Qubes OS: an overview on the most reasonably secure distro

A presentation for NCSULUG by Jackson Quigley

Overview

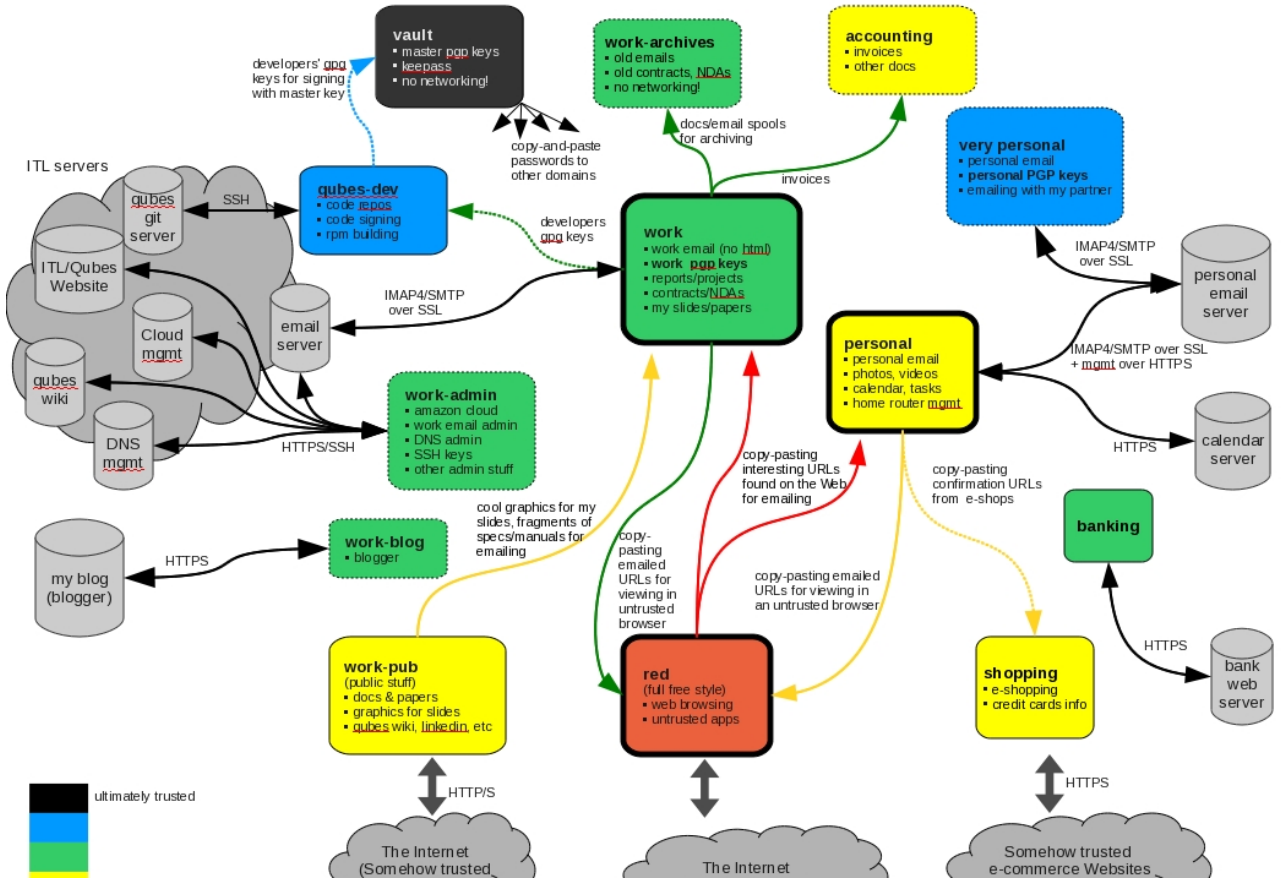
- What is Qubes?
- Who is Qubes for?
- Why is Qubes different?
- How does Qubes work?
- Using Qubes

Disclaimer

- I have barely used Qubes
- You will also likely never use it
- Doesn’t like being used in virtual box
- Painful hardware compatibility list (IOMMU mobo/VT-x/AMD-V/x86)
- Lots of ram usage
- Can’t demo it because you can’t screenshare (no one qube has full display control)

What is Qubes

- Security first operating system (assume all software has exploitable bugs)
- Operating system of operating systems
- Rank trust of programs or specific tasks
- Compartmentalization
- Compartmentalization
- Compartmentalization



[varia] Software Guard Extensions Programming Reference

34 of 156 | Software Guard Extensions Programming Refer... 329298-001.pdf | 88.66%

ENCLAVE OPERATION

After AEX has completed, the logical processor is no longer in enclave mode and the exiting event is processed normally. Any new events that occur after the AEX has completed are treated as having occurred outside the enclave (e.g. a #PF in dispatching to an interrupt handler).

3.2.3 Resuming Execution after AEX

After system software has serviced the event that caused the logical processor to exit an enclave, the logical processor can re-start execution using ERESUME. ERESUME restores registers and returns control to where execution was interrupted.

If the cause of the exit was an exception or a fault and was not resolved, the event will be triggered again if the enclave is re-entered using ERESUME. For example, if an enclave performs a divide by 0 operation, executing ERESUME will cause the enclave to attempt to re-execute the faulting instruction and result in another divide by 0 exception. In order to handle an exception that occurred inside the enclave, software can enter the enclave at a different location and invoke the exception handler within the enclave by executing the EENTER instruction. The exception handler within the enclave can attempt to resolve the faulting condition or simply return and indicate to software that the enclave should be terminated (e.g. using EEXIT).

3.2.3.1 ERESUME Interaction

ERESUME restores registers depending on the mode of the enclave (32 or 64 bit).

- In 32-bit mode (IA32_EFER.LMA = 0 || CS.L = 0), the low 32-bits of the legacy registers (EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI, EIP and EFLAGS) are restored from the thread's GPR area of the current SSA frame. Neither the upper 32 bits of the legacy registers nor the 64-bit registers (R8 ...R15) are loaded.
- In 64-bit mode (IA32_EFER.LMA = 1 && CS.L = 1), all 64 bits of the general processor registers (RAX, RBX, RCX, RDX, RSP, RBP, RSI, RDI, R8 ...R15, RIP and RFLAGS) are loaded.

Extended features specified by SECS.ATTRIBUTES.XFRM are restored from the XSAVE area of the current SSA frame. The layout of the x87 area depends on the current values of IA32_EFER.LMA and CS.L:

- IA32_EFER.LMA = 0 || CS.L = 0
  - — 32-bit load in the same format that XSAVE/FXSAVE uses with these values.
- IA32_EFER.LMA = 1 && CS.L = 1
  - — 64-bit load in the same format that XSAVE/FXSAVE uses with these values plus REX.W = 1

3.3 CALLING ENCLAVE PROCEDURES

3.3.1 Calling Convention

In standard call conventions subroutine parameters are generally pushed onto the stack. The called routine, being aware of its own stack layout, knows how to find parameters based on compile-time-computable offsets from the SP or BP register (depending on runtime conventions used by the compiler).

Because of the stack switch when calling an enclave, stack-located parameters cannot be found in this manner. Entering the enclave requires a modified parameter passing convention.

For example, the caller might push parameters onto the untrusted stack and then pass a pointer to those parameters in RAX to the enclave software. The exact choice of calling conventions is up to the writer of the edge routines; be those routines hand-coded or compiler generated.

3.3.2 Register Preservation

As with most systems, it is the responsibility of the callee to preserve all registers except that used for returning a value. This is consistent with conventional usage and tends to optimize the number of register save/restore opera-

[Dom0] Qubes VM Manager

System VM View About

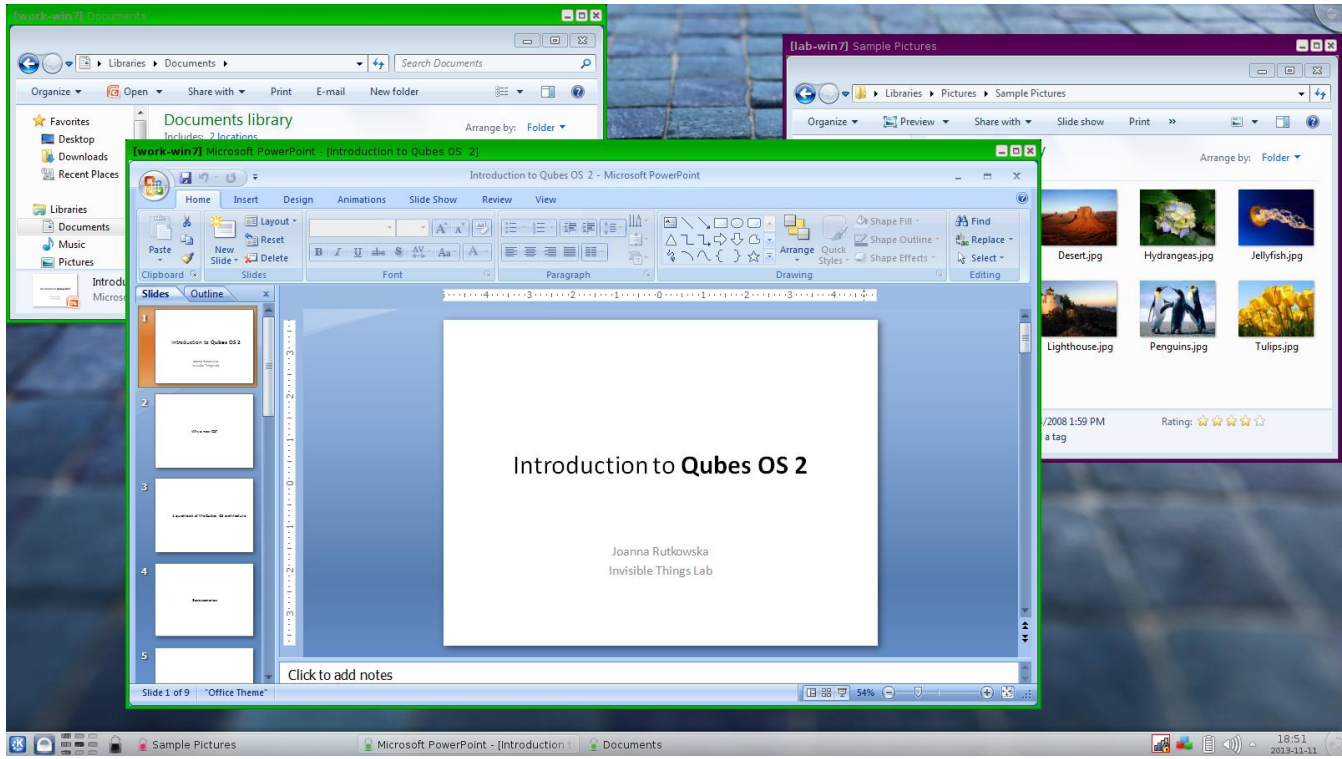
Name	State	NetVM	CPU Graph	MEM
dom0		n/a		2598 MB
sys-net		n/a		301 MB
sys-firewall		sys-net		301 MB
varia		sys-firewall		979 MB
work-web		sys-firewall		1173 MB
work-mutt		sys-firewall		604 MB
keys-itl-email		---		478 MB
work		sys-firewall		607 MB
personal		sys-firewall		750 MB

[work] user@work:~

```
[user@work ~]$
```

Who is Qubes for

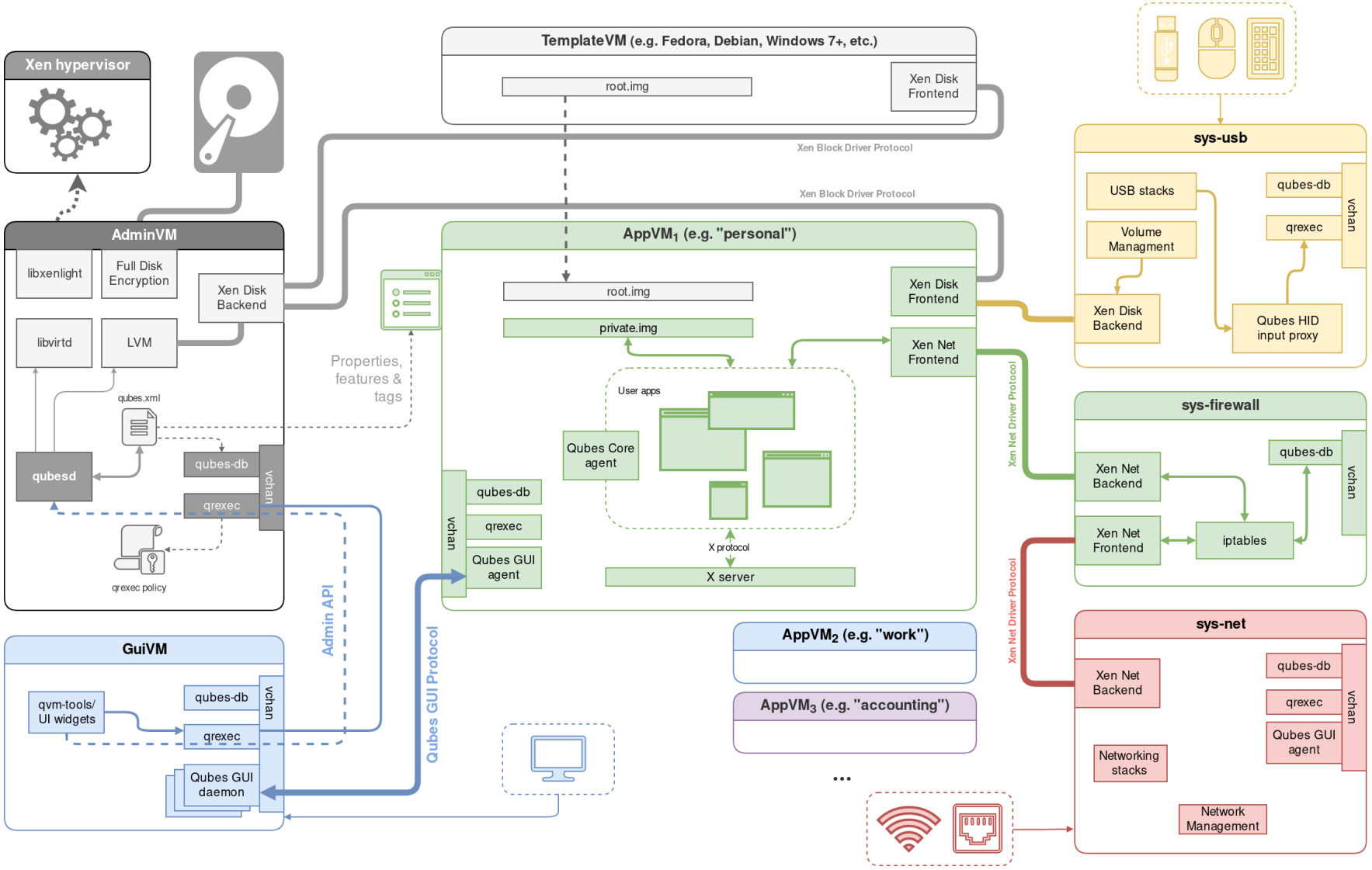
- Extremely security conscious people
- Isolate work from personal computing
- Running windows VMs in a more seamless fashion
- Easy network testing between qubes



Why is Qubes different

- Nothing interacts between domains without you specifically asking
- Xen hypervisor to control all operating systems
- Two levels of control dom0 and domU
- Visual control of security levels
- Easy firewall switching and Whonix integration
- Seamless integration of VMs

How does Qubes work



The virtual machines

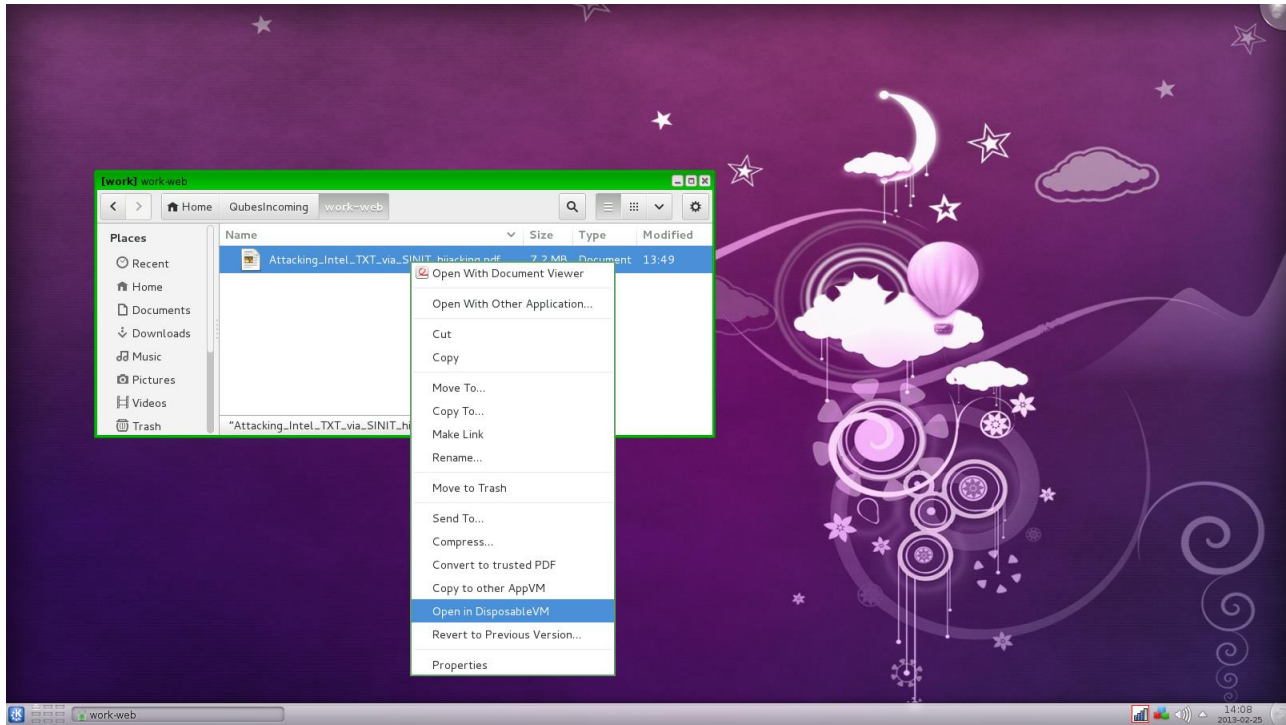
- Domain vs template
- Templates are read only for qube
- Download and update template updates qube
- Each qube has its own home directory
- Fedora and Debian official templates
- Whonix, Ubuntu, Arch, and Gentoo community templates
- StandaloneVM doesn’t share root with qubes
- Windows/BSD require standalone VMs

The virtual machines cont.

- Hardware Virtual Machine (HVM) vs Paravirtualized (PV) VMs
- Any iso can be run as a standalone HVM
- PV allows hypervisor (Xen) to bypass CPU emulation giving better performance (no virtualization tech needed on CPU)
- HVM require CPU virtualization
- Qubes uses PVH, I/O and network PV but rest of OS in HVM

Disposable VM

- Doesn’t create filesystem
- Open files in disposable VM edits pushed to origin
- Open untrusted files and VM is deleted after closing



Using Qubes

- Obtuse at times
- Clipboarding from one domain to another requires moving clip up to dom0 and then back down to final domain
- Similar requirement for file transfers
- Customizing dom0 is a pain
- USB requires extra mounting steps to allow qube to access it
- Attention paid to color of boarder, shows trust
- Booting new qube takes time
- Fullscreen? What's that?

Sources

- The Qubes OS documentation
- https://www.qubes-os.org/doc/