Qubes OS R2 Tutorial

INVISIBLE THINGS LAB

LINUXCON EUROPE, OCT 2014, V1.0-RC1

Agenda

Part 1 (for Users)

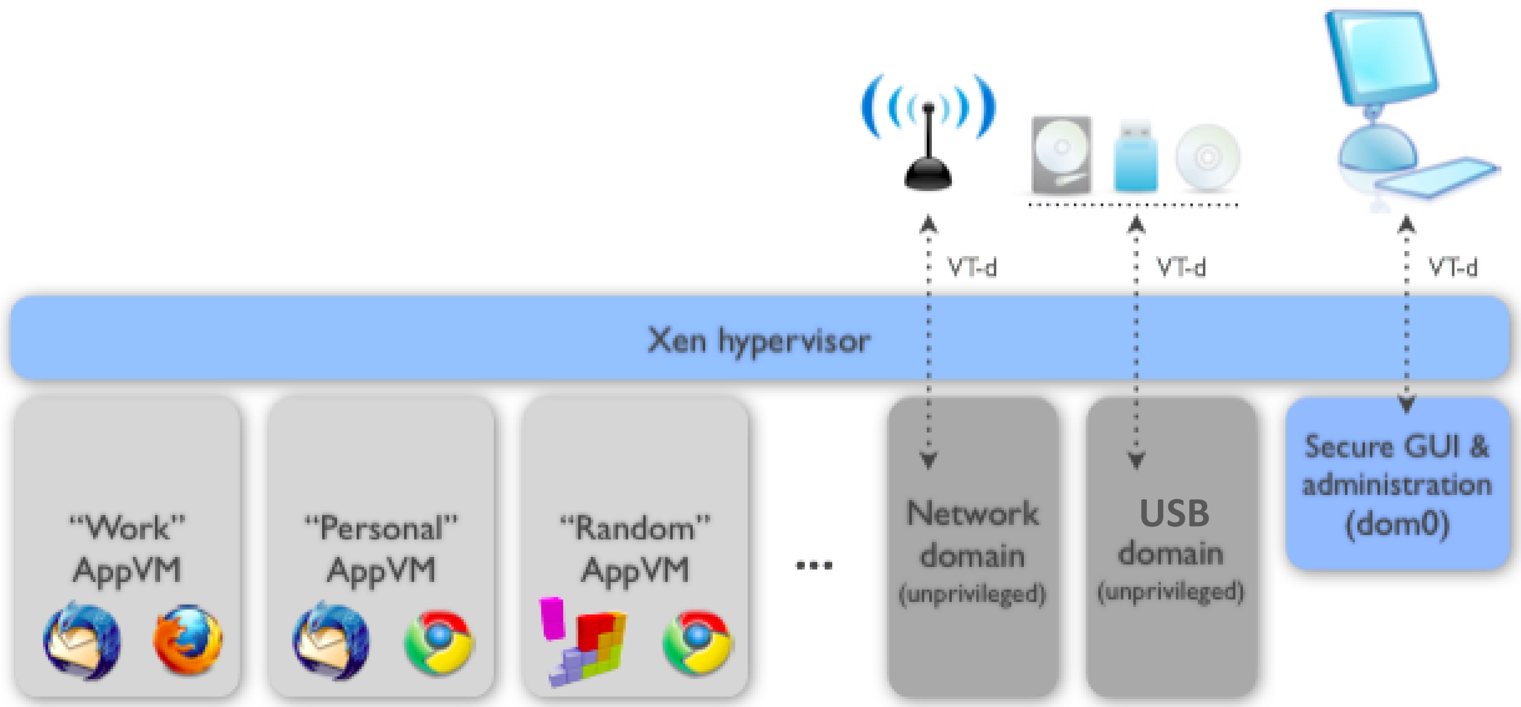
- Basics (Trusted Desktop, AppVMs, TemplateVMs)
- Networking (NetVMs, ProxyVMs, Firewalling, TorVM)
- Storage (Block devices handling, UsbVM)
- Disposable VMs (Unique features, customizations)
- Qubes Apps (qrexec basics, Split GPG, PDF convert)
- Windows AppVMs (installation, templates)

Part 2 (for Power Users & Devs)

- Qubes Inter-VM services (qrexec, policies)
- Hello World Qubes qrexec App
- Qubes Builder (Build your own Qubes, contribute patches, components)
- Porting Window Managers (e.g. Awsome)
- New templates (e.g. Debian-based)
- Quick look at Qubes R3 changes

Qubes OS Basics

Architecture


VT-d
VT-d
VT-d
Xen hypervisor
"Work" AppVM
"Personal" AppVM
"Random" AppVM
...
Network domain (unprivileged)
USB domain (unprivileged)
Secure GUI & administration (dom0)

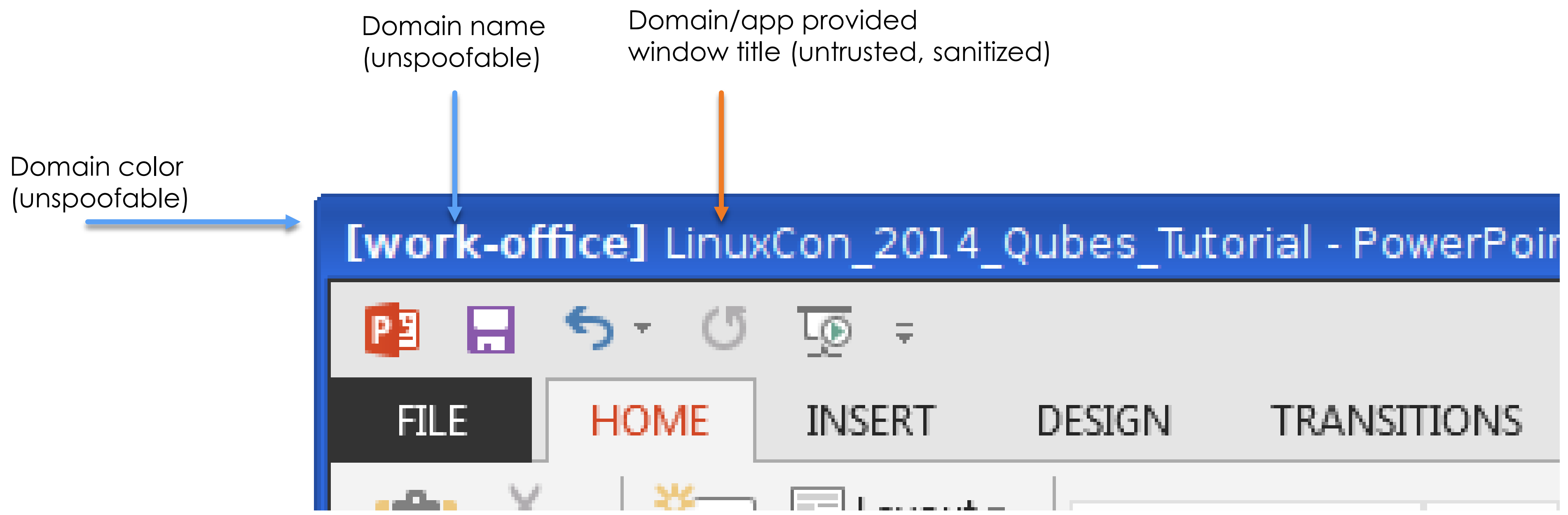
Qubes as multi-domain system

- Domains represent areas, e.g.
  - personal, work, banking
  - work-web, work-project-XYZ, work-accounting
  - personal-very-private, personal-health
- No 1-1 mapping between apps and VMs!
  - If anything, then user tasks-oriented sandboxing, not app-oriented
  - E.g. few benefits from sandboxing: The Web Browser, or The PDF Reader
- It's data we want protect, not apps/system!

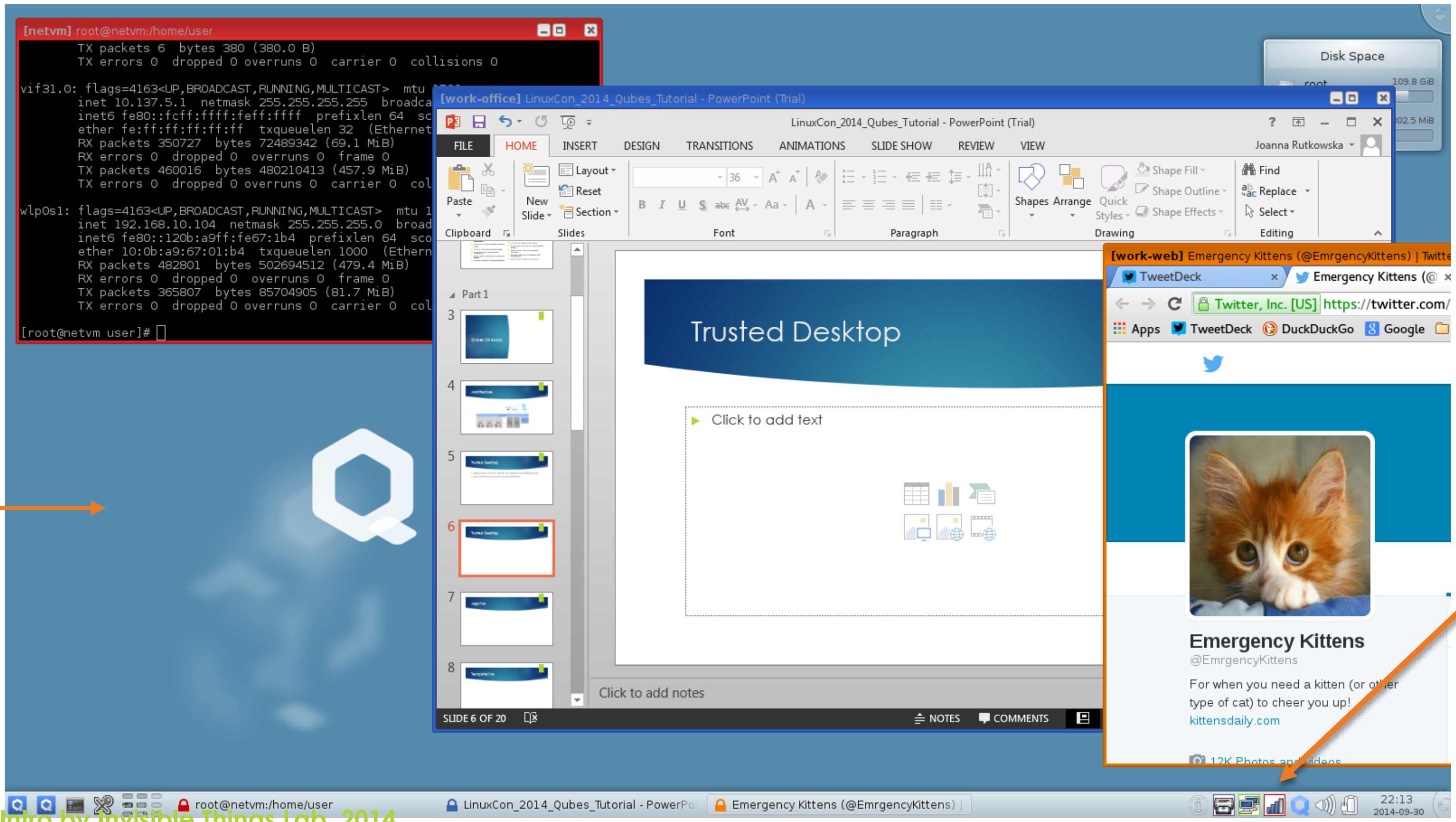
Trusted Desktop

- Apps windows “extracted” from VMs and composed onto **common desktop**
- Clear indications to which VM a given window belongs

Trusted Desktop Decorations



Trusted Desktop



```
[netvm] root@netvm:/home/user
        TX packets 6  bytes 380 (380.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vif31.0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
        inet 10.137.5.1  netmask 255.255.255.255  broadca
        inet6 fe80::fcff:ffff:feff:ffff  prefixlen 64  sc
        ether fe:ff:ff:ff:ff:ff  txqueuelen 32  (Ethernet
        RX packets 350727  bytes 72489342 (69.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 460016  bytes 480210413 (457.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  col

wlp0s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1
        inet 192.168.10.104  netmask 255.255.255.0  broad
        inet6 fe80::120b:a9ff:fe67:1b4  prefixlen 64  sco
        ether 10:0b:a9:67:01:b4  txqueuelen 1000  (Ethern
        RX packets 482801  bytes 502694512 (479.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 365807  bytes 85704905 (81.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  col

[root@netvm user]# 
```

[work-office] LinuxCon_2014_Qubes_Tutorial - PowerPoint (Trial)

Trusted Desktop

Click to add text

Click to add notes

SLIDE 6 OF 20

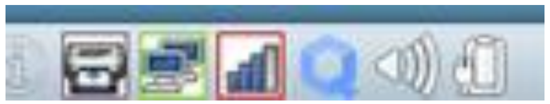
[work-web] Emergency Kittens (@EmrgencyKittens) | Twitte

Twitter, Inc. [US] https://twitter.com/

Emergency Kittens

@EmrgencyKittens

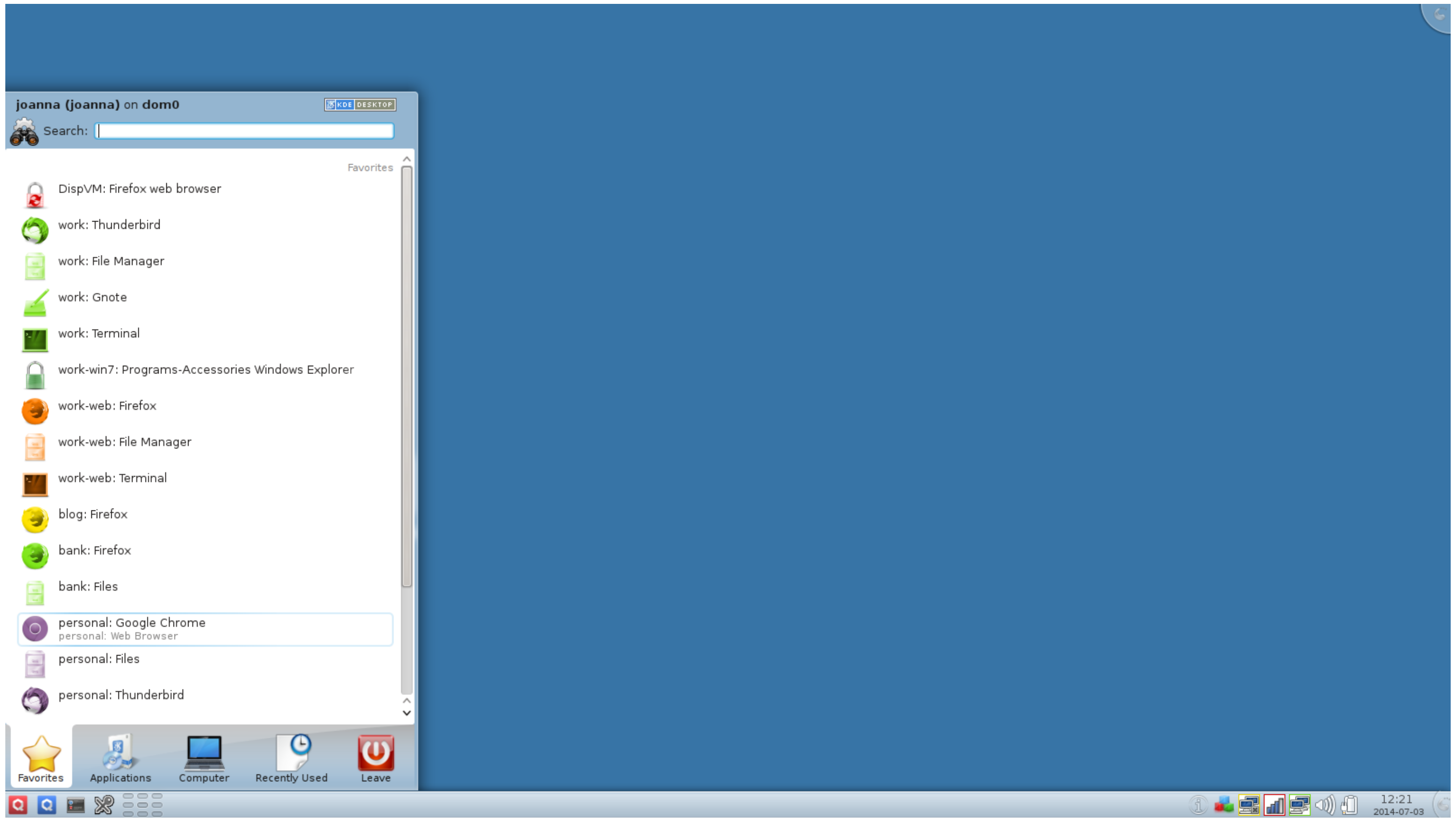
For when you need a kitten (or other type of cat) to cheer you up!
kittensdaily.com



Desktop (wallpaper) managed by Dom0 WM

"Start Menu" managed by Dom0 WM

Trusted Desktop (Apps launcher)



Secure clipboard

- Challenge: copy clipboard from VM “Alice” to VM “Bob”, don’t let VM “Mallory” to learn its content in the meantime
- Solved by introducing Qubes “global clipboard” to/from which copy/paste is explicitly controlled by the user (Ctrl-Shift-C, Ctrl-Shift-V)
- Requires 4 stages:
  - Ctrl-C (in the source VM)
  - Ctrl-Shift-C (tells Qubes: copy this VM buffer into global clipboard)
  - Ctrl-Shift-V (in the destination VM: tells Qubes: make global clipboard available to this VM)
  - Ctrl-V (in the destination VM)
- Ctrl-Shift-C/V cannot be injected by VMs (unspoofable key combo).
- In practice almost as fast as traditional 2-stage copy-paste (don’t freak out! ;)

Types of VMs in Qubes

According to role

- AppVMs (user apps and files run here)
- ServiceVMs (mostly invisible to the user)
  - NetVMs
  - ProxyVMs (e.g. FirewallVM, TorVM, VPN)
  - Dom0 (admin domain)
  - GUI domain (in R3)
- Templates

According to implementation

- PV (default) ^ HVM (e.g. Windows)
- Template-based ^ Standalone
- Persistent ^ Disposable

AppVMs

- Linux-based Para Virtualized
  - Most Desktop Environment stripped off, custom startup.
  - Quick boot, small memory footprint
- Based on a template (Fedora 20 default, but Debian & Arch also avail.)
  - This means rootfs is *non persistent* by default!
  - Separate volume (virtual disk) for user home (/rw)
- Disposable VMs
  - Like AppVMs, but without private volume (non persistent home dir)
  - Optimized to boot up even faster (restored from snapshot instead of boot)

TemplateVMs

- Started only for software upgrade/installation or global config mods
- By default limited networking only to apt/yum updates proxy
- Trusted – a compromised template can compromise all “children”
- Non-persistence of rootfs
  - as reliability feature
  - as security feature

Rootfs non-persistence as security feature?

- AppVM's rootfs gets automatically reverted back to "golden image" on each restart...
  - No malware persistence on root fs!
- ... but malware can still place its triggers in /home (generally /rw):
  - .bashrc
  - Thunderbird/Firefox/etc profile directory (e.g. subvert plugins)
  - Malicious PDF/DOC/etc (exploiting hypothetical bug in default handler app)
  - Malicious fs meta (exploiting hypothetical bug in kernel fs module)

Rootfs non-persistence as security feature? (cont.)

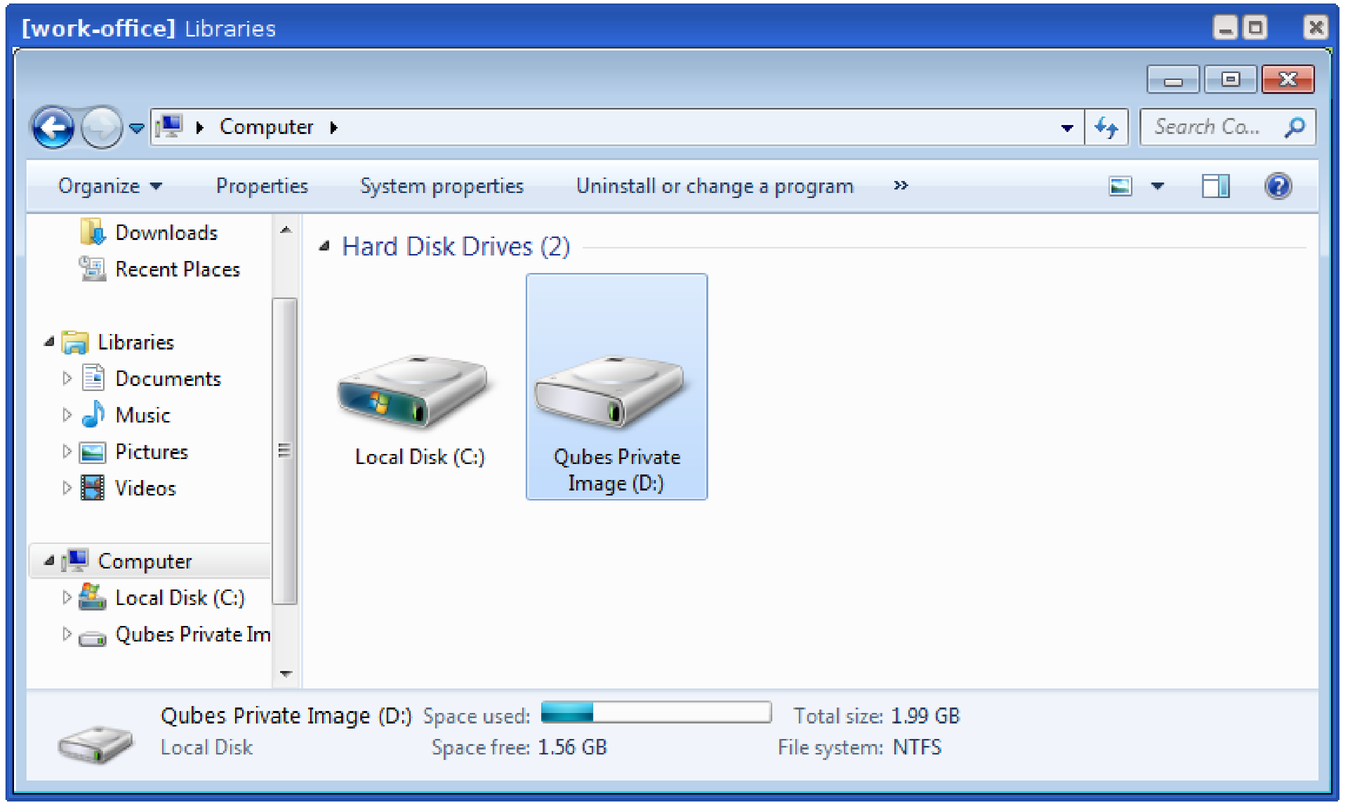
- … still has some unique security advantages though:
  - Malware inactive before /rw mounted/parsed, offers chances to scan reliably
    - Yet problem for malware scanning generally hard in general
    - But might be easier for limited scenarios (e.g. easy for .bashrc, difficult for TB profile)
  - Malware triggers via malicious docs or malformed fs will *automatically* stop working after template patched
    - Note how this malware in AppVMs cannot interfere with reliability of template patching

Where are the VM files?

- `/var/lib/qubes`
  - `appvms/`
  - `servicevms/`
  - `vm-templates/`
  - `vm-kernels/`

- `appvms/my-appvm/`
  - **`private.img`**
  - `volatile.img`
  - `my-appvm.conf (autogen!)`
- `vm-templates/fedora-20-x64/`
  - **`root.img`**
  - `root-cow.img`
  - `private.img   (template's home)`
  - `volatile.img`
  - `fedora-20-x64.conf (autogen!)`

HVM AppVMs (e.g. Windows-based)



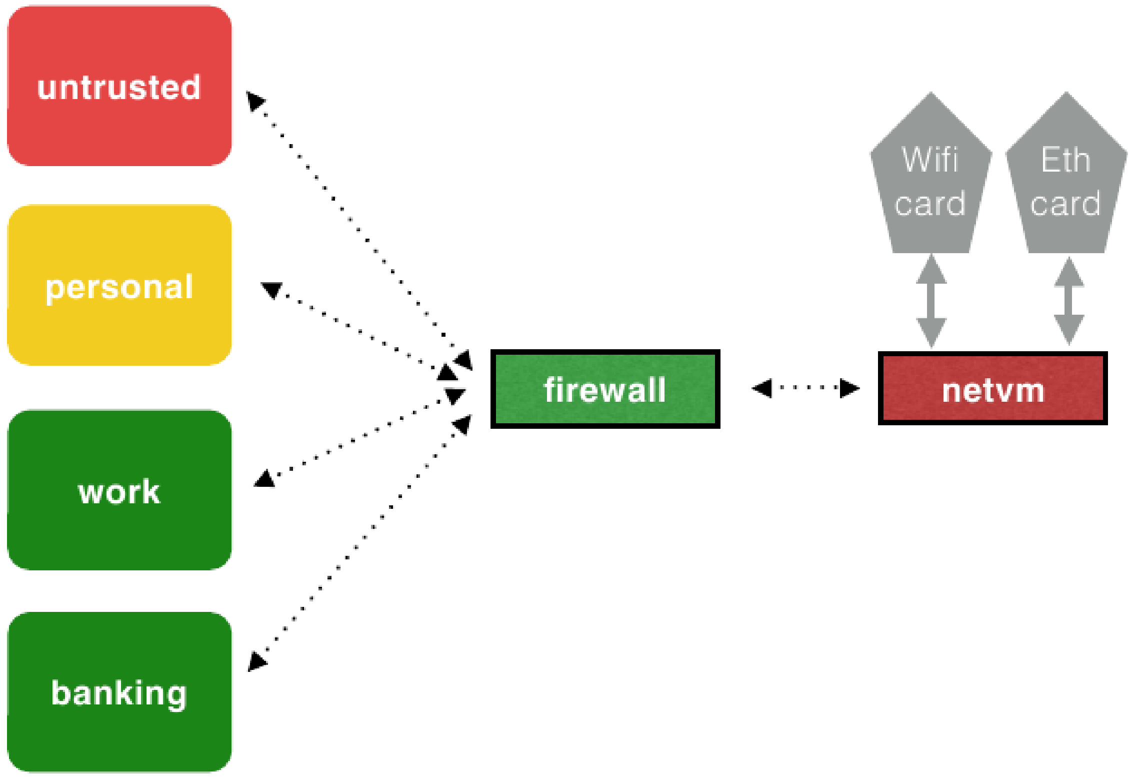
Qubes Private Image (D:) Space used: Total size: 1.99 GB
Local Disk Space free: 1.56 GB File system: NTFS

AppVMs configuration

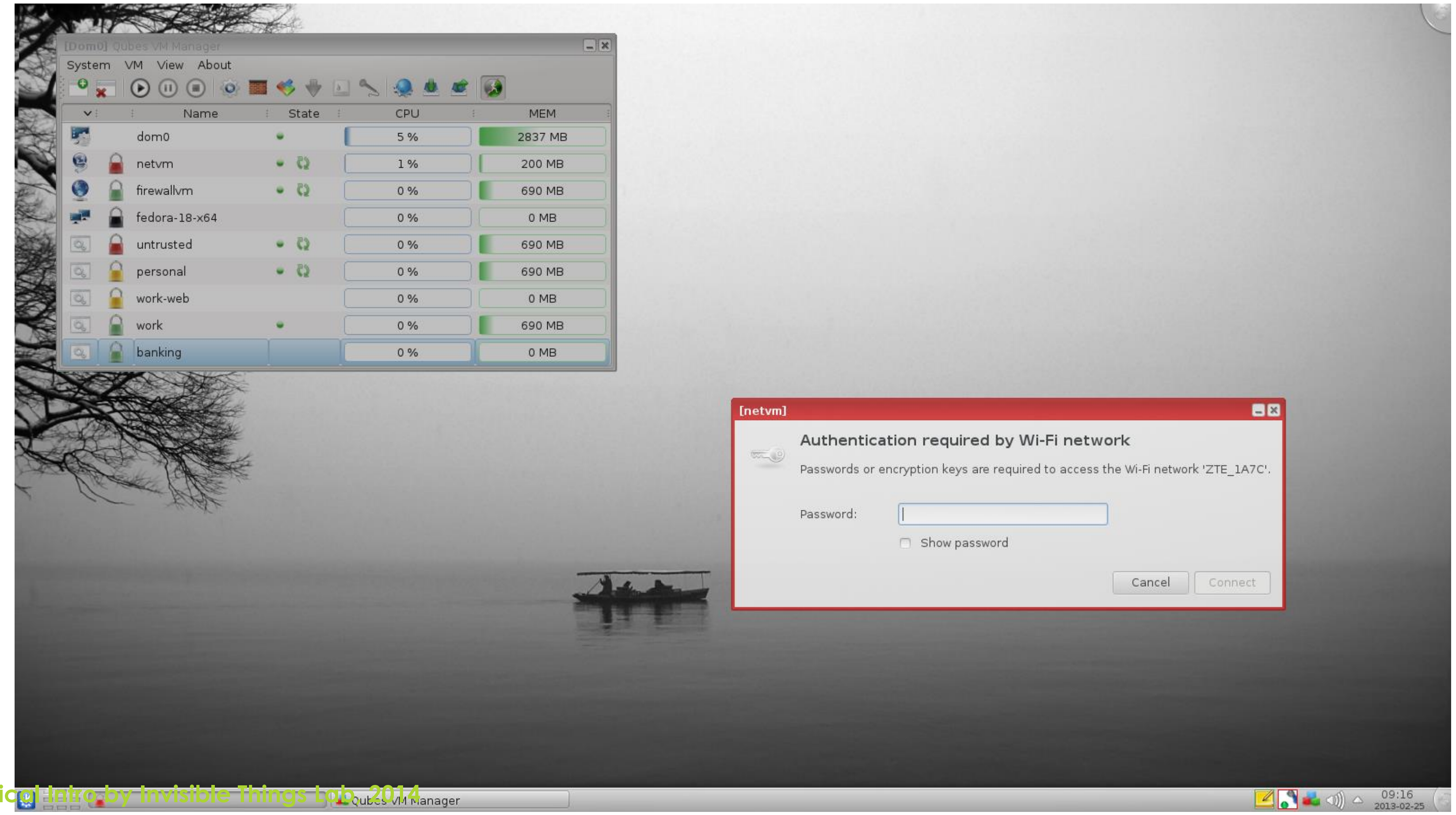
- /rw/config
  - /rw/config/rc.local
  - /rw/config/qubes-firewall-user-script
  - https://wiki.qubes-os.org/wiki/UserDoc/ConfigFiles
- qvm-service
  - Tells VM's scripts which (systemd) services should/shouldn't be started
  - Note: qvm-service will not warn you about service name spelling errors
  - https://wiki.qubes-os.org/wiki/Dom0Tools/QvmService

Networking in Qubes OS

Default networking topology



The whole networking stacks is sandboxed…



[Dom0] Qubes VM Manager

System VM View About

Name	State	CPU	MEM
dom0	●	5 %	2837 MB
netvm	●	1 %	200 MB
firewallvm	●	0 %	690 MB
fedora-18-x64		0 %	0 MB
untrusted	●	0 %	690 MB
personal	●	0 %	690 MB
work-web		0 %	0 MB
work	●	0 %	690 MB
banking		0 %	0 MB

[netvm]

Authentication required by Wi-Fi network

Passwords or encryption keys are required to access the Wi-Fi network 'ZTE_1A7C'.

Password:

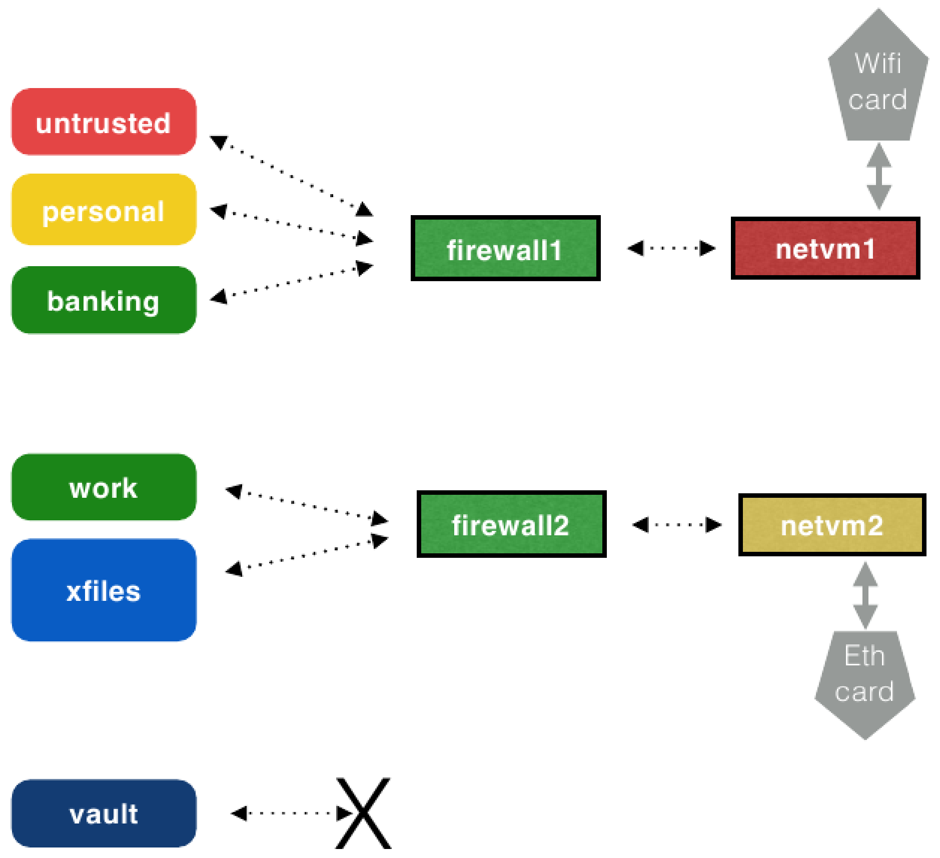
Show password

Cancel Connect

Type of VMs (networking-wise)

- NetVMs
  - Have NICs or USB modems assigned via PCI-passthrough
  - Provide networking to other VMs (run Xen **Net Backends**)
- AppVMs
  - Have no physical networking devices assigned
  - Consume networking provided by other VMs (run Xen **Net Frontends**)
  - Some AppVMs might not use networking (i.e. be network-disconnected)
- ProxyVMs
  - Behave as AppVMs to other NetVMs (or ProxyVMs), i.e. consume networking
  - Behave as NetVMs to other AppVMs (or ProxyVMs), i.e. provide networking
  - Functions: firewalling, VPN, Tor'ing, monitoring, proxying, etc.
- Dom0
  - has no network interfaces!

Example of more complex networking configuration...



FirewallVM: special role of *any* ProxyVM

- Any proxy VM becomes firewall VM for the AppVMs (or other ProxyVMs) directly connected to it
- Scripts running in each of the ProxyVM look at the global firewalling config provided by Dom0 (via XenStore) and use it to configure iptables rules for its direct children
- The role of the FirewallVM is *not* to prevent data leaks!
  - Sadly too many cooperative covert channels for this to be meaningful
  - They are to prevent user mistakes, config mistakes, and accidental leaks only

Networking config inside VMs

- Interfaces
- Firewall
- NAT

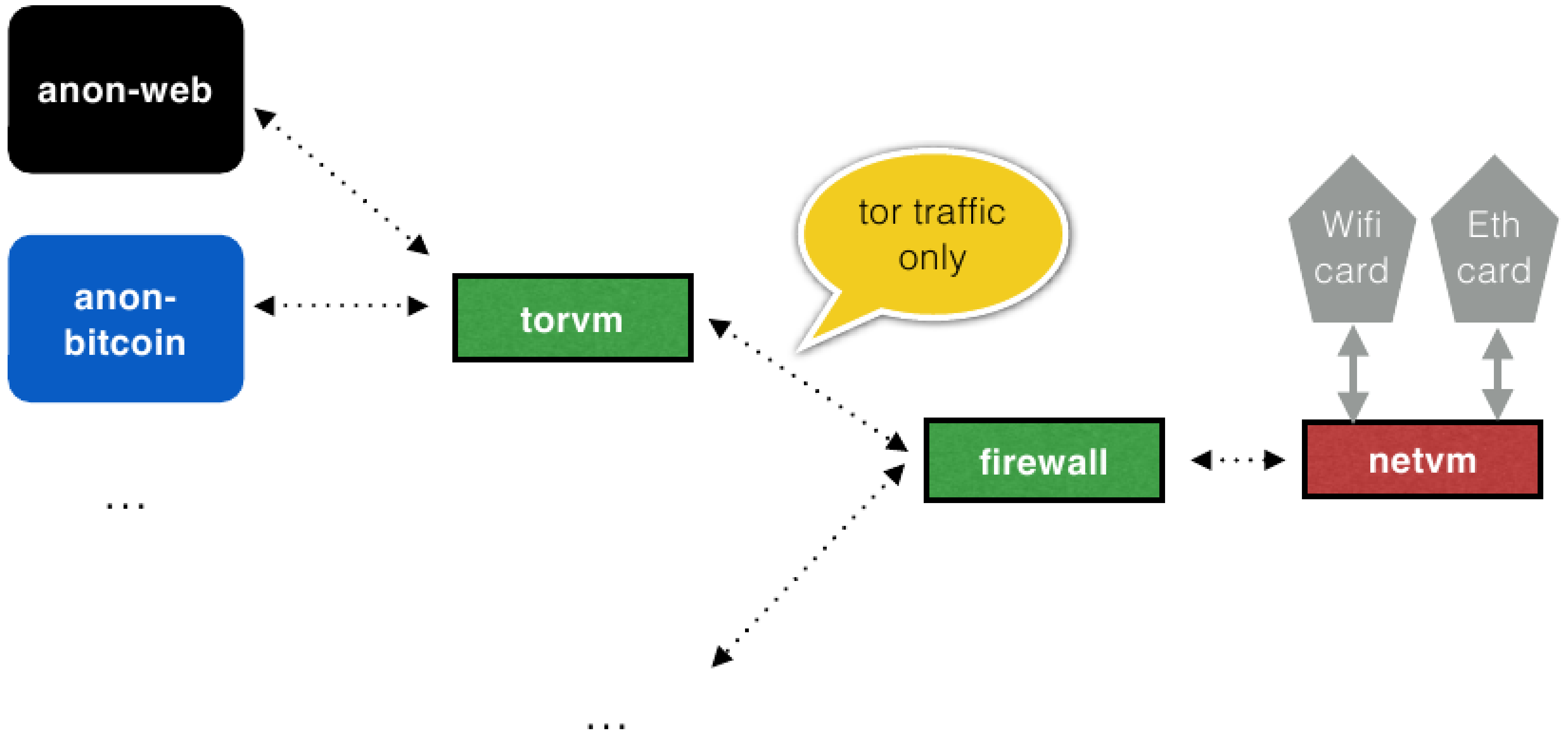
Customizing networking routings

- Allow networking between two AppVMs
- Allow port forwarding to an AppVM from outside world
- See:
  - https://wiki.qubes-os.org/wiki/QubesFirewall

Qubes TorVM

- Easy setup
  - qvm-create -p torvm
    - qvm-service torvm -d qubes-netwatcher
    - qvm-service torvm -d qubes-firewall
    - qvm-service torvm -e qubes-tor
  - In TorVM (or its template):
    - sudo yum install qubes-tor-repo
    - sudo yum install qubes-tor
  - Configure AppVMs to use it as proxy:
    - qvm-pres –s myanonvm netvm torvm

TorVM configuration example



Digression on using TorVM in Qubes OS

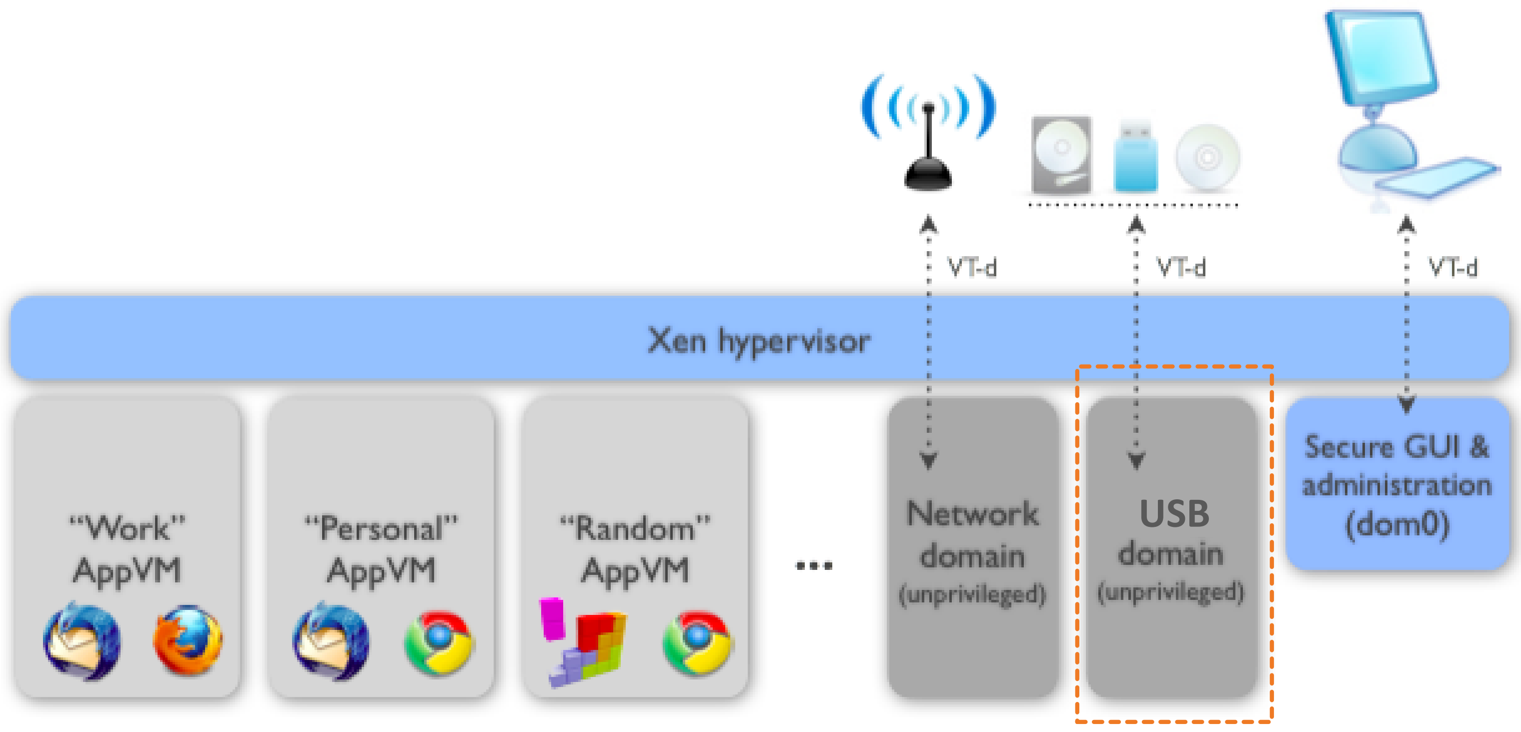
- TorVM cannot (obviously) anonymize anything beyond IP/MAC
  - Use e.g. TBB or Whonix workstation in/as clients of TorVM
- DispVMs as TorVM clients
  - Set DispVM's netvm to none, manually change to torvm (or set torvm for all DispVMs)
  - Note the volatile.img is backed to disk! (no anti-forensics yet)
- Potential leaks through:
  - qvm-open-in-dvm ...
  - Set default netvm to none, manually change to torvm (or set torvm for all DispVMs)
  - adjust qrexec policy to prevent that (qubes.OpenInVM, qubes.VMShell)

Further reading

- http://theinvisiblethings.blogspot.com/2011/09/playing-with-qubes-networking-for-fun.html
- https://wiki.qubes-os.org/wiki/UserDoc

Storage, Block devices and USB

Architecture

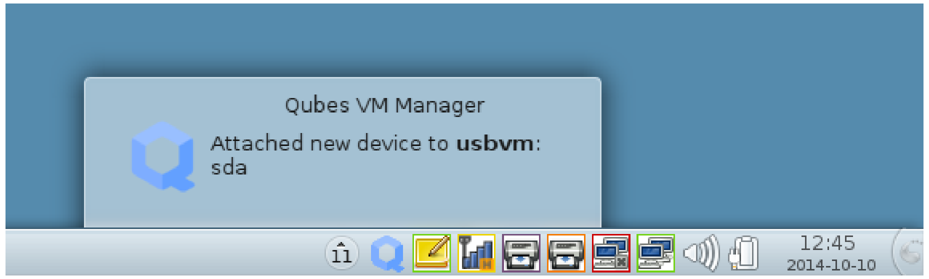

VT-d
VT-d
VT-d
Xen hypervisor
"Work" AppVM
"Personal" AppVM
"Random" AppVM
...
Network domain (unprivileged)
USB domain (unprivileged)
Secure GUI & administration (dom0)

Storage backend in Qubes

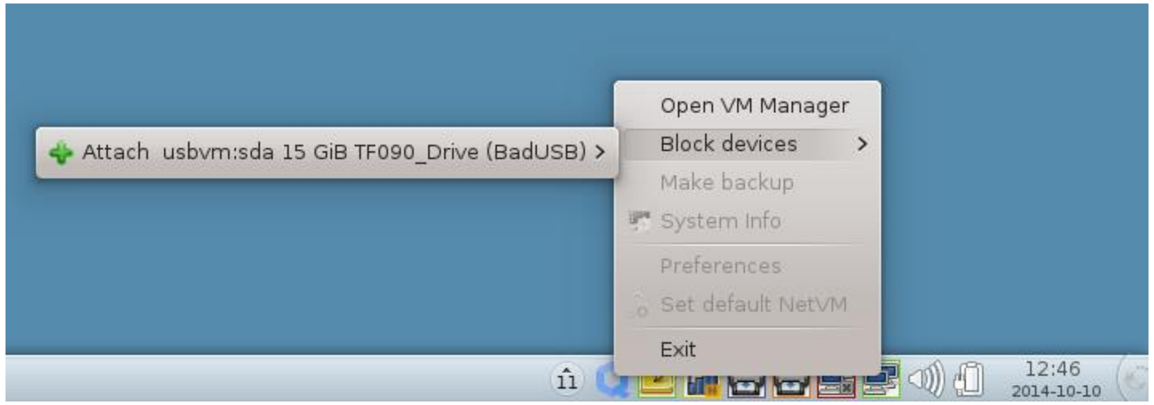
- Dom0 hosts block backend for all the VMs
  - The simplest possible block backend (Xen blkback), no qcow, no qemu!
- Also possible to use backends located in (untrusted) VMs
  - E.g. export block devices from UsbVM
  - qvm-block
  - https://wiki.qubes-os.org/wiki/StickMounting

Examples of USBVM usage

- USB storage device attach example


Qubes VM Manager

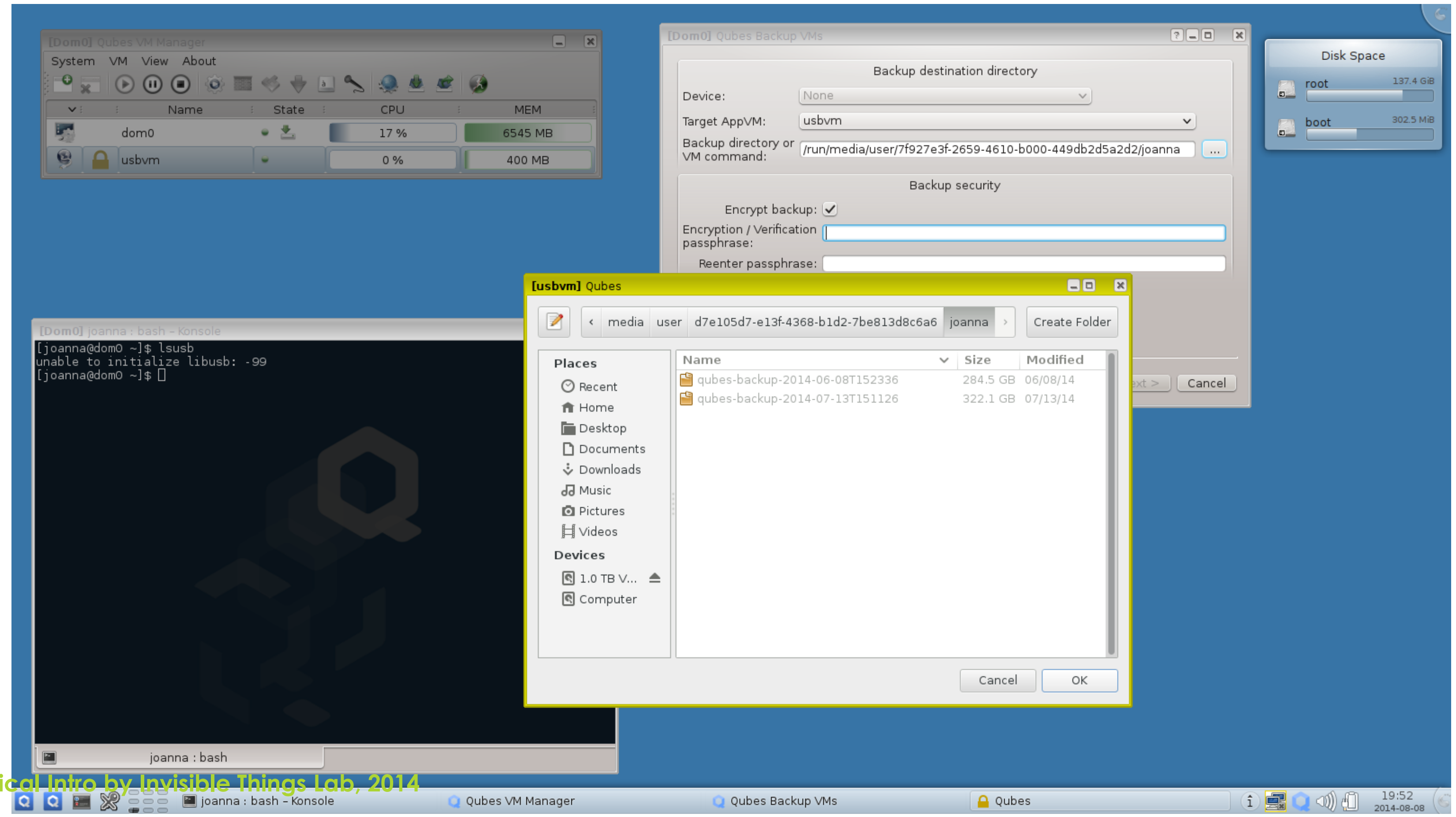
Attached new device to **usbvm**: sda



Keeping USBVM untrusted

- Backups to USBVM
- LUKS on /dev/xvdi exported from USBVM

Backing up to untrusted USB



```
[joanna@dom0 ~]$ lsusb
unable to initialize libusb: -99
[joanna@dom0 ~]$ 
```

Name	State	CPU	MEM
dom0		17 %	6545 MB
usbvm		0 %	400 MB

[Dom0] Qubes Backup VMs

Backup destination directory

Device: None

Target AppVM: usbvm

Backup directory or VM command: /run/media/user/7f927e3f-2659-4610-b000-449db2d5a2d2/joanna

Backup security

Encrypt backup: ✓

Encryption / Verification passphrase:

Reenter passphrase:

[usbvm] Qubes

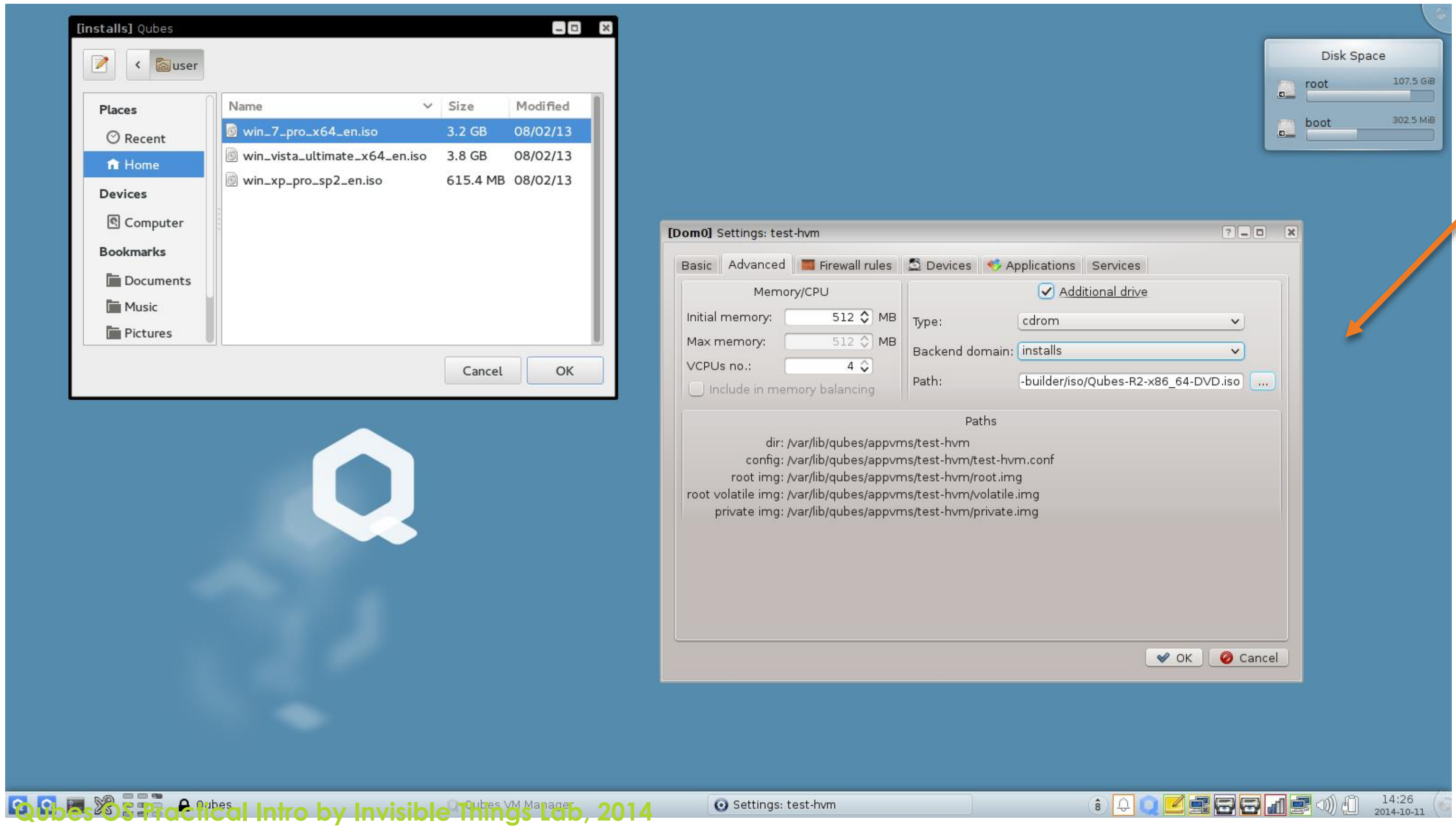
media › user › d7e105d7-e13f-4368-b1d2-7be813d8c6a6 › joanna

Name	Size	Modified
qubes-backup-2014-06-08T152336	284.5 GB	06/08/14
qubes-backup-2014-07-13T151126	322.1 GB	07/13/14

Disk Space

root	137.4 GiB	
boot	302.5 MiB	

Booting HVMs from images



[installs] Qubes

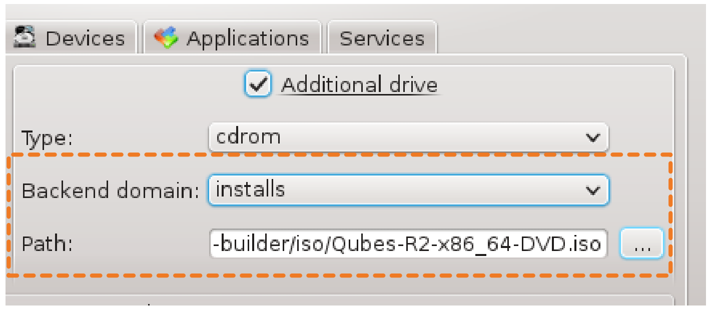
Name	Size	Modified
win_7_pro_x64_en.iso	3.2 GB	08/02/13
win_vista_ultimate_x64_en.iso	3.8 GB	08/02/13
win_xp_pro_sp2_en.iso	615.4 MB	08/02/13

[Dom0] Settings: test-hvm

Memory/CPU
Initial memory: 512 MB
Max memory: 512 MB
VCPUs no.: 4
Include in memory balancing

✓ Additional drive
Type: cdrom
Backend domain: installs
Path: -builder/iso/Qubes-R2-x86_64-DVD.iso

Paths
dir: /var/lib/qubes/appvms/test-hvm
config: /var/lib/qubes/appvms/test-hvm/test-hvm.conf
root img: /var/lib/qubes/appvms/test-hvm/root.img
root volatile img: /var/lib/qubes/appvms/test-hvm/volatile.img
private img: /var/lib/qubes/appvms/test-hvm/private.img



✓ Additional drive
Type: cdrom
Backend domain: installs
Path: -builder/iso/Qubes-R2-x86_64-DVD.iso

qvm-usb (EXPERIMENTAL)

- qvm-usb != qvm-block
  - qvm-block attaches *block* devices (e.g. USB mass storage device) to VMs
  - qvm-usb (is supposed to) attaches USB devices (e.g. camera) to VMs
  - qvm-block works, qvm-usb (mostly) doesn’t (currently)
- Current problems with qvm-usb
  - Xen PVUSB backend is immature, devel seem abandoned
  - We consider switching to USB-over-IP from Linux kernel, replace IP for qrexec

Dom0 storage

- /boot
  - Not encrypted, not integrity protected (cannot be!)
  - Use Anti Evil Maid
- Dom0 root filesystem
  - LUKS
  - No integrity protection!
    - Malleability attacks possible
    - Difficult to do integrity protection on block-level
  - No anti-forensics (yet)
    - DispVM volatile.img backing to Dom0 fs

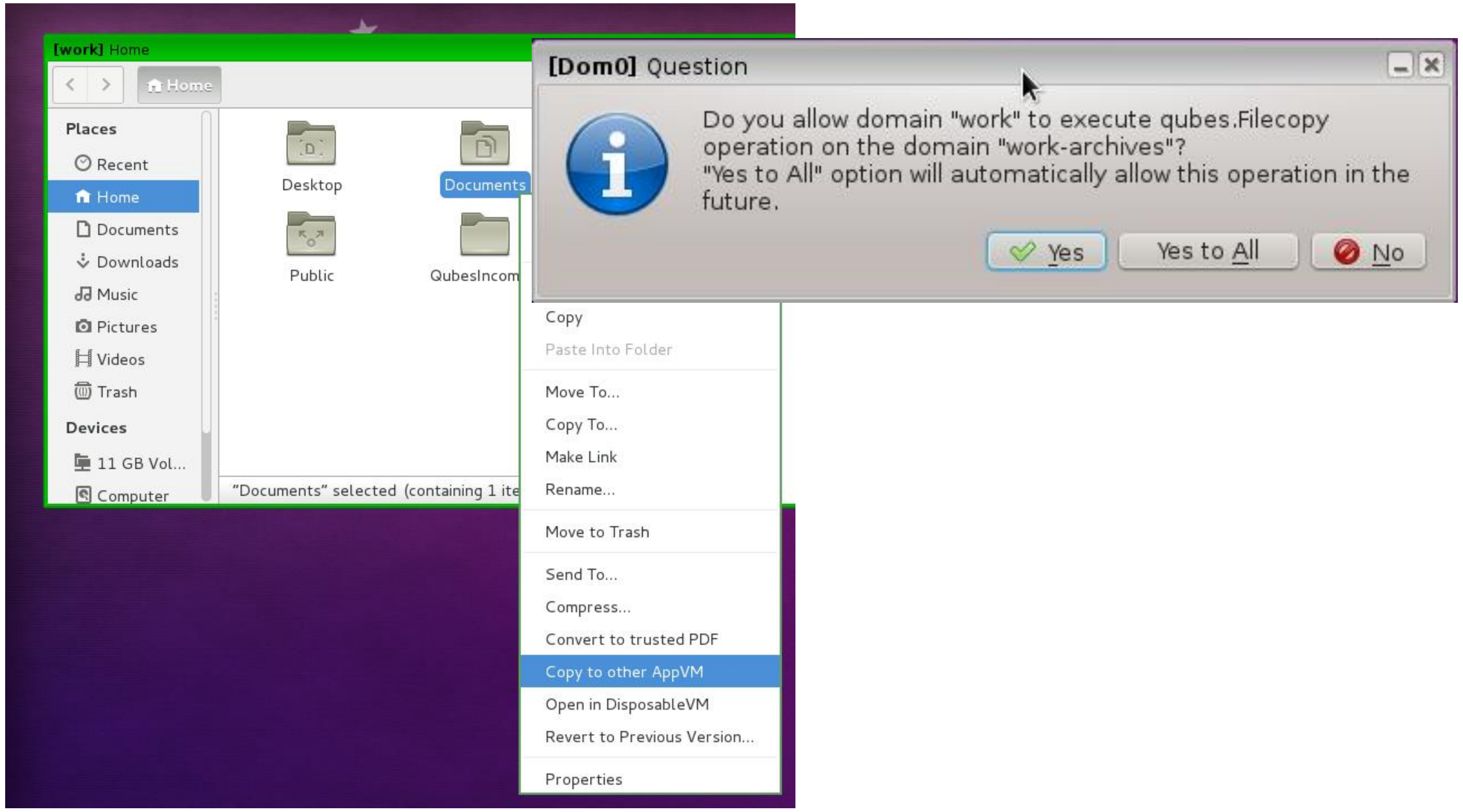
Qubes Apps

Qubes Apps Examples

- InterVM file copy
- Disposable VMs, qvm-open-in-(d)vm
- Thunderbird Open-in-DispVM extension
- PDF converter
- Split GPG
  - Comparison vs. Smartcard (need for USB controller passthrough)
  - Key import

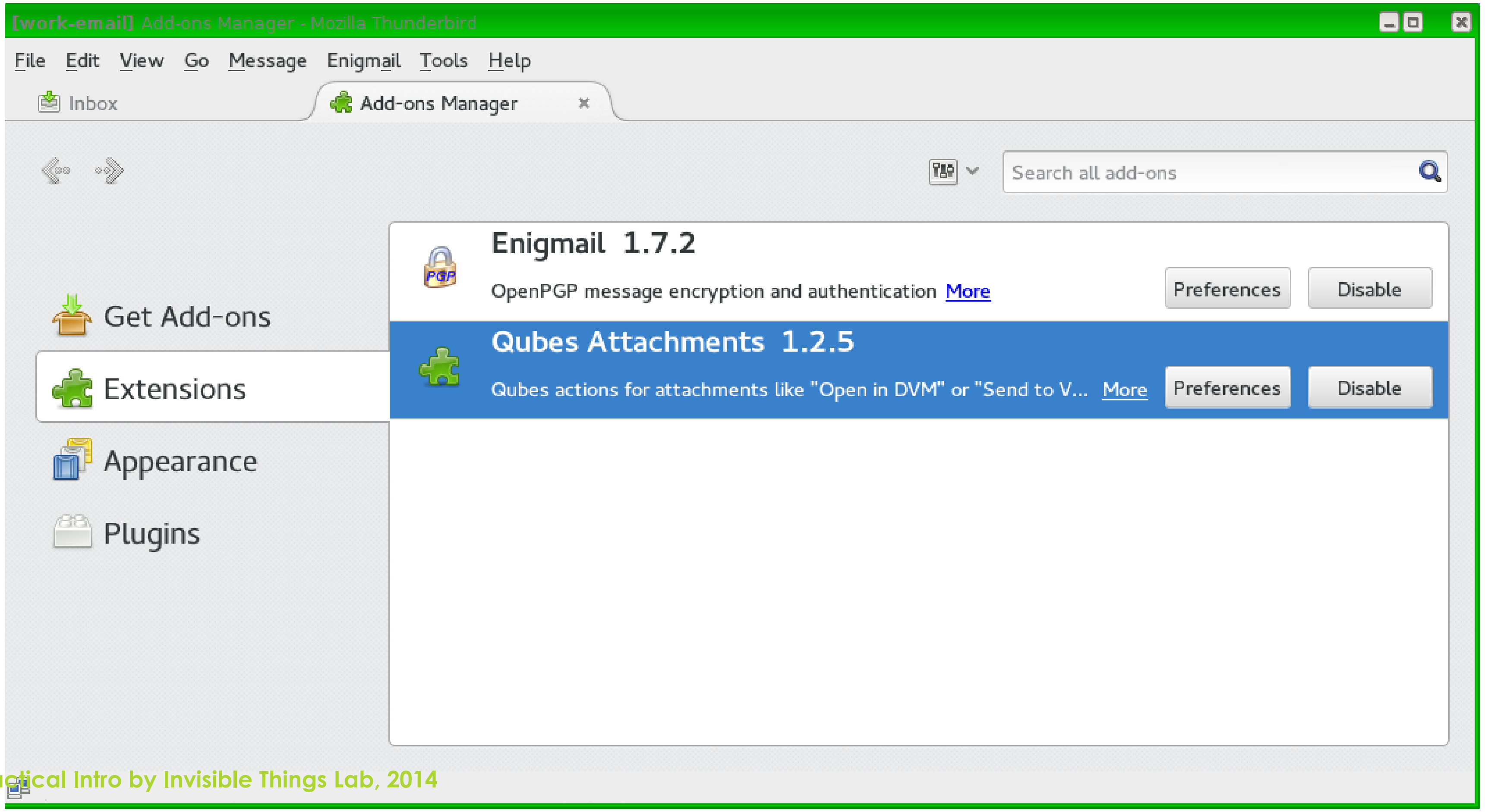
Trivial: Inter-VM filecopy

- Security challenges
  - Virtual pendrive: no good
  - Samba, NFS: no good
- qubes.FileCopy
  - minimal cpio-like format
  - qvm-copy-to-vm


[Dom0] Question

Do you allow domain "work" to execute qubes.Filecopy operation on the domain "work-archives"?
"Yes to All" option will automatically allow this operation in the future.

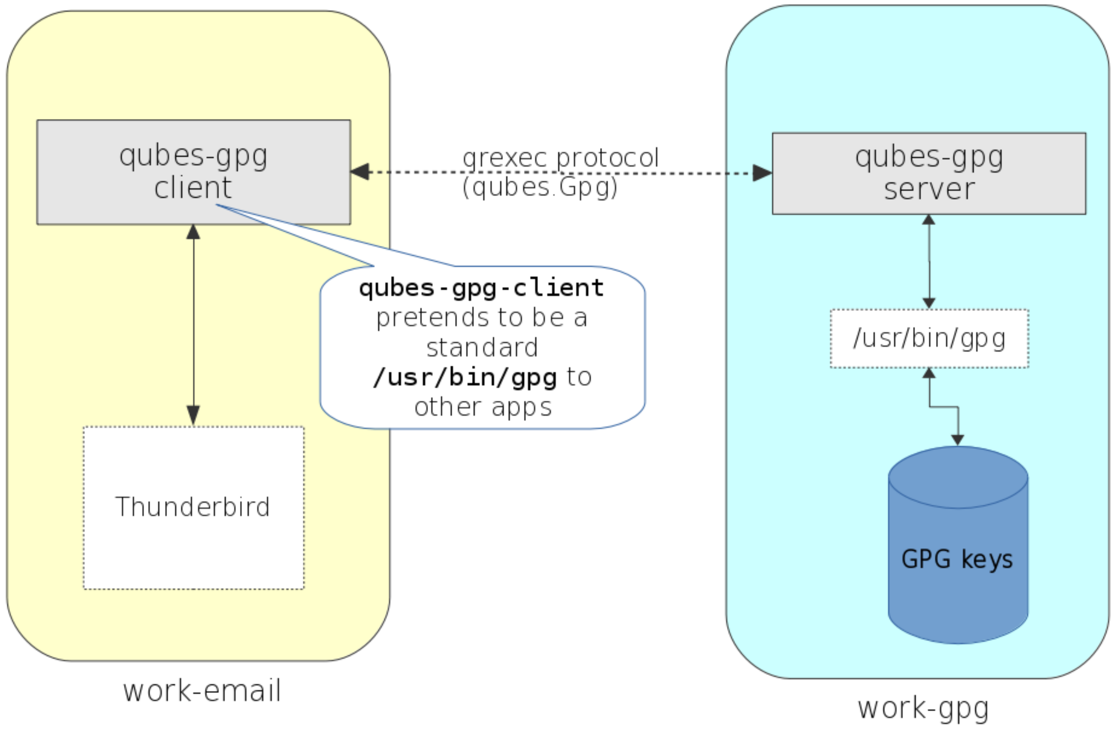
Thunderbird extension



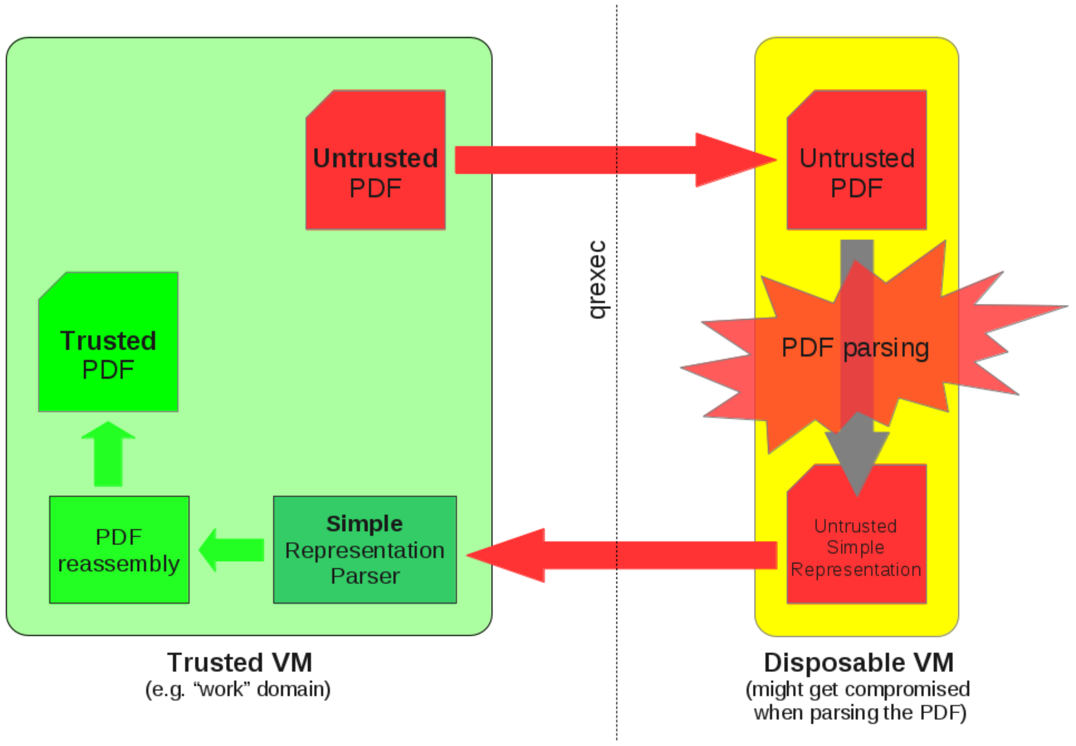
Enigmail 1.7.2
OpenPGP message encryption and authentication More

Qubes Attachments 1.2.5
Qubes actions for attachments like "Open in DVM" or "Send to V... More

Qubes Split GPG


qubes-gpg client
grexec protocol (qubes.Gpg)
qubes-gpg server
qubes-gpg-client pretends to be a standard **/usr/bin/gpg** to other apps
Thunderbird
/usr/bin/gpg
GPG keys
work-email
work-gpg

Qubes PDF Converter



qrexec policy

- Central place to allow/deny inter-VM services
- E.g. disallow “personal” to request qubes.FileCopy from “work”
- E.g. disallow any VM to copy clipboard to “vault”
- E.g. allow any VM to open files in a fresh DispVM without asking
- See /etc/qubes-rpc/policy (also later slides in Part 2)

Qubes OS: Part II For Power Users & Devs

Qubes Inter-VM Services: **qrexec**

Inter-VM interactions

- VMs, to be useful for anything, must be able to communicate with each other, and/or with devices
- Device virtualization (Networking, Storage, Other devices)
- XenStore (Xen system-wide registry)
- Qubes GUI virtualization (GUI protocol)
- **Qubes RPC: qrexec**

Inter-VM comm mechanisms in Xen

Level 1: Shared Memory

- **Grant tables** hypercall (setting up shared pages between VMs)
- **Event Channel** hypercall (inter-VM signaling, kind of like UNIX signals)
- All Xen frontends/backends communicate using these mechanisms
  - Xen hypervisor doesn't enforce/look into this communication
  - Xen provides convenient macros (via .h) that might be used for a primitive frontend-backend protocols (so called Ring Buffers)

Inter-VM comm mechanisms in Xen:
Level 2: vchan

- **vchan** is a usermode library
  - Written in early days by Qubes project
  - Merged into Xen starting from 4.2+ (much improved over "ours")
  - Qubes R2 still based on Xen 4.1, using "our old vchan", R3 uses Xen libvchan.
- Builds on top of **Grant Tables** and **Event Channels** (see the previous slide)
- Exposes socket-like "pipe" connecting two VMs
  - Qubes R2 limitation: one of the VMs is assumed to be Dom0
  - Qubes R3 (Xen 4.2+): between any two VMs

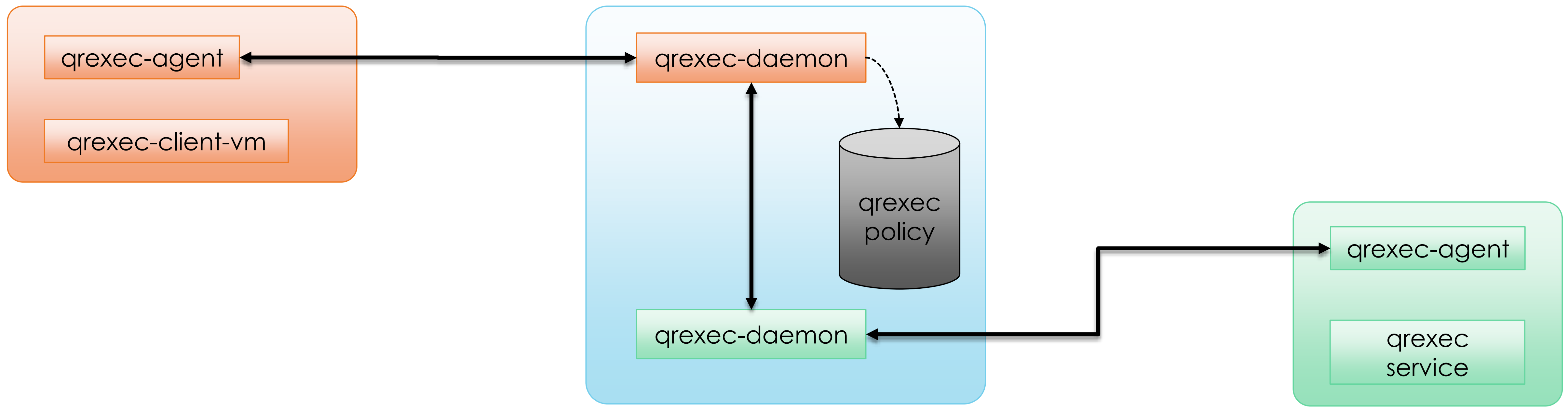
Inter-VM comm mechanism in Qubes:
Level 3: qrexec (only in Qubes OS currently)

- Qubes **qrexec** is:
  - Inter-VM protocol that runs over **vchan**
  - A set of tools (qrexec-agent, qrexec-client{-vm}, qrexec-daemon)
  - A policy framework
- qrexec is a framework for running programs in other VMs with their stdin/stdout piped together, all controlled by a central policy
- Each VM defines a set of services it can handle and by which programs:
  - /etc/qubes-rpc/qubes.XYZ files

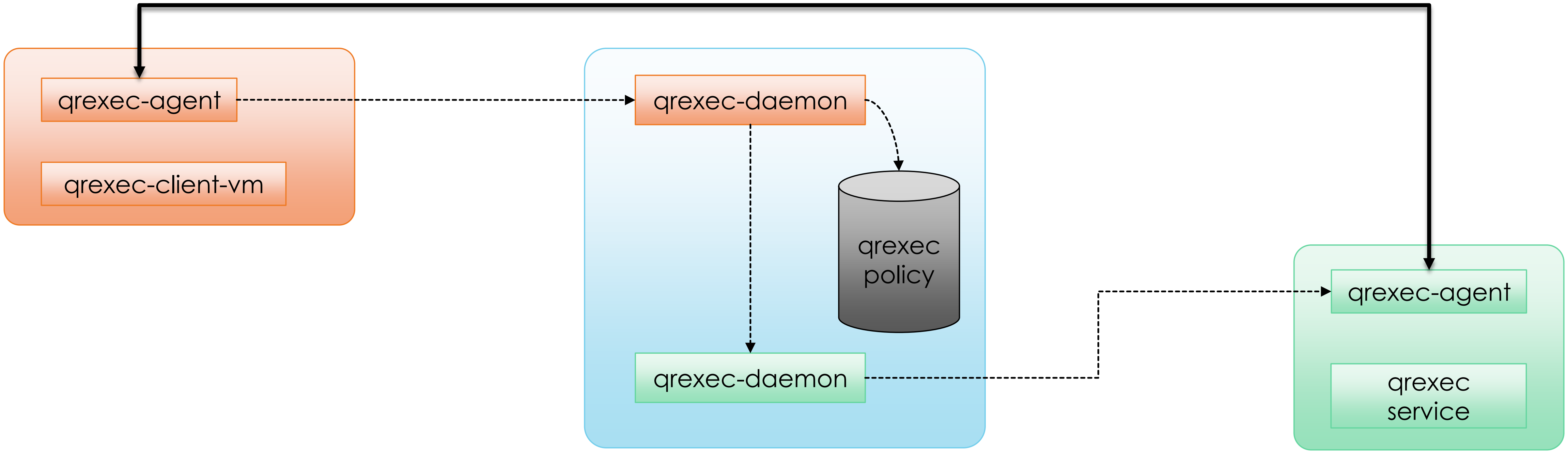
qrexec policy

- Example: prevent clipboard copy
- /etc/qubes-rpc/policy/
- Example services:
  - qubes.ClipboardPaste
  - qubes.FileCopy
- Example policies:
  - $anyvm vault deny
  - personal work deny
  - $anyvm $anyvm ask

qrexec implementation (Qubes R2)



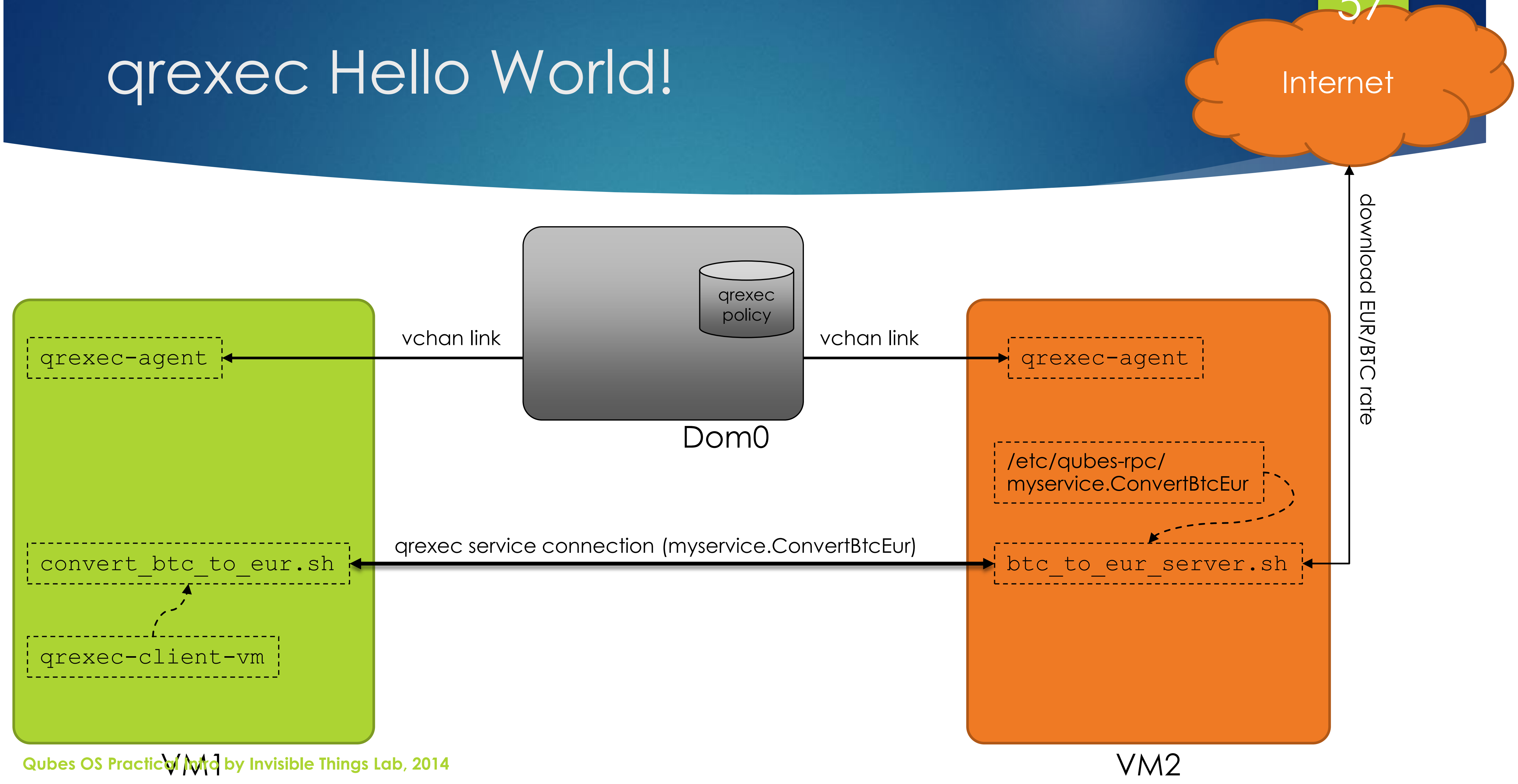
qrexec implementation (Qubes R3)



qrexec: Hello World!

- Client program (runs in VM1):
  - convert_btc_to_eur.sh <btc amount>
  - Let's assume client VM has no networking
- Server program (runs in VM2)
  - btc_to_eur_server.sh
- Service name: myservice.ConvertBtcEur
  - /etc/qubes-rpc/myservice.ConverterBtcEur in VM2
  - /etc/qubes-rpc/policy/myservice.BtcConverter in Dom0

qrexec Hello World!



qrexec: further reading

- https://wiki.qubes-os.org/wiki/Qrexec
- http://theinvisiblethings.blogspot.com/2013/02/converting-untrusted-pdfs-into-trusted.html
- https://wiki.qubes-os.org/wiki/Qrexec2Implementation
- https://wiki.qubes-os.org/wiki/Qrexec3Implementation

Qubes Builder

Why?

- Fetches all Qubes git repos, downloads all the additional source code
  - Additional code e.g.: linux-3.12.23.tar.xz, xen-4.1.6.1.tar.gz, zlib-1.2.3.tar.gz
- **Verifies digital signatures** on *all* downloaded code, *before* it gets used!
  - BTW, special patch for Xen Makefile not to wget dozens of components over http... ;)
- Allows to specify which public keys are trusted for which components
  - E.g. we don't want KDE devel signs to approve our GIT commits!
  - Actually KDE doesn't use any keys, but that's another story ;)
- Useful targets: get-sources, prepare-merge, etc. All check for known digital signature on the *latest* commit (HEAD)

qubes-builder anatomy

- One big Makefile, really :)
  - Plus some helper scripts
- chroot()-based
  - chroot not considered a security container

qubes-builder usage

- builder.conf
  - Which components (git repos)
  - Where repos are to be fetched from, which branches, etc
  - What distros to build for

- makefile targets
  - get-sources
  - prepare-merae GIT_SUBDIR=marmarek
  - qubes
  - sign-all
  - update-repo-current{-testing}

Building Qubes from scratch

- git clone qubes-builder.git
- gpg –import …
- git describe && git tag –v
- cp builder.conf.default builder.conf
- [vim builder.conf]
- make get-sources qubes [sign-all] iso

Building selected components

- make core-admin qubes-app-pdf-converter
- make COMPONENETS="core-admin qubes-app-pdf-converter" qubes

Future

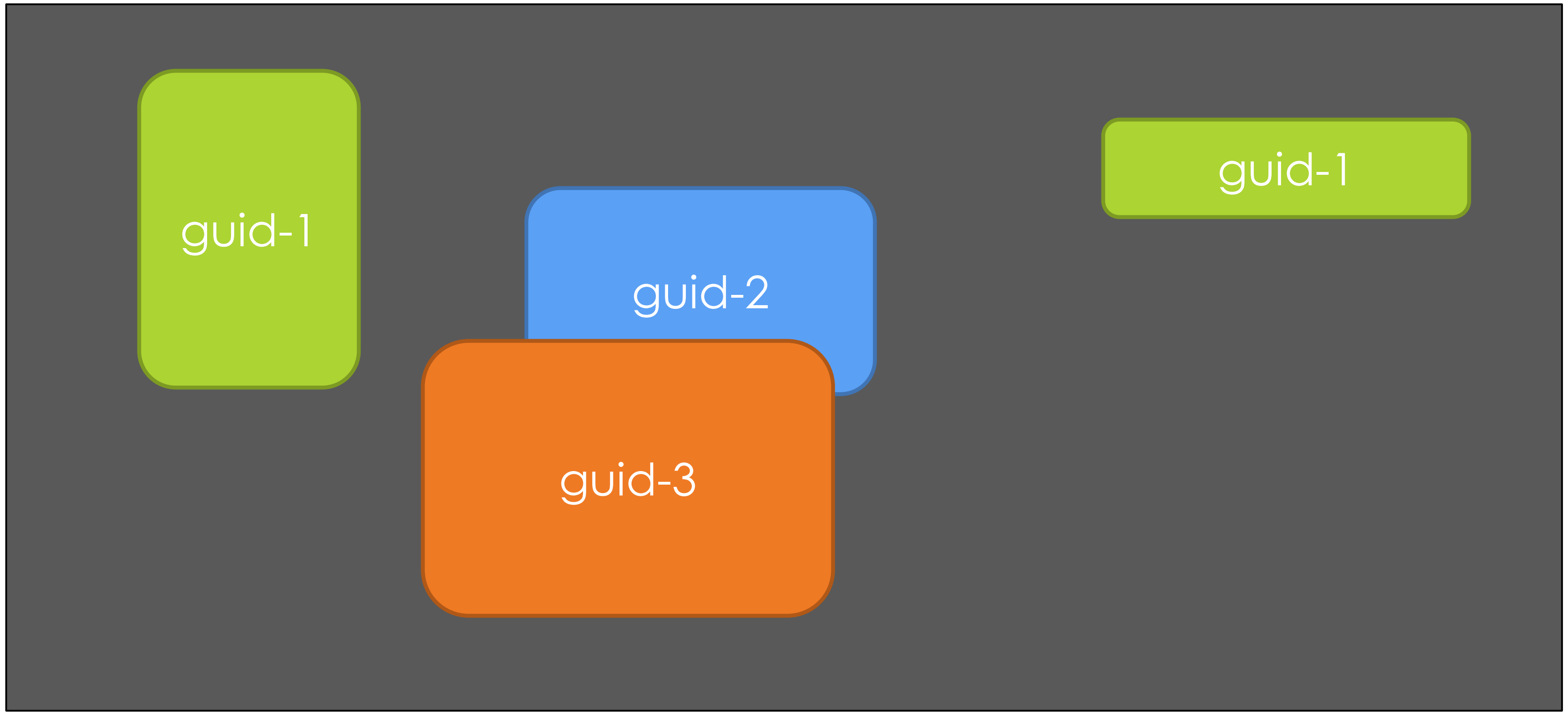
- Support for deterministic builds
- Make it work without networking
  - with qubes yum proxy at least
- Better support for non-Linux-based systems
- Use VMs instead of chroot for better isolation of components builds

Porting Window Managers

Porting Windows Managers to Qubes

- Intro
  - Trusted desktop
  - guid
  - Exported window properties
- Customization
  - Remove useless features (e.g. File Manager, Web browser not needed in Dom0)
  - Ensure “Start Menu” usable with Qubes app shortcuts
- Packaging
- Example: Awesome

Qubes desktop



guid process exports X properties

- _QUBES_VMNAME
  - String containing VM name
- _QUBES_LABEL
  - Index to table of supported VM “colors”

Appmenus

- Shortcuts for starting apps in AppVMs
- Standard .desktop files
  - Exec = qvm-run –a personal Firefox
  - Automatically generated by core-admin-linux, data obtained from VMs
  - qubes.GetAppmenus
  - qubes.GetImageRGBA
- For most Window Manager nothing is required to be done here

Preparing New VM Templates

What makes a Qubes VM template?

- root.img
- Qubes-specific agents
  - GUI
  - qrexec
  - qrexec service (qubes.FileCopy, qubes.OpenInVM)
- startup core scripts
- VM services stripping
- /rw linking (e.g. /home, /usr/local)
- Qubes templates in R2 are assumed to be PV, not HVM

Typical steps when making template

- Boot distro as HVM
- Build and try Qubes core & GUI agents
- Packaging of the Qubes components (rpm, deb, etc)
- Installing distro in chroot (e.g. debootstrap)
  - For qubes-builder (for components builds)
  - Also for template builder (for root.img build)
- Adjusting builder scripts
- Building and packaging the template
- Posting to qubes-devel, getting template uploaded to templates-community "AppStore" :)

Example of practical porting

- Debian and Arch Linux Templates builds examples
- https://wiki.qubes-os.org/wiki/BuildingNonFedoraTemplate
- https://wiki.qubes-os.org/wiki/BuildingArchlinuxTemplate

What's coming in Qubes R3?

Qubes Odyssey Philosophy

- Qubes is not a hypervisor/VMM
- Isolation is (relatively) easy, it's integration requires for desktop system that is hard. Qubes specializes in the latter.
- Users of other VMMs might also benefit from "Qubes approach"
- Let's treat VMM as "isolation provider", allow to use different VMMs
- Other VMMs might offer following benefits
  - Better h/w compatibility (perhaps sacrificing some security)
  - Better isolation, e.g. covert channel reduction (perhaps sacrificing some performance)
  - Different usemodels, e.g. Qubes in the Cloud?

Qubes Odyssey HAL
(Hypervisor Abstraction Layer)

- Run any hypervisor instead of Xen
  - KVM, VMWare, Hyper-V, Linux LXC even perhaps
- Porting comprises
  - libvirt driver for particular VMM
  - VMM-specific implementation of vchan
  - A few config files (VMM config, storage config)
  - GUI daemon: (presently ugly)#ifdef replacement for xc_map_foreign_pages()

Qubes R3

- Qubes R2 rewrite to Odyssey HAL
- Core to be released soon
- New release cycle & versioning:
  - 3.0-rc1, -rc2, ... (ISO might not build even!)
  - 3.0.0, 3.0.1, 3.0.2 < stable release, bug fixes only
  - 3.1-rc1, -rc2, ... < introducing new features
  - 3.1.0, 3.1.1 < stable release with bug fixes
  - etc

Master Signing Key

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