QUBES OS

Joanna Rutkowska
Invisible Things Lab
Qubes OS

- A reasonably secure desktop OS
- Security by Compartmentalization
- Qubes != Hypervisor/VMM (Qubes is a user of a VMM, presently Xen)
- Qubes != Linux Distro
Why?
Because we need secure client systems
We really need secure CLIENT systems

• Client systems are our Eyes, Ears, and Fingers!
• 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 through (compromised) 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)
Desktop systems ≠ server systems
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?
“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
Monolithic means: bloated, complex, difficult to understand, and manage
How?
Security by 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)

• Be careful about inter-VM interfaces and code that handles it!
Ask your hypervisor 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.
NetVM

• 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
USBVM

- 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
Monolithic system

Tradeoff between usability & security?

Powered-down “Airgaps”
Qubes OS Releases

- Qubes OS R1
  - 2010-2012

- Qubes OS R2 (HVM & Windows support, gazillion other features)
  - 2012-2014

- Qubes OS R3 (Hypervisor Abstraction Layer, UX improvements, H/W compat)
  - 2013-
Qubes R2 implements everything we talked about so far (plus more!)

qubes-os.org
Use of Linux (and other OSes)

- Currently default template based on Fedora 20
- Debian and ArchLinux templates also available (community contributions)
- Also our Dom0 based on Fedora 20
  - But this mostly irrelevant to the user, as no user apps or data are in Dom0
  - (Think about Dom0 as a thin and dumb terminal to work with AppVMs)
- Windows 7-based templates also supported
  - User must install Windows and provide licensing keys though
Qubes as a platform for secure/privacy-oriented Apps

- Integration with Tor
  - TorVM since 2012
  - Currently on-going work to fully integrate Whonix
- 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 R3 ("Odyssey")

- Hypervisor Abstraction Layer (HAL)
  - Don’t like Xen?
  - No problem, use KVM, LXC, MS Hyper-V, [some academic u-kernel/hypervisor]
  - Allows for security-performance-compatibility tradeoffs

- Reworked architecture
  - More modular, even more decomposed
  - GUI domain != Admin domain (planned)
  - Qubes Admin API: semi-untrusted remote management VM(s) (planned)
Wednesday, October 15 • 2:30pm - 4:20pm

Tutorial: Qubes OS: Practical Intro for Users and Developers - Joanna Rutkowska & Marek Marczykowski-Gorecki, Invisible Things Lab
MASTER KEY FINGERPRINT

427F 11FD 0FAA 4B08 0123 F01C DDFA 1A3E 3687 9494
THANKS!