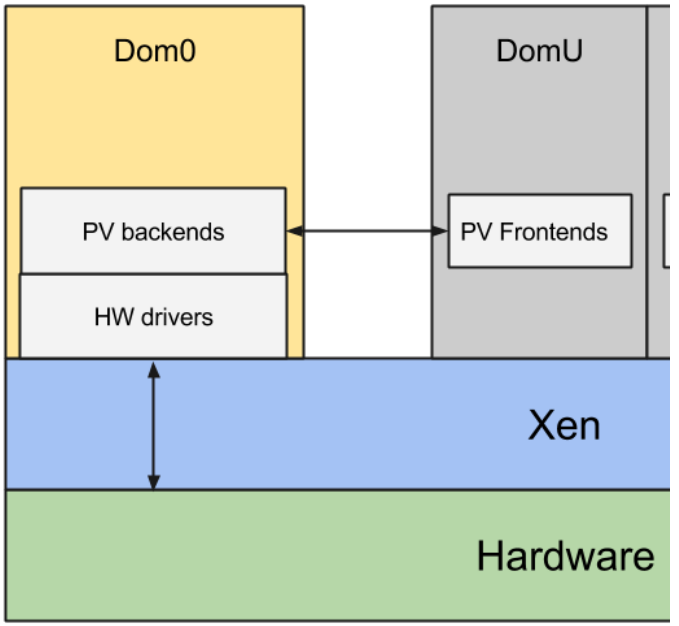
Source: *ARM ARMv8 Whitepaper*

Upstream Xen on ARM

The upstream Xen community has been actively working to support ARM platforms; Xen-on-ARM has been usable since about Xen 4.4.0.

ARM support is a "refinement" of the existing Xen architecture:

- Only one type of guest: PVH.
  - This means no required device models, but drivers must be Xen-aware.
  - Performance-wise, this seeks to be the best of both worlds: hardware-assisted virtualiztaion without emulated devices.
- Unified virtualization support allows for a smaller codebase: 90k SLOC vs 150 k SLOC



Source: *XenProject "Xen on ARM" Whitepaper*

Challenge: Hardware Availability

Only a limited number of commercially available phones and tablets support the ARM virtualization extensions.

It can be challenging to find COTS phones and tablets for ARM development:

- The Google Pixel C (pictured) is an ARMv8 tablet that supports hypervisor applications and has case-closed access to a debug UART.
- Allwinner tablets (inexpensive, insecure) tend to provide simple ARMv7 development boards.
- Many development boards now support hypervisor applications; and OEM partnerships may yield hypervisor-enabling bootloaders.

Challenge: Hardware Support

devastating capability, revolutionary advantage

The implementation of ARM hardware is significantly more varied than its x86 counterparts, down to the details of individual chipsets.

ARM is an IP vendor, not an IC manufacturer: most IC vendors build their own componentry around an existing ARM core. As a result, Xen modifications are often required to support new SoC families.

Examples:

- The NVIDIA Tegra family (ARMv7, ARMv8) has a special "legacy" interrupt controller that must be correctly programmed by Xen.
  - Not yet supported in upstream Xen; we have a WIP patchset to address this.
- The Exonos family has a non-standard timer architecture; platform quirks exist upstream to get timers working.

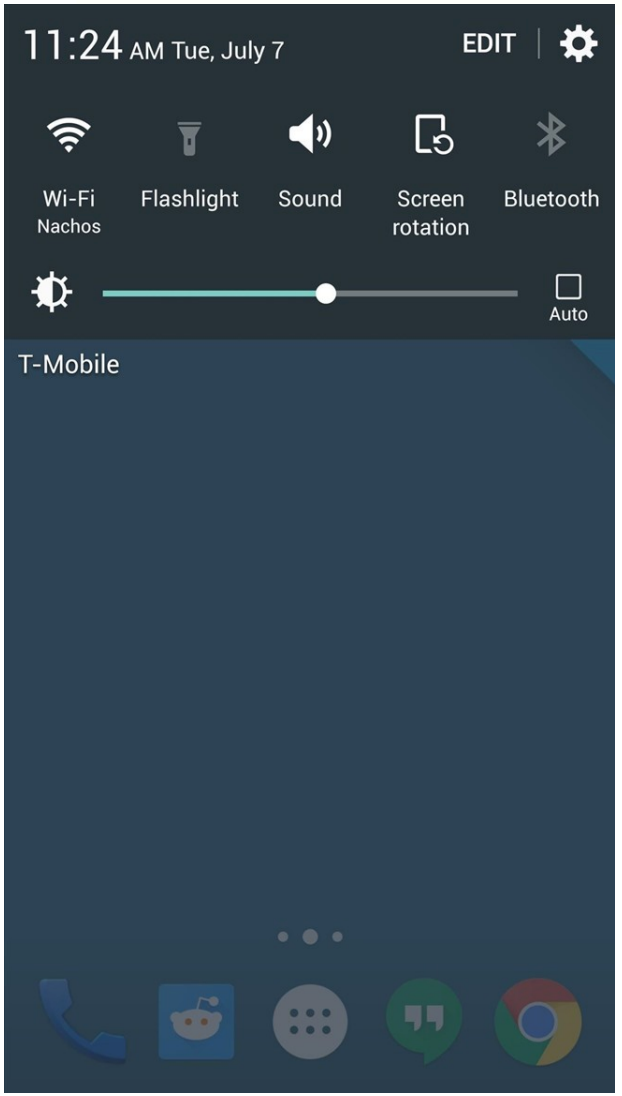
Challenge: Android Support

devastating capability, revolutionary advantage

As PVH guests require PV drivers, guest OS builds must be customized to run on Xen.

Android's Linux-kernel core makes support for PV drivers relatively straightforward; but some sections are are more complex.

- Android versions starting with 4.0 technically require a (v)GPU with OpenGL ES support.
  - Out of tree patches enable software rendering; but significant use of transparency leads to poor performance, especially in terms of power.
- Peripheral devices (accelerometers, gyroscopes, light sensors) require new classes of PV driver.



Open Questions

devastating capability, revolutionary advantage

Questions for community consideration:

- Should OpenXT expand into the ARM arena?
  - If so, how do we want to proceed in the future?
  - Does this provide value to existing derived products?

- How would an ARM project mesh with the OpenXT roadmap?

Questions?

devastating capability, revolutionary advantage

153 Brooks Road, Rome, NY | 315.336.3306 | http://ainfosec.com