Qubes OS
Towards Secure & Trustworthy Personal Computing

Joanna Rutkowska
Invisible Things Lab and Qubes OS Project
Qubes OS

- A reasonably secure OS for *personal computers* (currently: laptops & desktops)
- Security by Compartimentalization
- Qubes ≠ Hypervisor/VMM (Qubes is a *user* of a VMM, presently Xen)
- Qubes ≠ Linux Distro
Why?
We really need secure personal computers…

- Client systems are our Eyes, Ears, and Fingers!
- … are extensions of our brains!
- Nothing works when the client system is compromised
  - Crypto
  - (2-factor) authentication
  - VDI/thin terminals (“secure cloud” not secure)
Present client systems are... insecure
Problems with current (desktop) systems

- Attacks coming through (exploited) apps (Web browser, PDF readers, etc)
- Attacks coming from (malicious) apps (Spyware, Backdoors, etc)
- Attacks coming from (compromised or malicious) USB devices
- Attacks coming through networking stacks (DHCP client, WiFi driver/stacks)
- Attacks coming through (malformed) FS/volume metadata (USB Storage, CDs)
- Lack of GUI isolation (sniffing content & clipboard, sniffing & spoofing keystrokes)
- Malware persisting itself in the BIOS, other firmware
Monolithic systems are hard to secure (especially desktop systems!)
Monolithic kernel is bad for security

- WiFi & NIC & BT drivers & stacks
- USB drivers & stacks
- Filesystem modules & other volume processing code
- All the various APIs (e.g. debug, VFS, sockets API, etc)
- Why should all these be part of TCB?
(Big) TCBs we don’t like!
Trust

- Trusted: Can ruin the whole system's security! (this is by definition)
- Secure: Resistant to attacks, but might be malicious! (e.g. well written backdoor)
- Trustworthy: Resistant to attacks and also “good” (whatever that really means!)
“Monolithic” is not only about the kernel...
Monolithism beyond kernel

- GUI server (Xorg)
- Various system services
  - Network Manager and other D-Bus endpoints
  - udev services (e.g. block device mounting)
  - CUPS, desktop indexing, etc
- Not only root considered as “TCB” from user-data point of view
  - e.g. “root-less” Xorg not a big deal, really
- Finally: the hardware (think: all the devices making up our laptops)
Monolithic means:
bloated, complex, difficult to understand, and manage
How?
Solving the “trust problem”?

Ensure TCB is “good”

i.e.: secure (difficult to attack)

Anti-exploitation DEP, ASLR, etc

i.e.: not backdoored

Code review, Opensourceing

Get rid of the TCB (as much as possible!)

No need to think if its “good” or not anymore!

Qubes OS
Security by Isolation
(or Compartmentalization)
Virtualization?

- Yes, we use virtualization (VMs) to isolate domains from each other...
- But why would VMs provide any better isolation than OS processes?
- Is there anything wrong with x86 good old MMU/page/ring separation?
- “Solving” problems by adding another layer of abstraction?
What so special about Virtualization?

- It allows to **REDUCE** the interfaces (VM-VM & VM-TCB)...
- ... and preserve compatibility with **LEGACY apps & drivers** at the same time
But before we get too excited...
VM<->hypervisor is not the only interface that is security critical...
Strong isolation “by virtualization”...

... not anymore!
Lesson:

- Don’t get too excited about “hardware virtualization” isolation
  - Virtualization nothing magic, offers little more than traditional MMU isolation
  - Except for IOMMU, but that’s for devices, more later
  - Admittedly also SLAT (e.g. Intel EPT) reduces code for memory virtualization

- Be careful about inter-VM interfaces and code that handles it!
Ask your hypervisor/VMM vendor if/how they DO:

- Device emulation (is qemu part of TCB?)
- Networking virtualization (is net backend part of TCB?)
- Storage virtualization (protocols used, any fancy & complex features?)
- USB virtualization (is USB backend part of TCB?)
- GUI virtualization (also OpenGL/DirectX/GPU backend complexity?)
- Inter-VM communication framework?
- Inter-VM file & clipboard copy?
“Virtualization gold rush” brought some useful new h/w technology though...
IOMMU (AKA Intel VT-d)

- Allows for truly de-privileged driver domains (Xen pioneer in using it)
- We can have NetVM, UsbVM :)
- BTW, microkernels without IOMMU made no sense from security point of view.
Qubes OS sandboxes networking h/w and stacks

- Ever used WiFi in an airport lounge or hotel?
- Ever wondered if your WiFi driver, stack or DHCP client could be exploited?
  - Remember Bashocalypse?
- How about sandboxing all these components?
- This is what a NetVM is about
Qubes OS sandboxes USB h/w and stacks

- How much code involved in processing when plugging in a USB device?
  - BadUSB?
- UsbVM can sandbox all the USB drivers and stacks
- Then we can carefully export select devices to other AppVMs
BONUS: resistance to malicious hardware!

- Backdoored NICs
- Backdoored USB devices
- Theoretically we could also deal with a malicious disk and SATA controller
  - But much harder, requires trusted boot in addition to IOMMU, not implemented yet.
- Theoretically we could also deal with a malicious GPU device
  - In practice GPU is part of the processor package (if we don’t trust it, we’re screwed anyway)
Qubes OS vs. BadBIOS?

- But the BIOS/SMM is a different story…
- While Qubes makes it hard for malware to infect the firmware…
- … it cannot do much to protect itself from one that is already compromised :(
  - In theory Intel TXT should help, in practice it does not
- … it can do even less to protect itself from a malicious agent in the processor!
  - Intel ME :(
Compartmentalization vs. Integration
What makes you think each of these VMs won’t get compromised independently sooner or later?
What makes you think each of these VMs won’t get compromised independently sooner or later?
Isolation itself doesn’t solve many problems...
Smartly used *integration* of isolated containers can provide benefits though.
Other examples of smart integration

- Qubes Split GPG
- Qubes Tor/VPN VMs
- Qubes File Copy
- Using untrusted storage medium (e.g. USB stick) for trusted data
- ...
What is Qubes OS, really?

- Qubes implements “Security by Isolation” using some VMM (currently Xen)
- But, more importantly: clever integration of the isolated compartments
  - “clever” = not weakening the isolation
- “Security by Compartmentalization”
- Focus on personal computing (desktops, not servers!)
But today personal computers are not just insecure, they are also untrustworthy…
Trustworthiness as a challenge for the OS

- Development security (source-code-level backdoors)
- Build security (binary-level backdoors)
- Dealing with potentially backdoored hardware?
Status
Qubes OS Releases

- Release 1 (2010-2012)
- Release 2 (2012-2014)
  - HVM & Windows support, gazillion other features
- Release 3 (2014-2016)
  - 3.0, 3.1, etc.
  - Hypervisor Abstraction Layer, Mgmt/pre-configuration stack,
  - Debian, Whonix templates
  - UX & H/W compatibility improvements
- Release 4 (2016-…)

Qubes OS as a platform for secure/privacy-oriented Apps

- Integration with Tor
  - TorVM since 2012
  - Full Whonix integration since 2014
- Secure email
  - Open attachments in Disposable VMs
  - Split GPG to protect user private keys
  - PDF converter (make PDFs trusted)
- Secure networking
  - Isolated VPN VMs
- More coming!
Qubes OS Master Key Fingerprint

427F11FD 0FAA4B08 0123F01C DDFA1A3E 36879494

qubes-os.org
@QubesOS