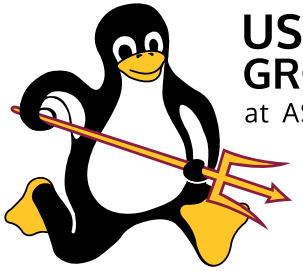
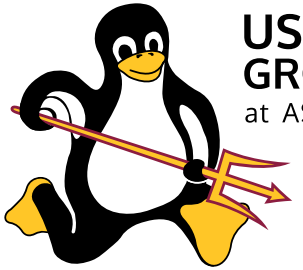


LINUX
USERS
GROUP
at ASU

Qubes OS
A Technical Introduction

Daylam Tayari

Technical?

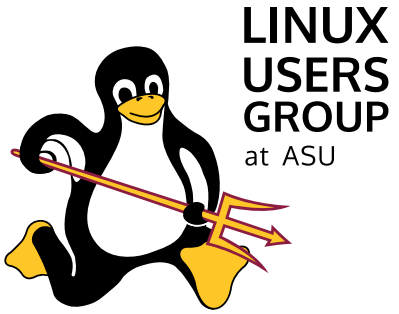


Don’t worry – This won’t be exclusively technical

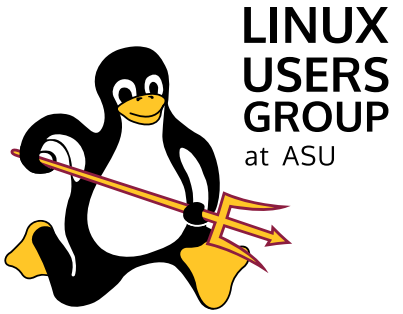
Overview → Specificities → Technical Details

Questions at any point?
Ask!

Talk Layout

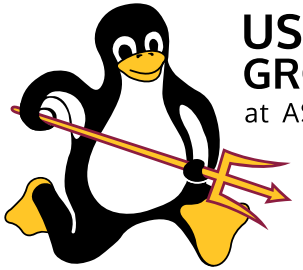


1. Overview
2. Usage and Security Implementations
3. Drivers, Interfaces and Policies
4. Technical Analysis of the Architecture
5. Security and Attack Surfaces



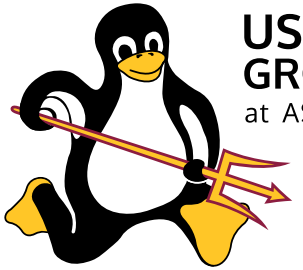
Section 1 – Overview

What is Qubes?



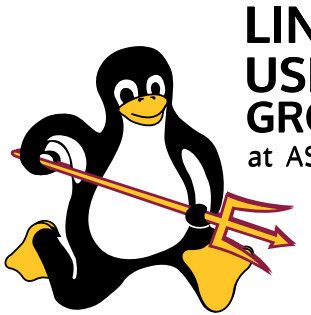
- An operating system based on a hypervisor.
- Everything compartementalised.
- Separation at every level – Hardware does not interact directly with the end usage (normally).
- All your regular actions are in VMs.
- Entirely open source.

History

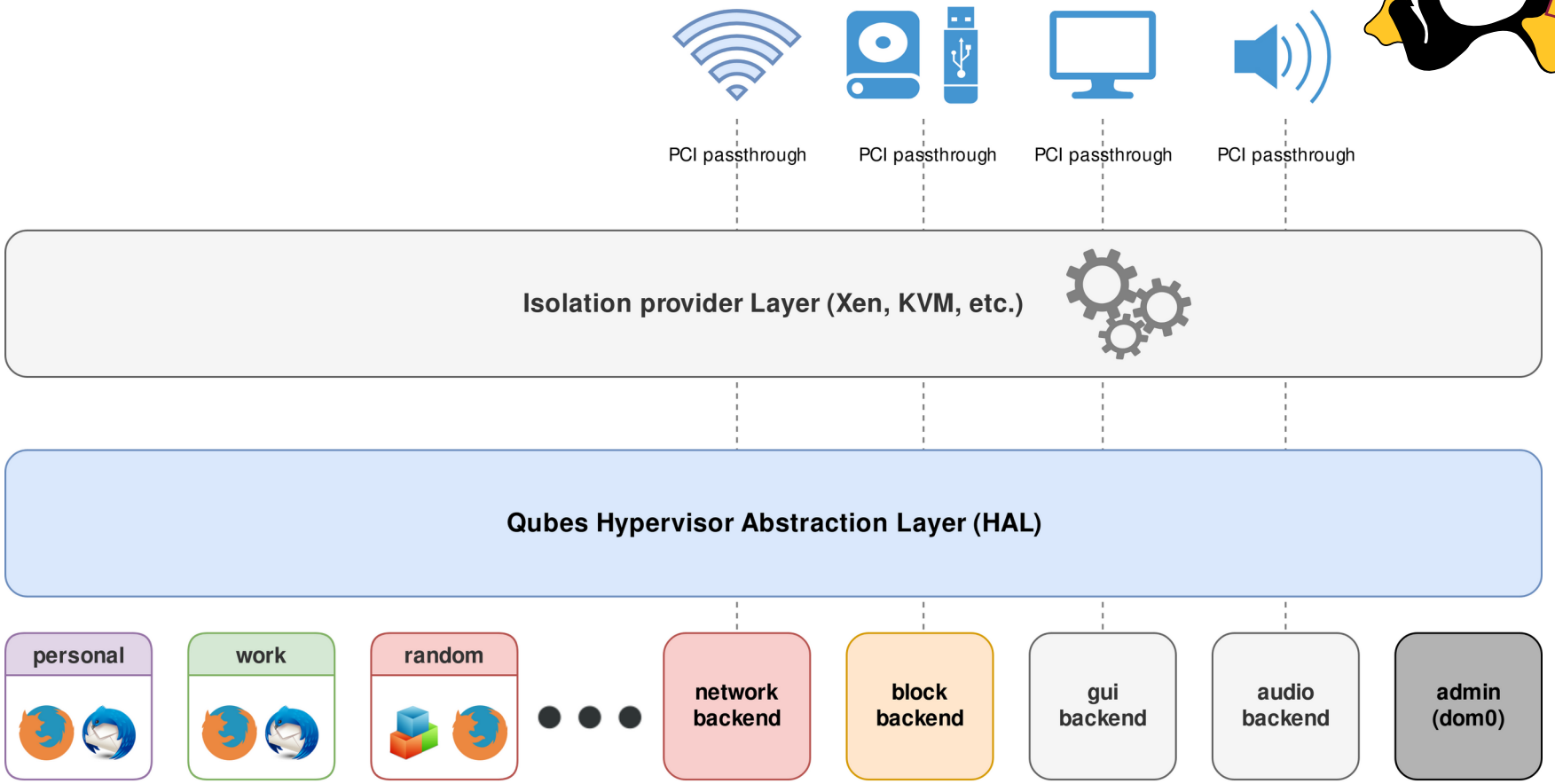


- Introduced in January 2010.
- Founded by Joanna Rutkowska.
- Compartementalised approach for security.
- Currently led by Marek Marczykowski-Górecki.
- Latest version is 4.1.0 released 4th of February 2021.

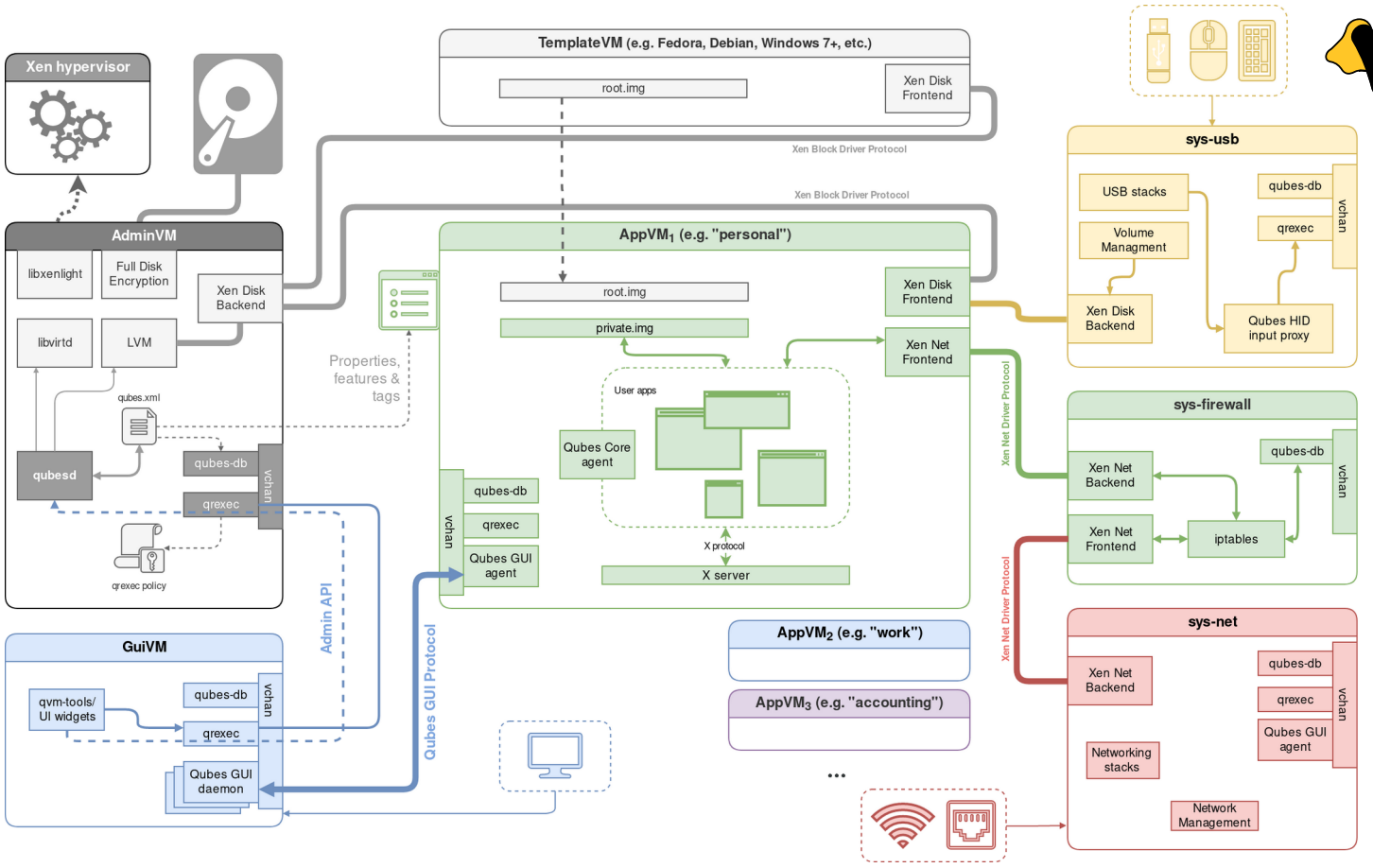
Architecture



LINUX USERS GROUP at ASU

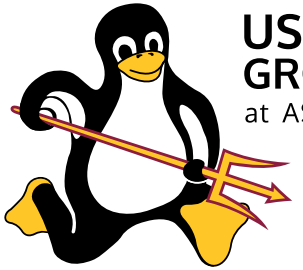


Core Stack



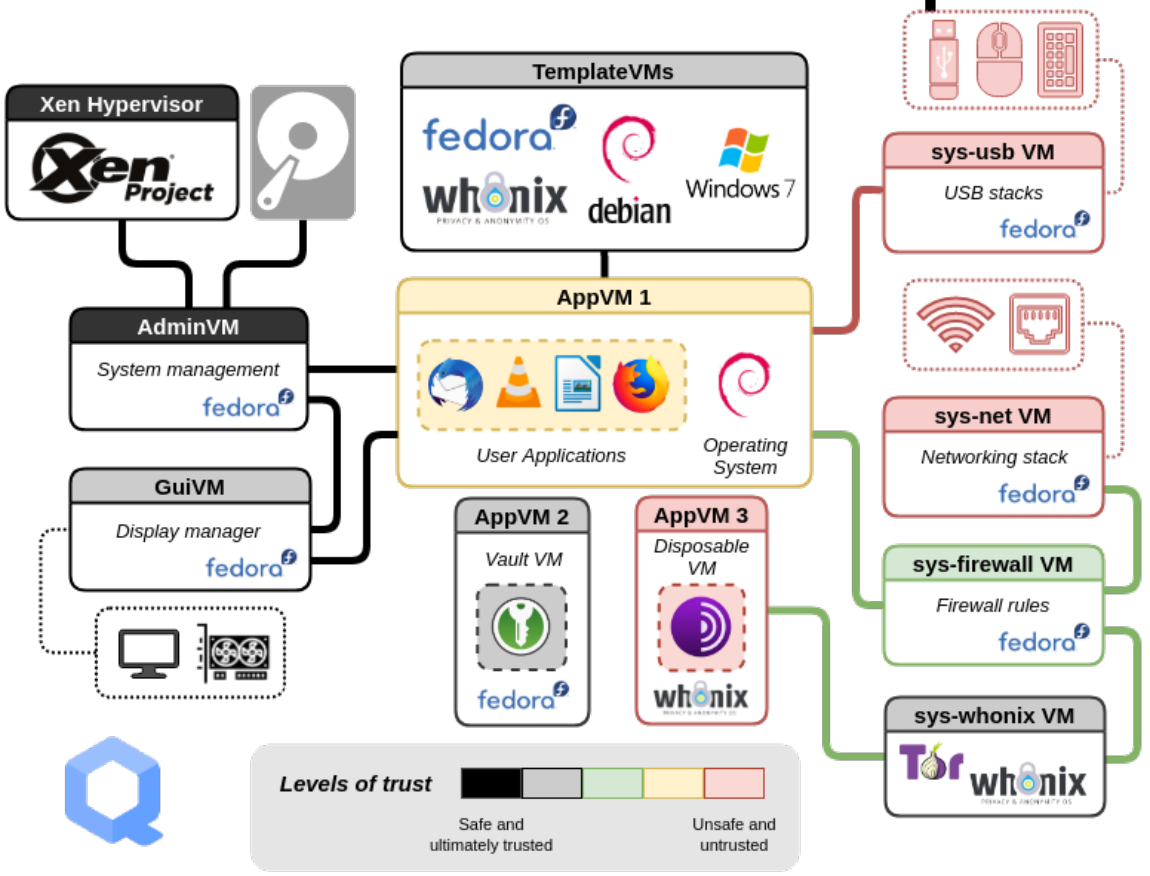
LINUX USERS GROUP at ASU

Working in AppVMs

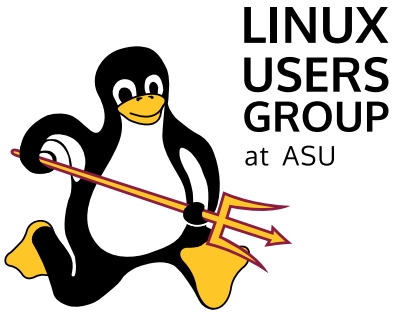


- AppVMs are where you do all your work.
- They can run everything – even Windows
- Connected to the hardware through other VMs.
- Interact with the core and other VMs with drivers.

Use Case Example

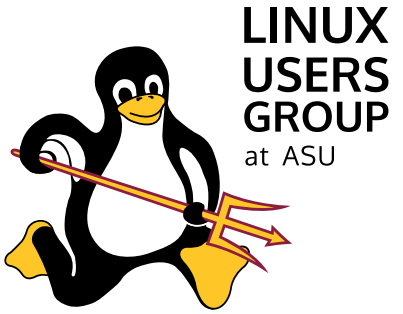


Why?



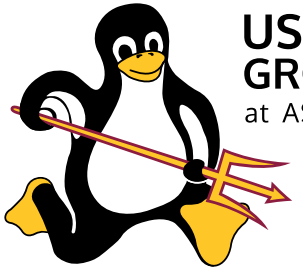
- Security by Isolation Approach
  - Virtualisation
  - Compartementalisation.
  - Need to Have approach.
  - Disposable VMs.
- Privacy
  - Whonix and Tor
  - Preventing Fingerprinting.
- Compartmentalisation of your Workspace.
- Changing Systems Often

Why Not?

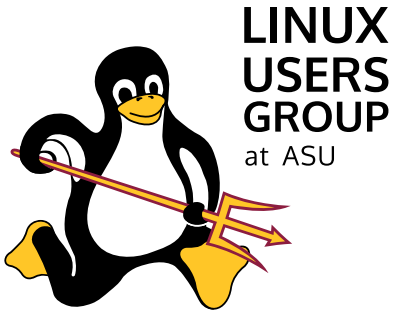


- Technical experience occasionally required.
- Resources
  - Overhead
  - Sharing
  - Device Specifications
- Thinking in a Qubes way
- Technical Restrictions

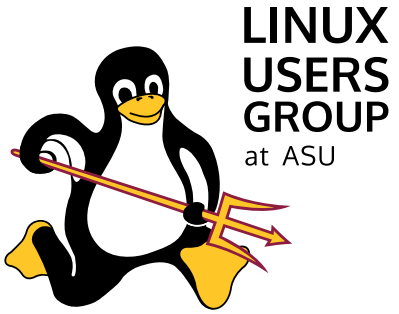
Why do I?



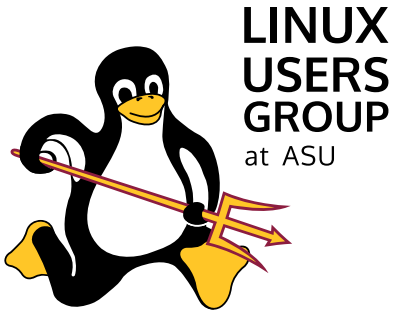
- Compartementalisation
- Security
  - Isolation
  - Need to Have
- Mirroring across devices
- Whonix



Section 2 – Usage and Security Implementations

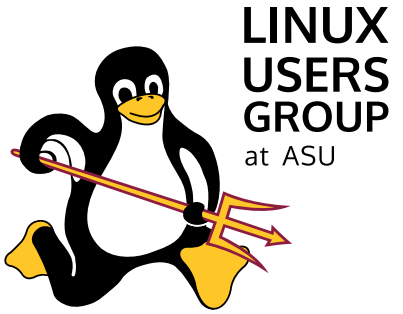


Section 2.1 – Virtual Machines in Qubes



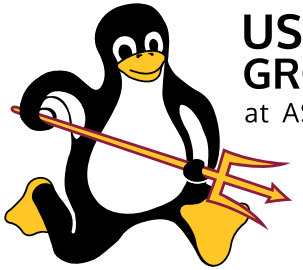
Section 2.1.1 – Virtual Machine Types

Types of VMs



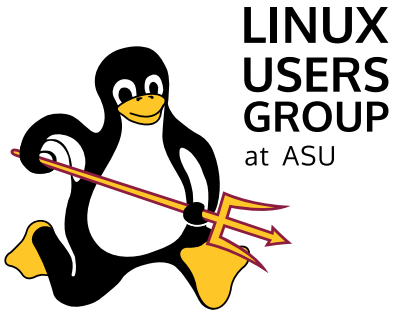
- DispVMs – Based off template and no persistance.
- AppVMs – Based off a template and limited persistance.
- TemplateVMs – Root file system for AppVMs.
- StandaloneVMs – ‘Traditional’ VM.
- AdminVM – The core, dom0.

DispVMs



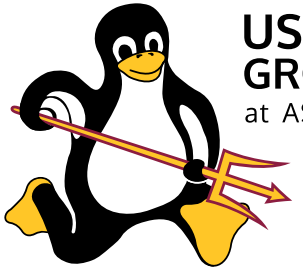
- Disposable VMs are temporary disposable systems.
- Based off a template.
- Inherits the entire template including `/home/`.
- No persistance whatsoever, fully wiped on exit.
- Has a temporary volume though can be set to only use RAM.
- Files can be easily opened to be viewed.
- Randomly numbered between [1-10000).
- **Demo!**

AppVMs



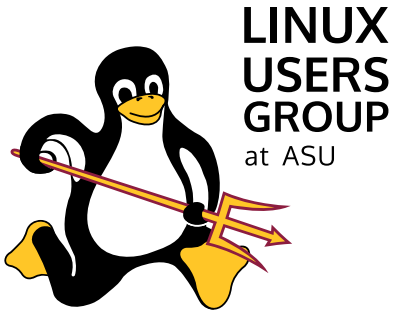
- Based off a template.
- Entire `/rw/` directory persists which includes:
  - `/home/`
  - `/usr/local/`
- Inherits part of its `/home` and `/usr/local`.
- Is provided an image of the template VM which was **generated at its last shutdown**.
- Where you will do all your work.
- **Demo!**

Template VMs



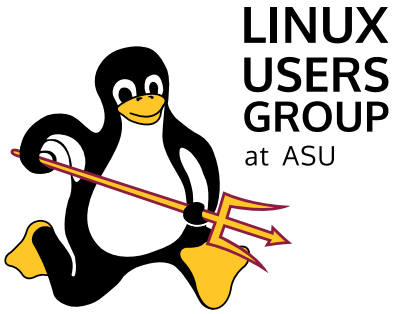
- Templates have no inheritance, recursive templates is not a possibility.
- Everything persists so **be careful**.
- No network by default.
- Its image is updated at shutdown.

Standalone VMs

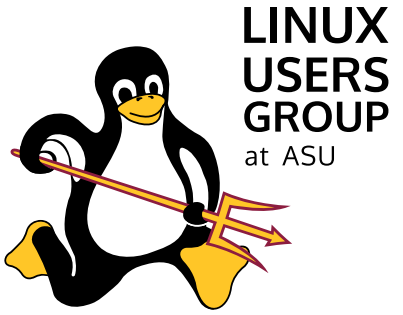


- Traditional VMs.
- Either independent clone of template or seperately installed system.
- No integration though possible with drivers.

AdminVM

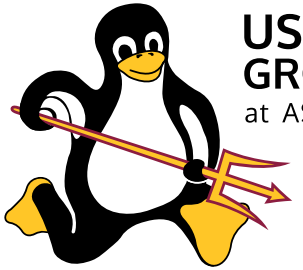


- AdminVM is dom0.
- Is the backbone of the operating system.
- Where the entire backend of the OS is ran.
- No network connectivity.
- Updates can only be made to specific repositories.
- Runs Fedora 32.
- Isolated from the rest of the system – No clipboard integration.



Section 2.2 – Template VMs

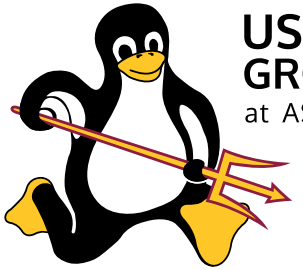
Template Repositories



LINUX USERS GROUP at ASU

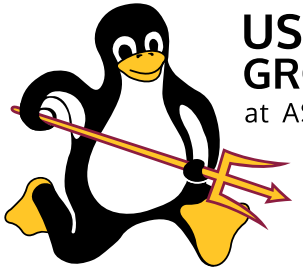
- Official Templates – Built and released by the Qubes team.
- Community Templates – Designed, built and maintained by the community.

Official Templates



- Fedora 34
  - Minimal Template
  - Xfce Template using Xfce instead of GNOME.
- Debian 11
  - Minimal Template
- Whonix 16 Gateway and Workstation

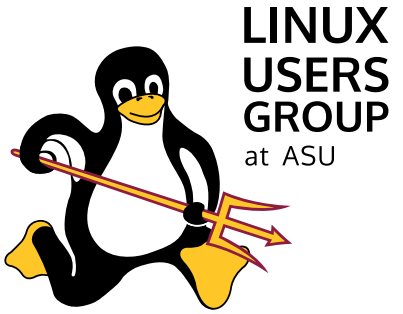
Community Templates



LINUX USERS GROUP at ASU

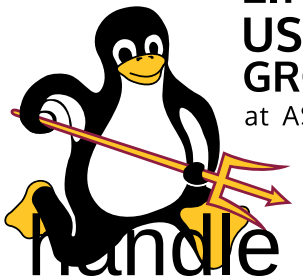
- Whonix
- Ubuntu
- Arch Linux
- CentOS
  - Minimal Template
- Gentoo
  - Minimal Template
- NetBSD

Custom Templates



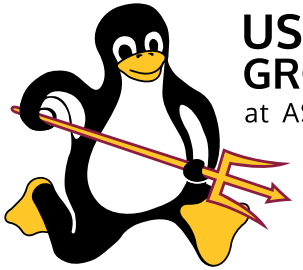
- Templates which aren’t standalone but can be implemented.
- Kali Linux
- Parrot

Updating



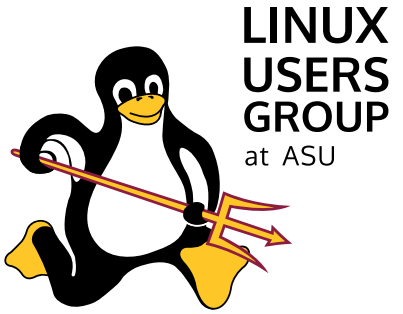
- Qubes has an updating utility which will handle proper updating for Templates.
  - Creates a mirror.
  - Updates.
  - If successful → Updates the actual Template
- Manually in the individual TemplateVM.

Installing Programs



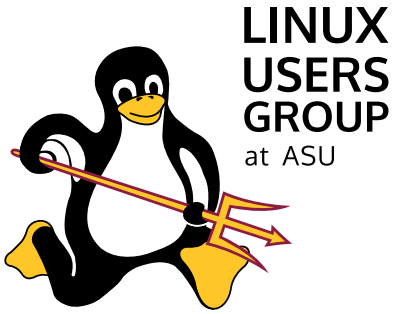
LINUX USERS GROUP at ASU

- Using the package managers.
- Sending over binaries and repositories and installing them locally.
- Adding them manually in directories such as `/opt/` and creating respective desktop files.



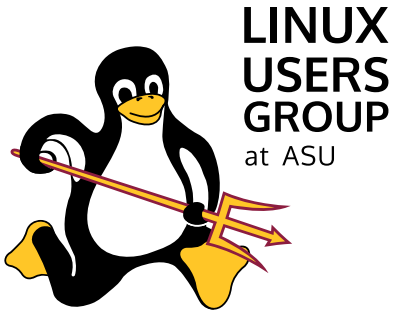
Section 2.3 – Migration and Backups

Cloning



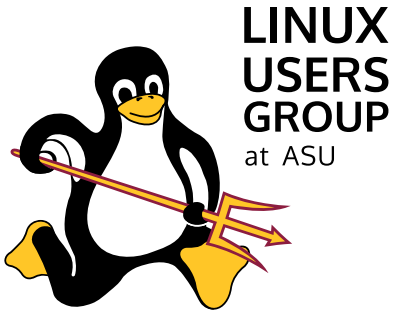
- Any VM, regardless of type can be easily cloned.
- Cloning a VM is accessible directly in the basic settings.

Qubes Backups



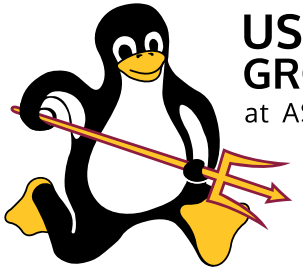
- Qubes official backup utility.
- Packages a set of VMs into a backup.
- Saves the version of the last shutdown.
- Saves specific file systems:
  - AppVMs – private.img
  - TemplateVMs – root.img
  - StandaloneVMs – root.img and private.img
- Encrypts backups with scrypt.

Backup Encryption



- Uses the scrypt encryption utility.
- Initially developed for the Tarsnap online backup system.
- Uses a complex cryptographic system including both signing and encryption, both which are required for decryption.
- Qubes uses their release signing key for the signing.

Tarsnap Cryptography

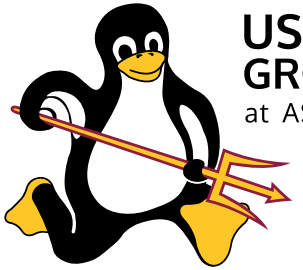


LINUX USERS GROUP at ASU

- A 2048-bit RSA key used for signing archives. This is used in combination with SHA256 and a Merkle hash tree to verify the authenticity of stored archives.
  - Cryptographers: RSASSA-PSS is used, with SHA256 as the hash function.
- A 2048-bit RSA key used for encrypting session keys. All the data which the Tarsnap client sends to the server to store is encrypted with per-archive random AES-256 keys; those keys are encrypted with this RSA key and attached to the stored data.
  - Cryptographers: RSAES-OAEP is used to encrypt the session keys, using MGF1 and SHA256, and the padding verification performed when decrypting is carefully written to be free of timing side channels. The AES-256 keys are used in CTR mode, with sequentially incrementing nonces — generating a new AES-256 key for each session ensures that a key-nonce pair will never be used twice.
- A 256-bit HMAC-SHA256 key used to protect each individual block of data from tampering. From a cryptographic perspective, this is unnecessary, since a Merkle hash tree protects each archive; but data is compressed using zlib before being stored, so this provides protection against a theoretical attacker who can tamper with stored data and has found a security flaw in zlib decoding.
  - The zlib decoding process is quite complex — for that matter, the same is true of all decompression algorithms — and such complex code has a relatively large risk of having security vulnerabilities, hence the decision to add an extra layer of security here.
- Two 256-bit HMAC-SHA256 keys used to generate names for blocks of data stored. Tarsnap uses the same reference-by-hash trick as the author's Portsnap and FreeBSD Update utilities; using HMACs instead of raw SHA256 hashes prevents any information from leaking via the hashes.
  - Why two keys? One is used to hash data, and the other is used to hash archive names. Yes, that's right, even the names of archives are stored securely!

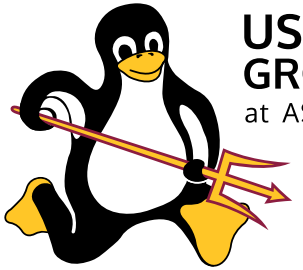
https://www.tarsnap.com/crypto.html

Backup Restore

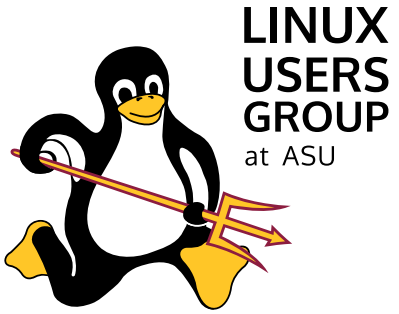


- Official Qubes Restore utility.
- Restores backups without a hitch.
- Has the possibility to solely verify integrity.
- Can either restore alongside or overwrite.
- Restoration has no difference machine to machine as long as they are on the same Qubes release.

Migration to an ISO

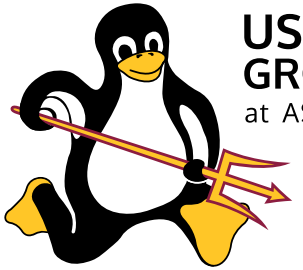


- Official support with Qubes Builder
- Allows you to turn a Qubes image into an ISO
- Theoretically should be flashable onto another computer just like any ISO.



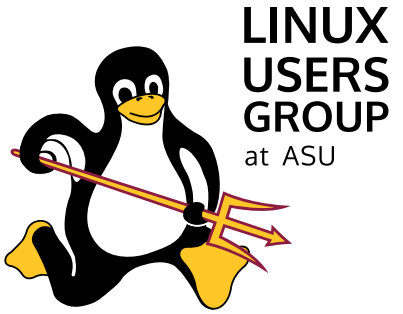
Section 2.4 – Interactions between VMs

Clipboard Interactions



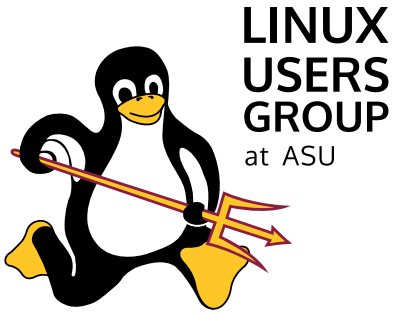
- A single elevated clipboard.
- Any VM can send their clipboard to the elevated clipboard with `Ctrl+Shift+c`.
- Any VM can retrieve the contents of the elevated clipboard with `Ctrl+Shift+v`.
- Elevated clipboard is wiped on retrieval.
- AdminVM is not integrated with the elevated clipboard.

File Interactions

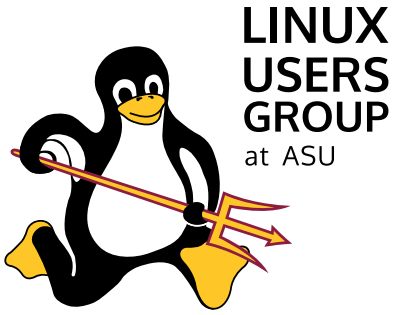


- All VMs can send files to another VM.
- All VMs except the AdminVM can receive files.
- Files will arrive in the `~/QubesIncoming/VM_NAME/` folder.
- AdminVM can only receive files from other VMs by passing the IO from the VM to the AdminVM.
  - i.e. `qvm-run --pass-io QUBE_NAME 'cat NAME_OF_FILE ' > NAME_IN_DOM0`

Split Utilities

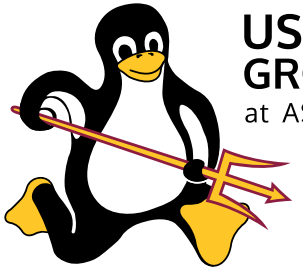


- Split GPG
- Split SSH
- Split Electrum Wallet
- Split dm-crypt
- U2F Proxy



Section 2.5 – Networking

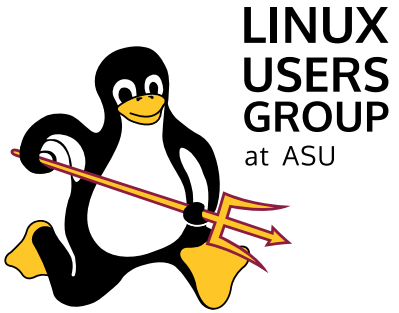
Networking Overview



LINUX USERS GROUP at ASU

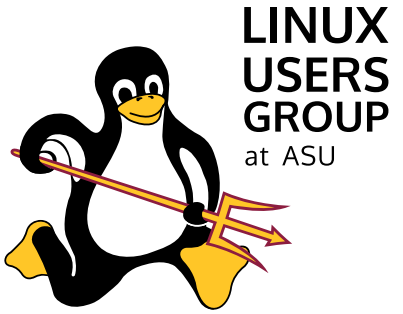
- Any VM can be set to be a networking VM.
- There are two primary VMs responsible:
  - Hardware VM (i.e. WiFi, Ethernet, etc.)
  - Firewall VM
- Other VMs can be configured to handle and adapt a network connection.
- Network VMs can be chained.
- Changing networks is as simple as changing one setting and is applied instantly.
- You select the final VM as the one you want to use.

Hardware VMs



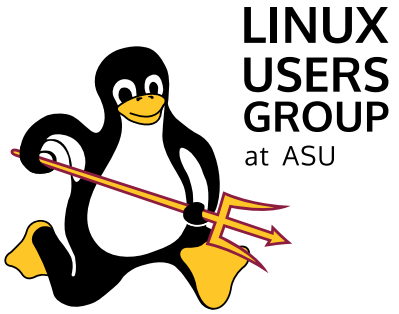
- Hardware networking VMs are those that interact with the actual PCI devices.
- Good practice is one per PCI device.
- Lets everything through.
- Uses NetworkManager by default.

Firewall VM



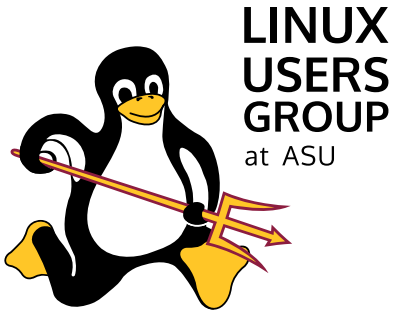
- Firewall rules can be set for each Qube.
- Applies all network-level policies.
- Responsible for blocking inter-VM networking.

Custom VMs

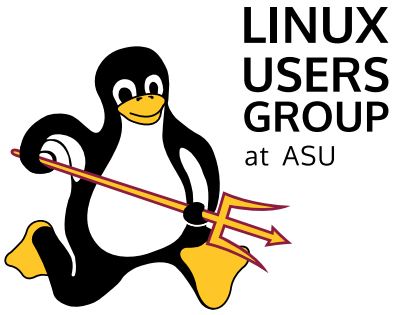


- No limit in the amount of network VMs that can be created and made.
- You can create a VPN VM and everything that is passed through goes through the VPN.
- You can deploy a full on DNS sinkhole (i.e. Pi-hole) or a firewall if you wanted and chain it.

Whonix Networking

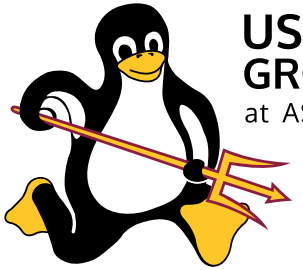


- sys-whonix VM running Whonix Gateway.
- All connection passed through it are passed through Tor.
- Tor connection is fully configurable.
- Just like any other VM.



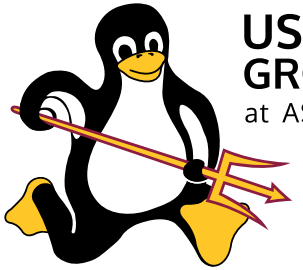
Section 2.6 – USB Handling

USB Qubes

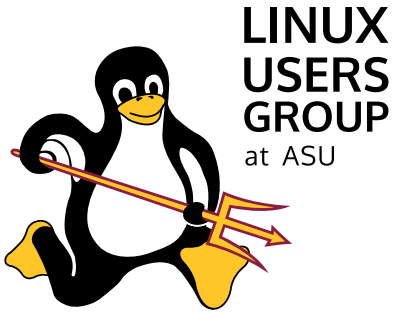


- Separate VM which interacts directly with the USB PCI device.
- Without a USB Qube, the USB PCI device interacts directly with dom0.
- To use or have access to a USB device in a VM, it HAS to be manually passed throught the Qubes Devices taskbar menu.
- **Demo!**

Complications

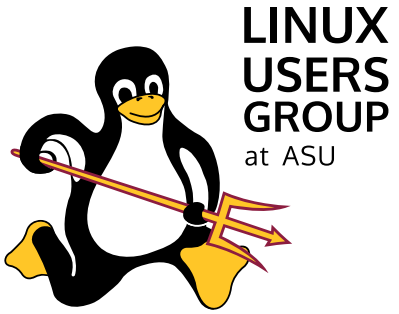


- Necessary passthrough for login
  - Passthrough of USB keyboard device to dom0 on boot through a policy.
- Connection not as stable as if directly passed through.



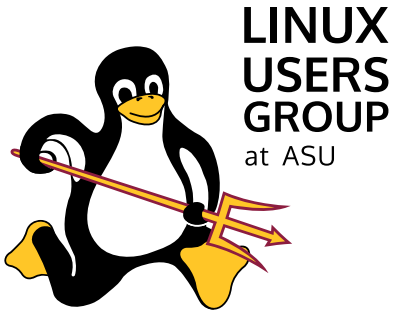
Section 2.7 – VM Configuration

Basic Settings



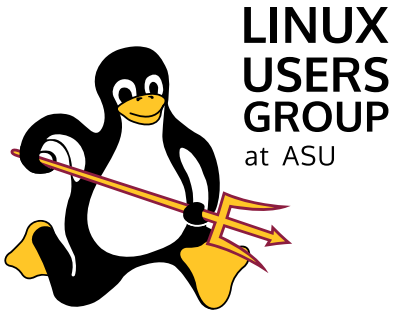
- Name and label colour
- Template
- Networking VM
- Private storage size
- Start on boot
- Include in backups
- Debug mode

Advanced Settings



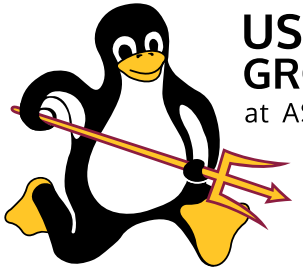
- Initial Memory
- Max Memory (only if memory balancing is enabled)
- VCPUs Number
- Kernel
- Virtualisation
- Disposable VM Template (if not itself)
- Provides Network Toggle
- Boot from ISO

Firewall Rules



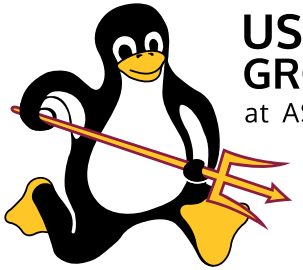
- Configure firewall rules.
- Configurable by:
  - Address
  - Port
  - Protocol
- Temporary Access

Devices



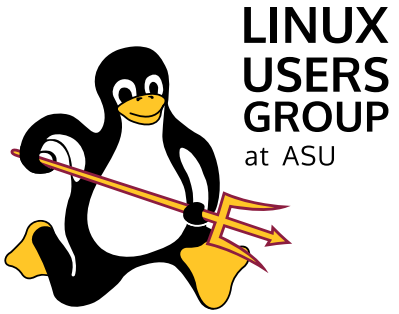
- PCI Passthrough
- Must use HVM or PV virtualisation and have disabled memory balancing for ideal results.
- Limit in types of PCI device that can be passed depending on the VM type.

Applications

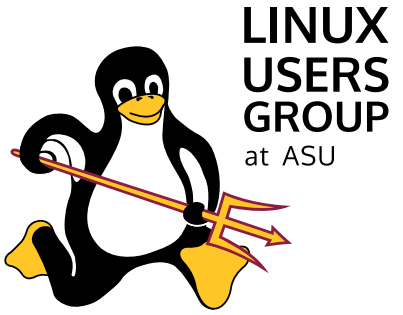


- Select which applications show up in the application menu for that VM.
- Grabbed from all `.desktop` files in the `/usr/share/applications/` directory.
- Refreshed from Template VM.

Services

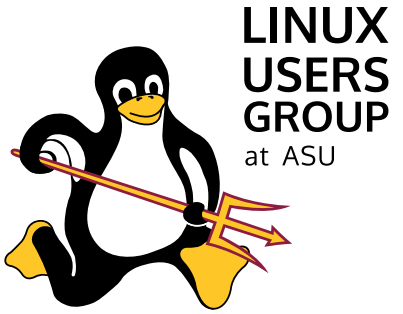


- Add or remove specific services.
- Enable or disable specific added services.



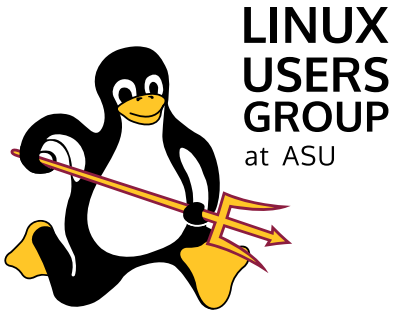
Section 2.8 – Customisation

DEs and WMs

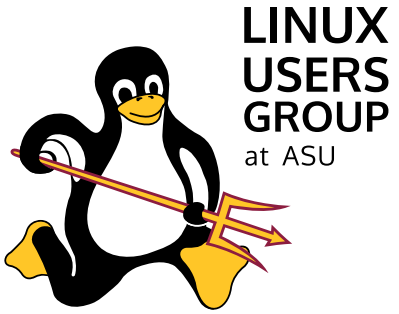


- Desktop Environments officially supported:
  - Xfce (default)
  - KDE
- Window Managers (community support):
  - i3
  - AwesomeWM
  - bspwm (unofficial)

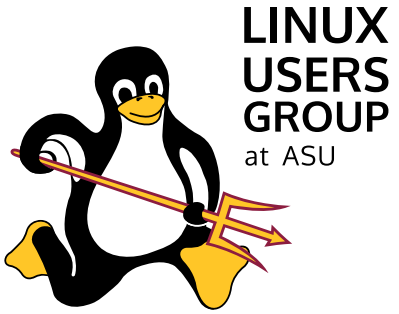
Security when Ricing



- Passing programs to dom0
- Affecting dom0
- Restriction in availability of programs.

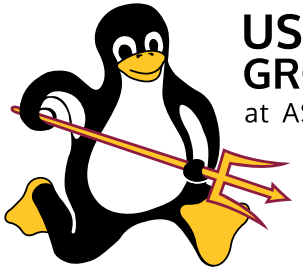


Section 3 – Drivers, Interfaces and Policies



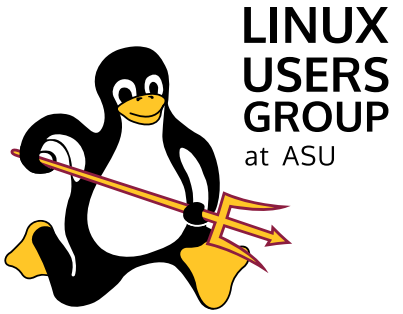
Section 3.1 – Intercommunication Drivers

Clipboard

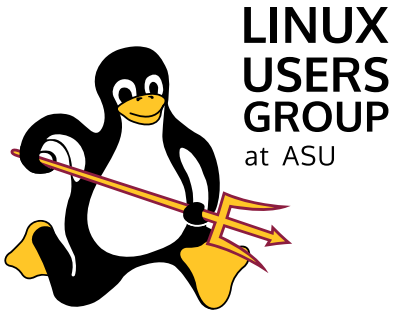


- A single elevated clipboard.
- Any VM except dom0 can copy and paste from it.
- It can only store a maximum of one entry.
- When it is pasted in a VM, it is wiped.

File Transfer

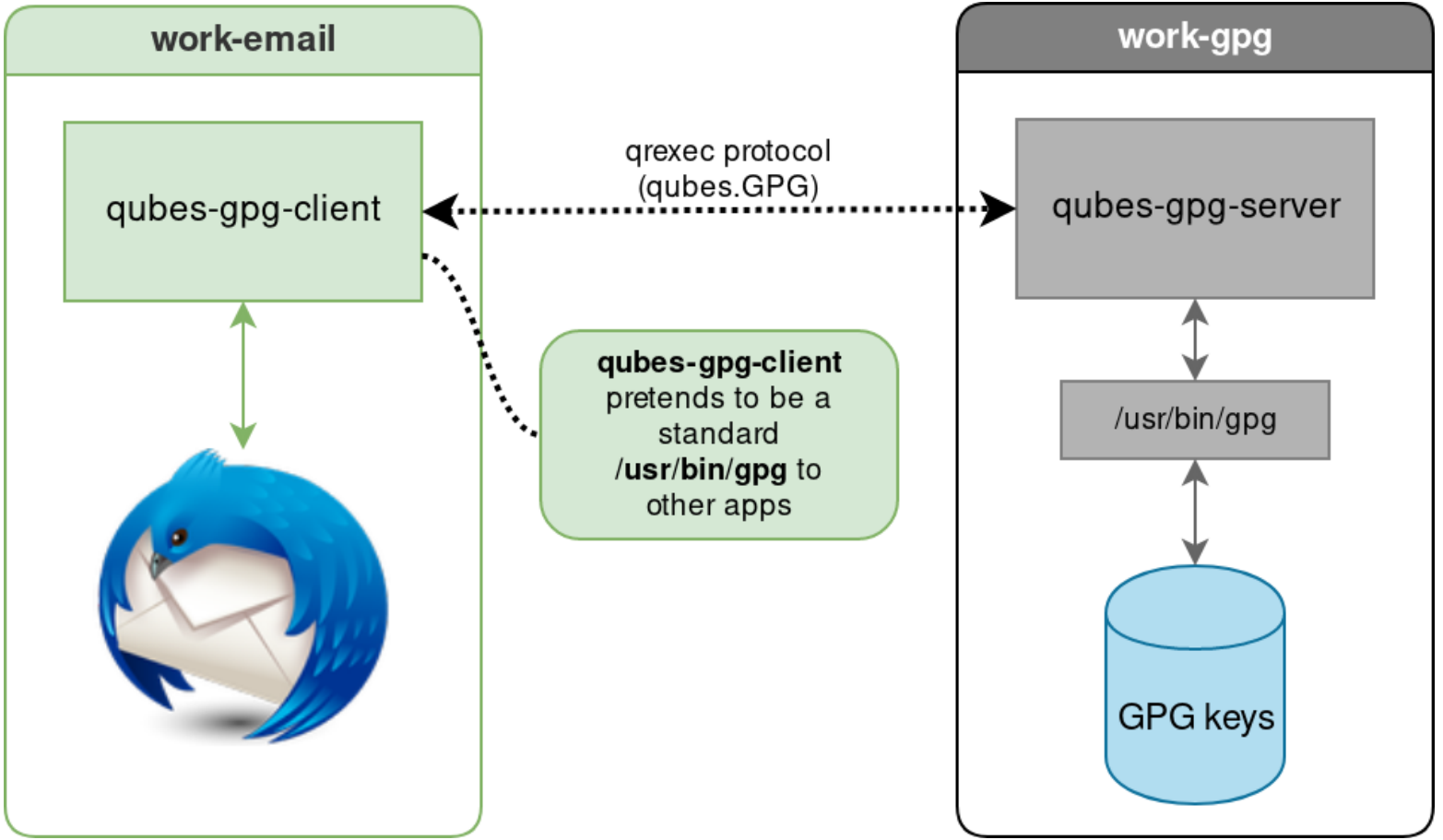


- Files can be transferred between all VMs except dom0.
- Files will be sent to the `~/QubesIncoming/VM_NAME/` directory.



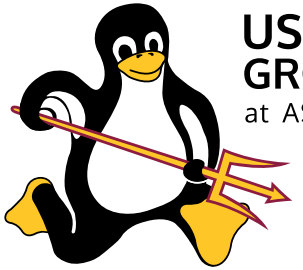
Section 3.2 – Split Interfaces

Split GPG - 1



LINUX USERS GROUP at ASU

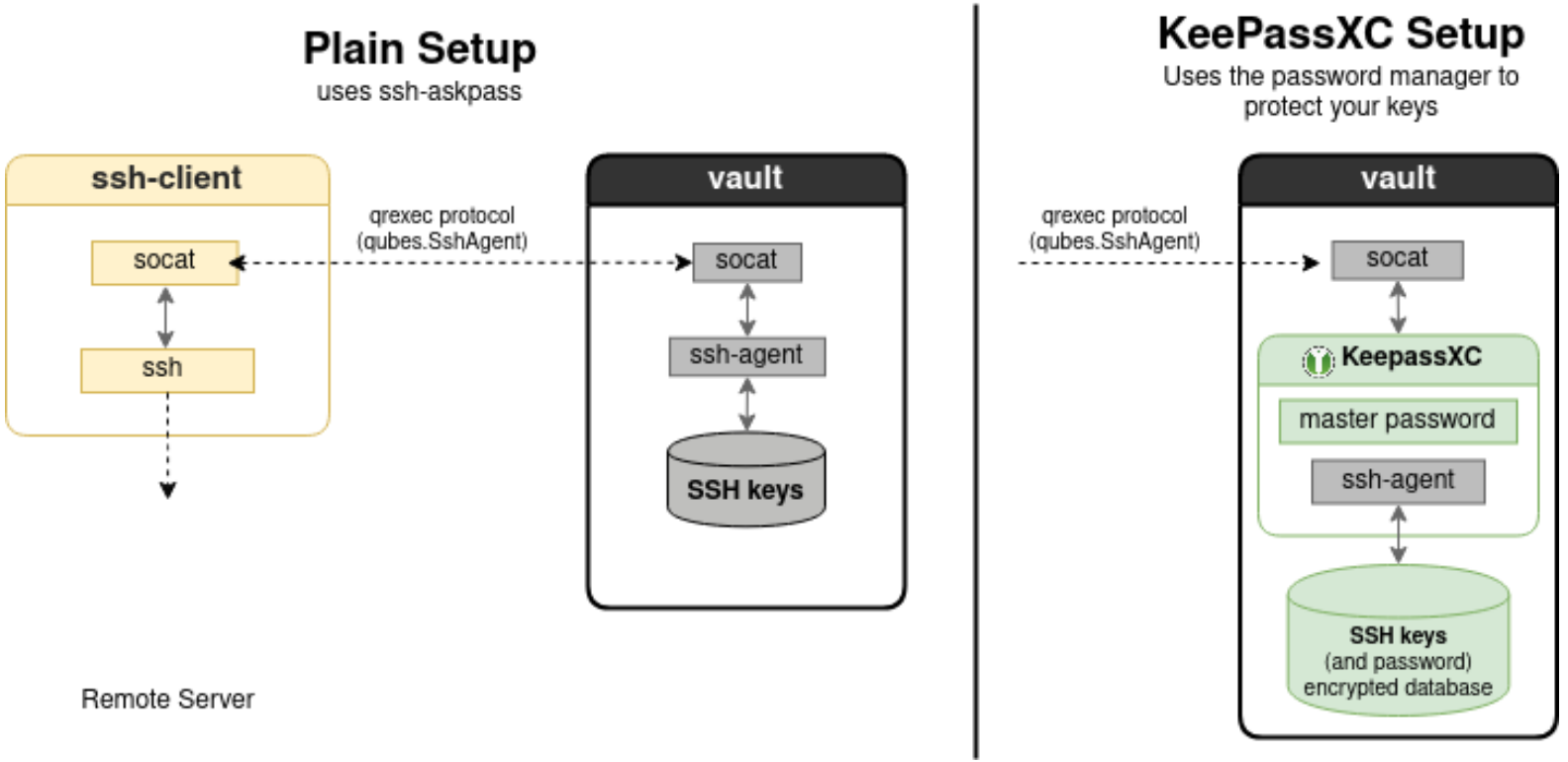
Split GPG - 2



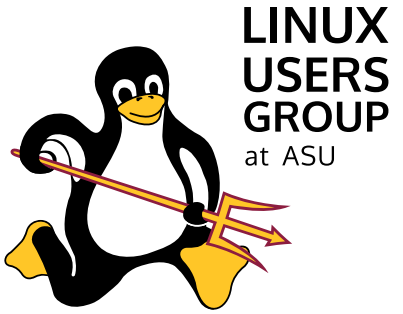
- All keys are stored in a seperate VM which is ideally fully isolated.
- The qubes-gpg-client is what interacts with the client, passing all the commands over to the GPG program on the GPG Qube and sending its output back.
- Two sets of prompts appear at each GPG usage.
- Access can be determined to a set amount of seconds.
- Passwords are entered and validated directly in the GPG Qube.
- You set your GPG client to be the Qubes client in programs.
- **Demo!**

Split SSH - 1

LINUX USERS GROUP at ASU

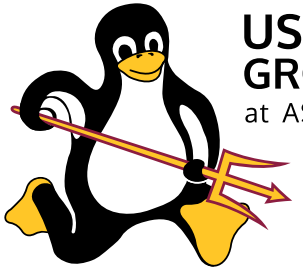


Split SSH - 2



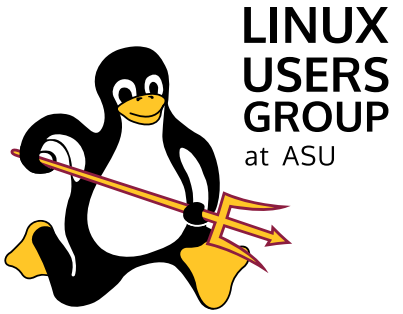
- SSH keys safe in an isolated VM.
- SSH keys are retrieved and passed through when requested.
- SSH private keys are kept in the isolated VM.
- Unfortunately, use can be held on for the entire of the existence of the VM.
- Community made.

Spit dm-crypt



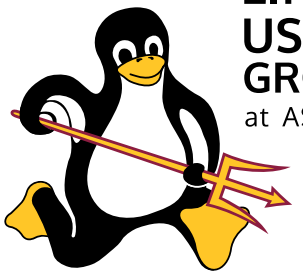
- Splits up the process of decryption of a LUKS1 partition.
- Encrypted VM is attached to a specifically configured offline disposable VM.
- Prompts for a password and decrypts the partition in the disposable VM.
- Mounts the partition into the destination VM.
- The destination VM does not have access to the password or encryption key.

Split Electrum Wallet

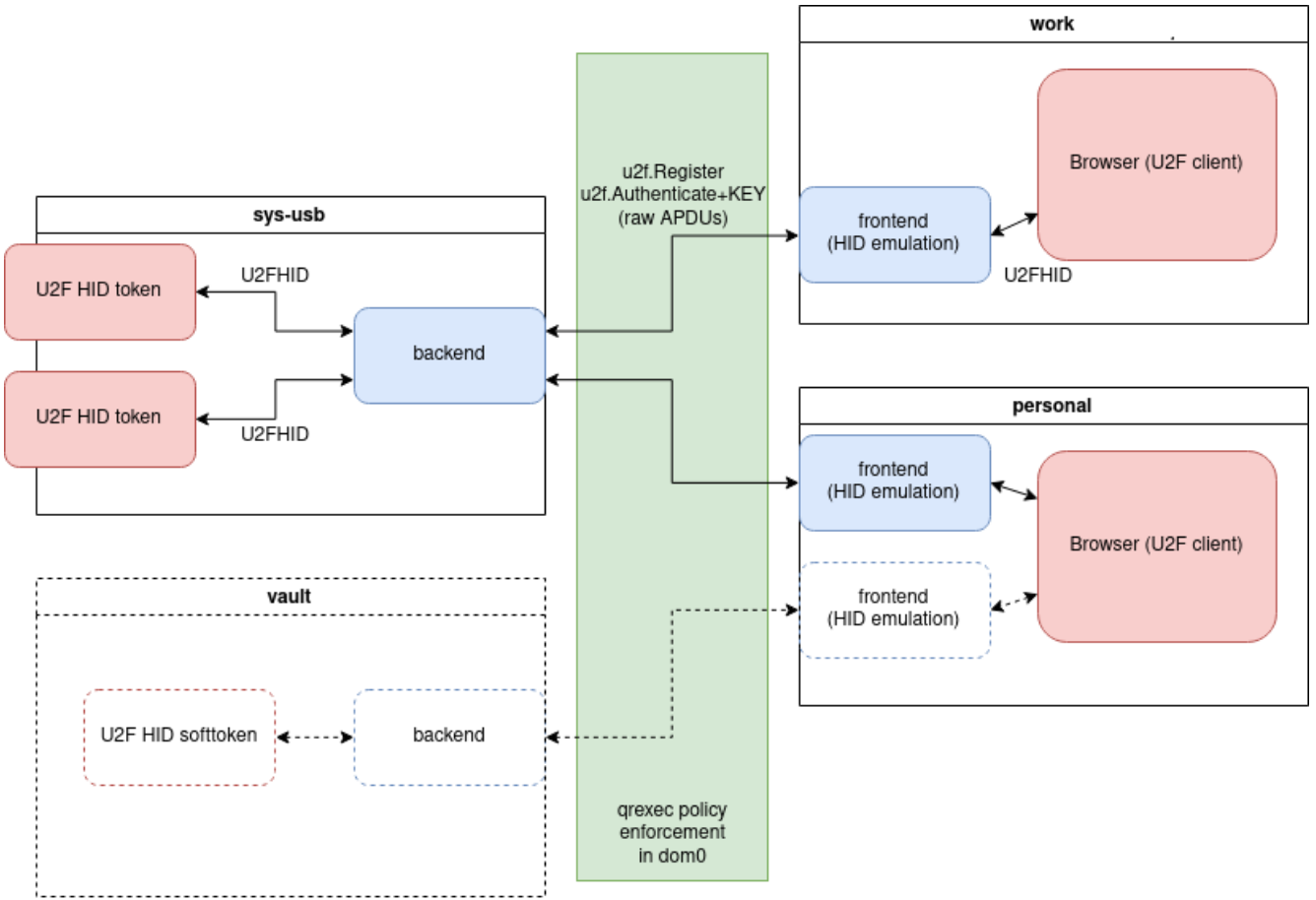


- Wallet is stored in one isolated offline VM.
- Wallet is accessed via an online watching-only Electrum wallet which connects to the isolated wallet.

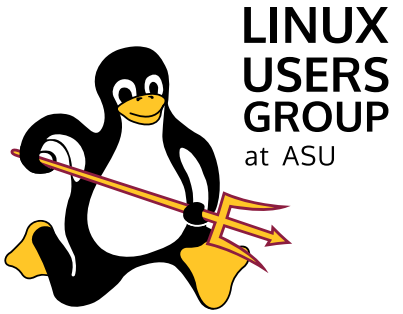
U2F Proxy - 1



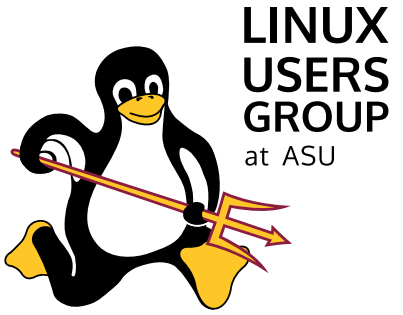
LINUX USERS GROUP at ASU



U2F Proxy - 2

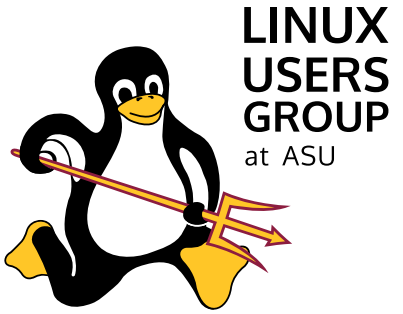


- The proxy only passes the necessary to the actual U2F key, performs the transaction and returns it to the browser without passing the entire U2F key to the device.
- Can be set to only allow specific tokens per VM.
- No direct interaction between the U2F device and the VM that is requesting the token.



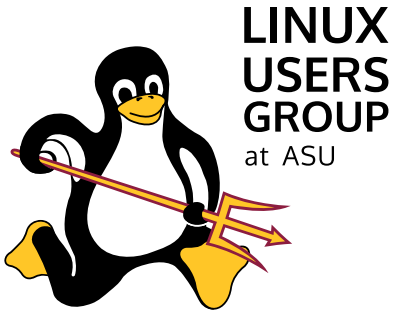
Section 3.4 – Policies

Policies in Qubes



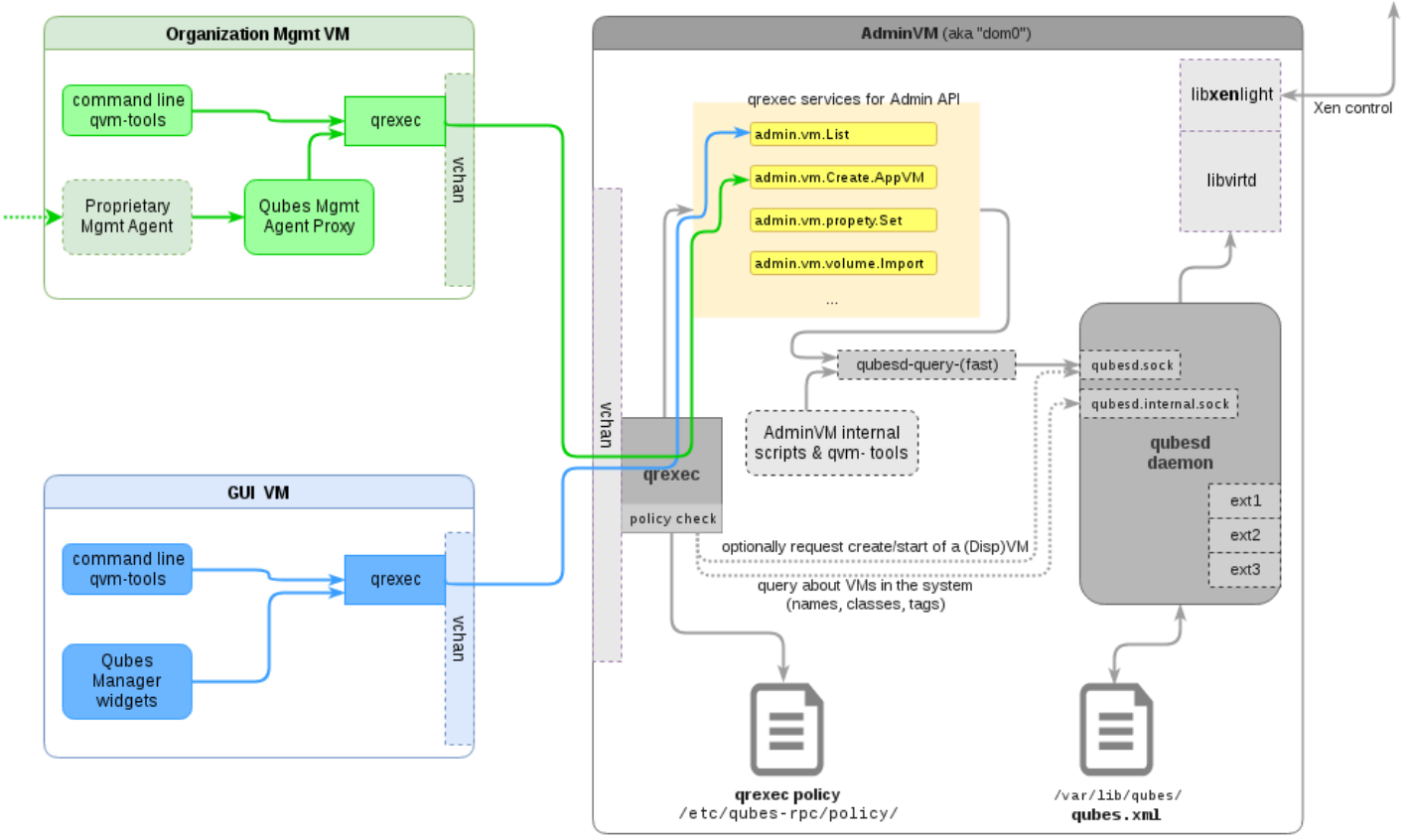
- Each interaction with a driver is controlled by a policy.
- Policies can be set both inside of a VM for its drivers or for the system.
  - `/rw/config/`
  - `/etc/qubes-rpc/policy/` in dom0

Control of Policies

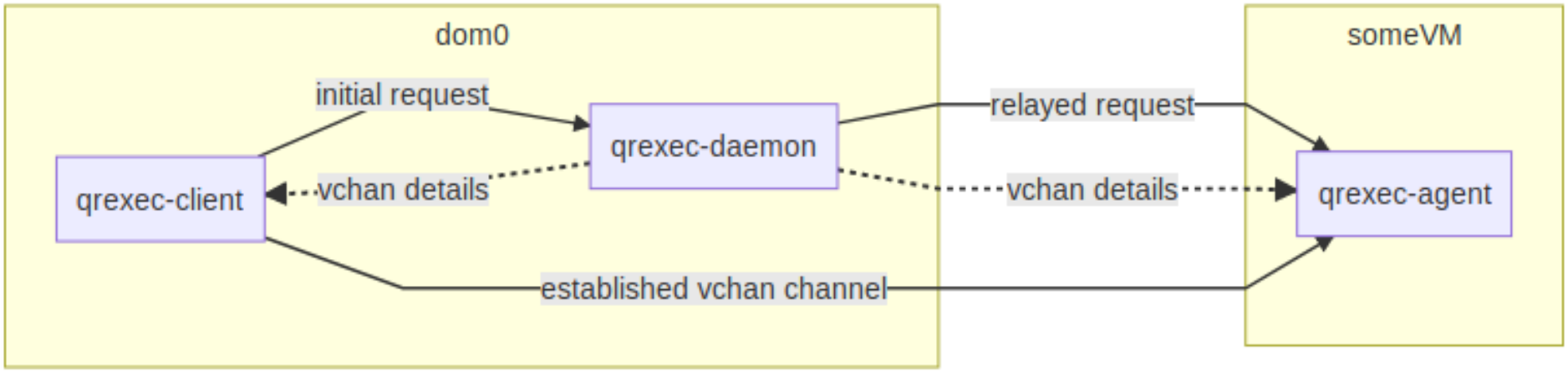


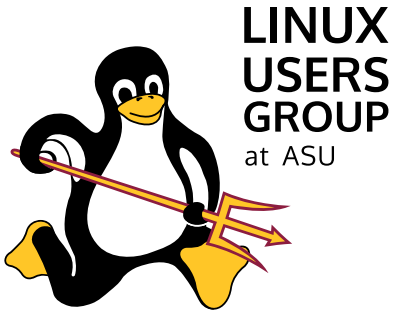
- Which VMs does the policy apply to.
- How to handle any VM not explicitly referred to.
- Handle policies depending on the tag of a VM.
- Allow or deny specific actions.

Policy Overview

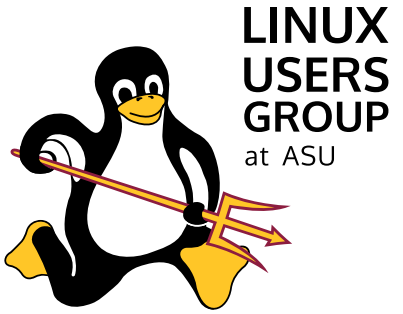


Policy Client Overview



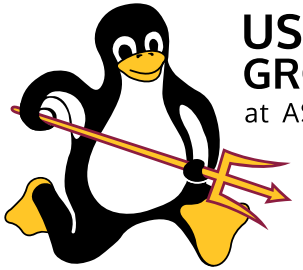


Section 4 – Technical Analysis of the Architecture



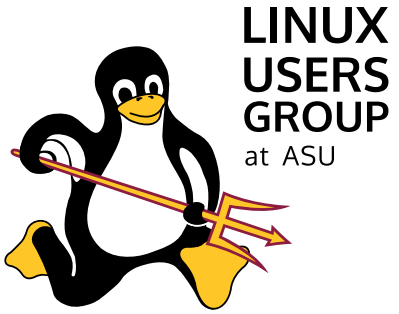
Section 4.1 – Volume System

Volume Types



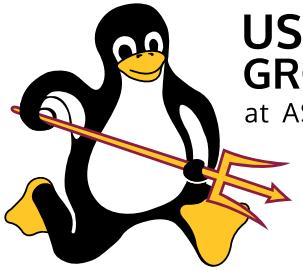
- root.img – Root device (`/`).
- private.img – Persistant volume of a VM.
- volatile.img – Swap and temporary `/`. Discarded on reboot.
- modules.img – Kernel modules and firmware.

Block Devices



- xvda – Base root (**NOT** root.img)
- xvdb – private.img
- xvdc – volatile.img
- xvdd – modules.img

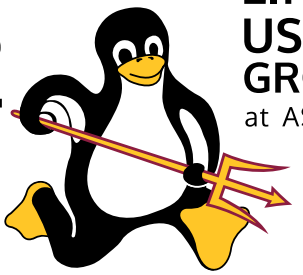
Root Implementation - 1



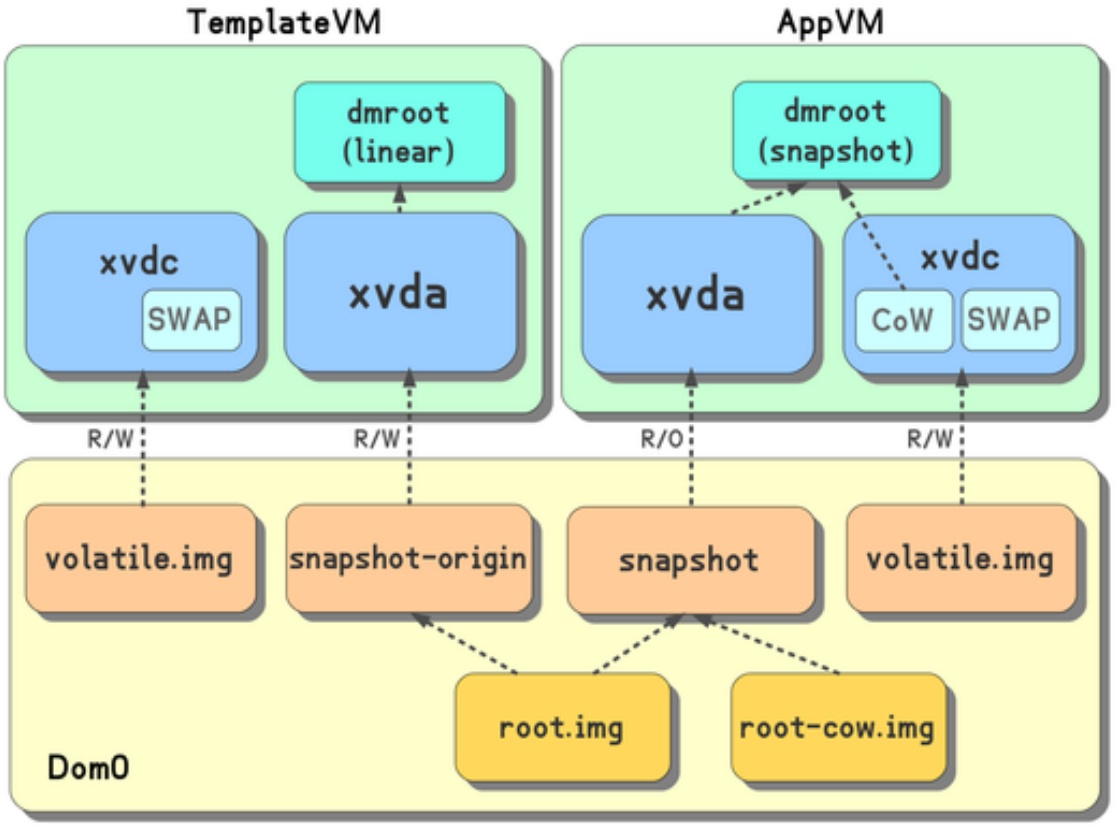
LINUX USERS GROUP at ASU

- Two versions of `root.img`:
  - `root.img` – True root filesystem.
  - `root-cow.img` – Snapshot of the root filesystem.
- Everytime an AppVM is started, a `root-cow.img` is made (unless no changes).
- AppVMs are based off the `root-cow.img`.

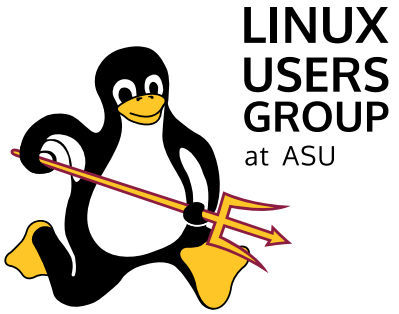
Root Implementation - 2



LINUX USERS GROUP at ASU

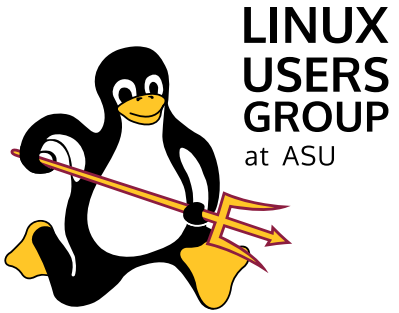


private.img



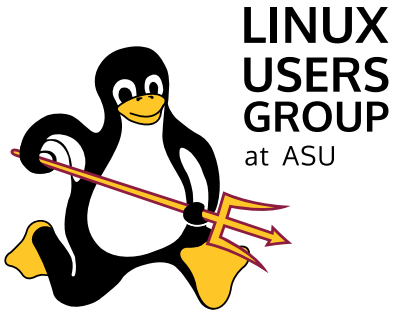
- Persistent storage for an AppVM.
- What is mounted in `/rw/`.
- Mounted in read-write.
- Contains:
  - `/home/` – Bind mounted on `/rw/home/`
  - `/usr/local/` – Symlinked to `/rw/usrlocal/`
  - Qubes configuration files in `/rw/config/`.

volatile.img

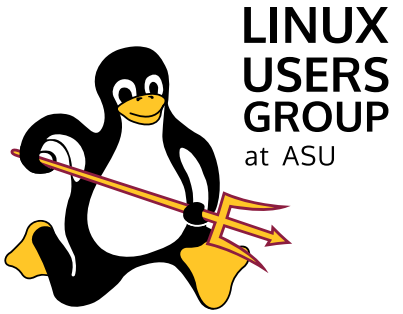


- Contains temporary changes.
- Discarded after AppVM shutdown.
- Contains two primary elements:
  - Changes to the root partition.
  - Swap partition.

TemporaryVMs

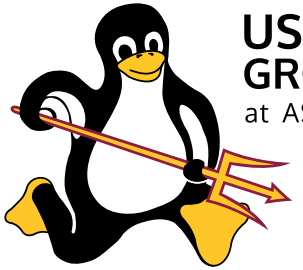


- TemporaryVMs are a unified version.
- Each have their own `root.img` and `private.img` which act in the same way except both are persistent.
- Still use a `volatile.img` for swap.



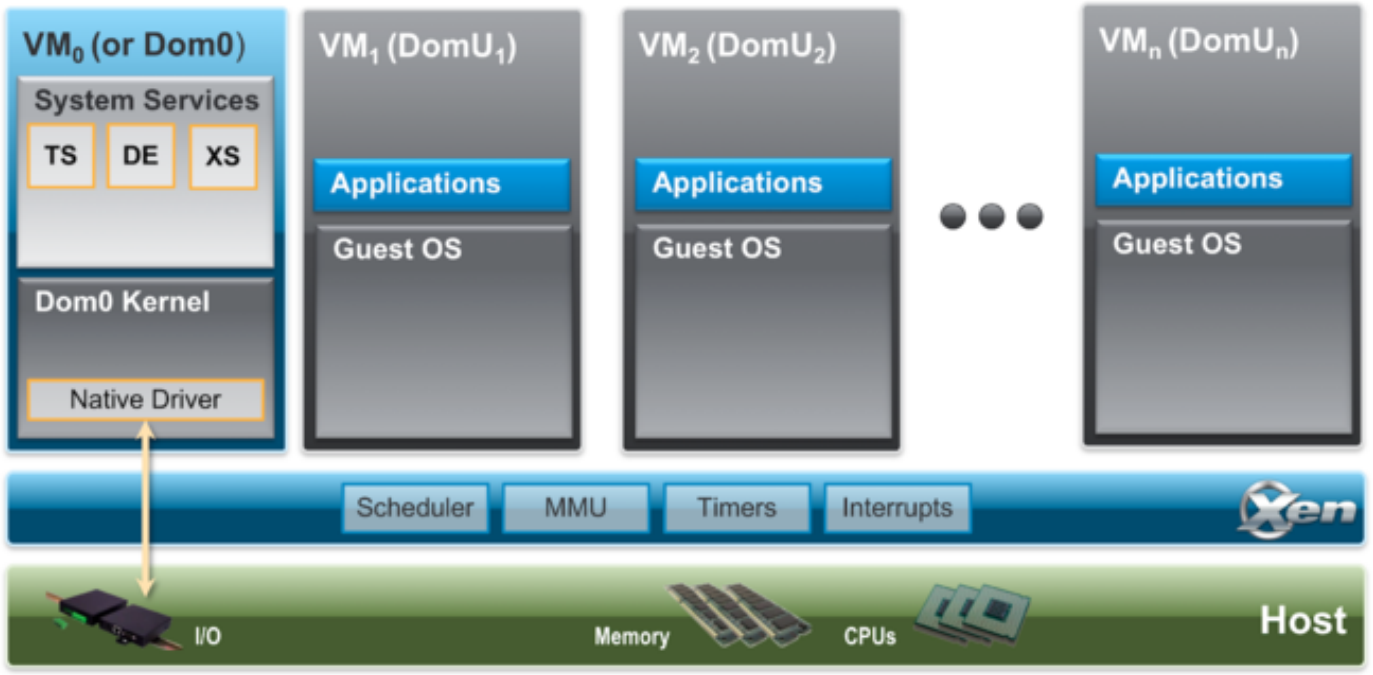
Section 4.2 – Virtualisation

Hypervisor

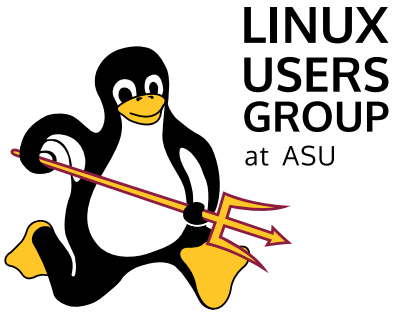


- Qubes uses the Xen Project Hypervisor.
- Xen is an open source type 1 hypervisor.
- Qubes architecture founded on the Xen Project Architecture

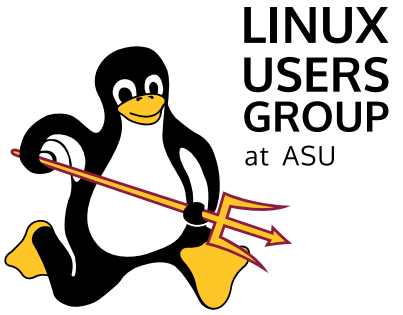
Xen Architecture



VM Virtualisation

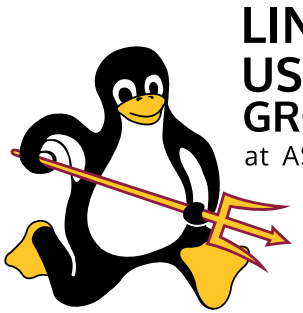


- Virtual Machines can be virtualised in three virtualisation types:
  - PVH (default)
  - HVM
  - PV

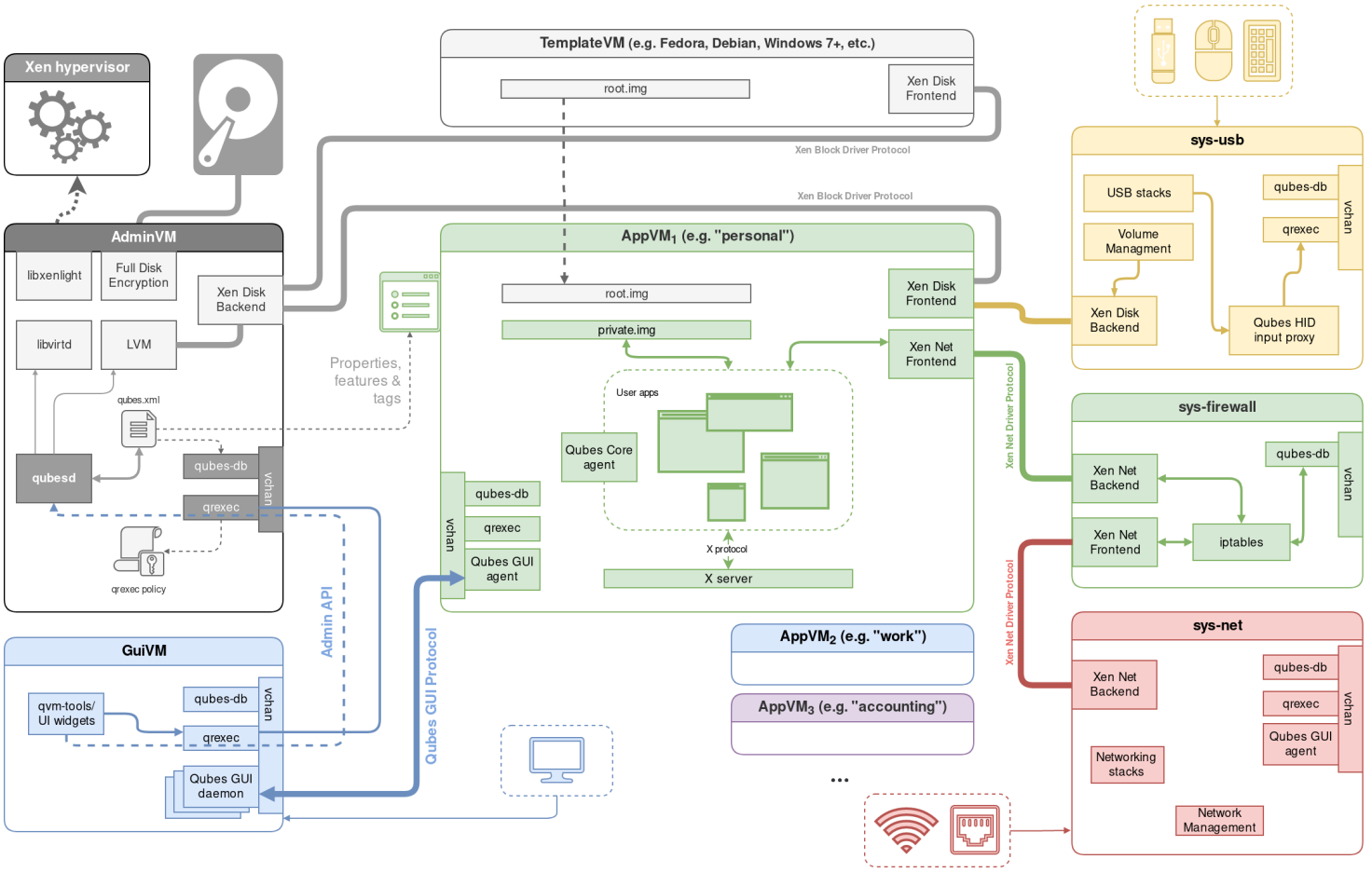


Section 4.3 – GUI

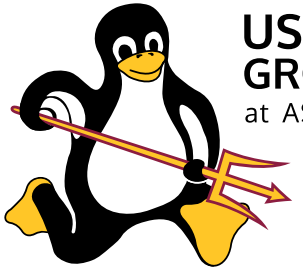
Architecture Overview



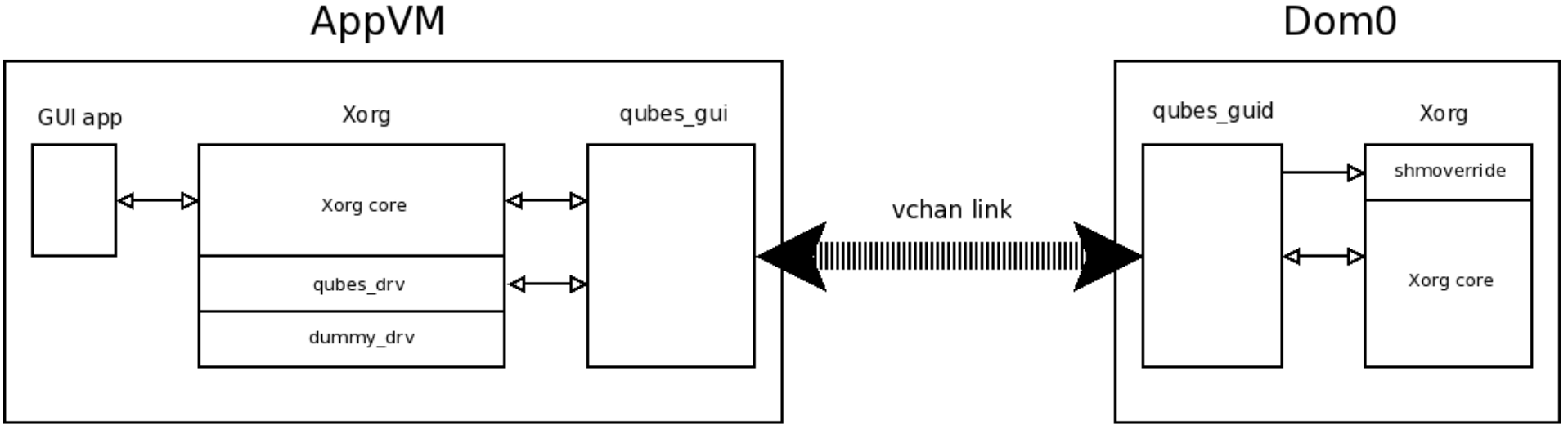
LINUX USERS GROUP at ASU



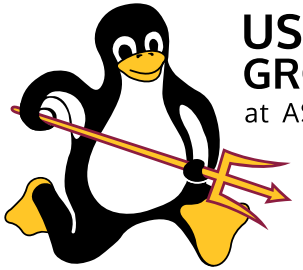
GUI Protocol Overview



LINUX USERS GROUP at ASU

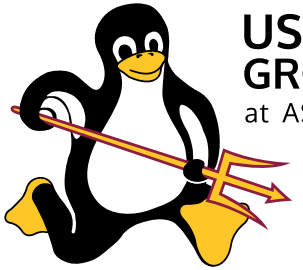


GUI Drivers



- “Hardware” Drivers
  - dummyqsb_drv – Video driver that paints onto the framebuffer in the RAM.
  - qubes_drv – Provides a vritual keyboard and mouse.
- Qubes Drivers
  - qubes_gui – Runs in the AppVM.
  - qubes_guid – Runs in dom0.

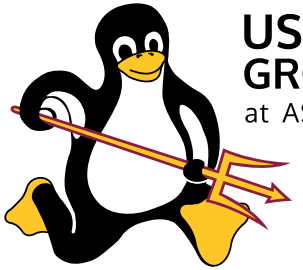
qubes_gui Driver



- Call XCompositeRedirectSubwindows on the root window, so that each window has its own composition buffer.
- Instruct the local Xorg server to notify it about window creation, configuration and damage events; pass information on these events to dom0.
- Receive information about keyboard and mouse events from dom0, tell qubes_drv to fake appropriate events.
- Receive information about window size/position change, apply them to the local window.

https://www.qubes-os.org/doc/gui/

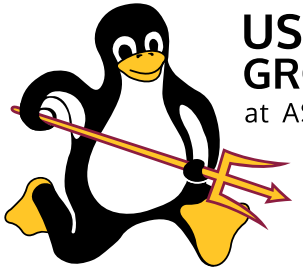
qubes_guid Driver



- Create a window in dom0 whenever an information on window creation in AppVM is received from qubes_gui.
- Whenever the local window receives XEvent, pass information on it to AppVM (particularly, mouse and keyboard data).
- Whenever AppVM signals damage event, tell local Xorg server to repaint a given window fragment.
- Receive information about window size/position change, apply them to the local window.

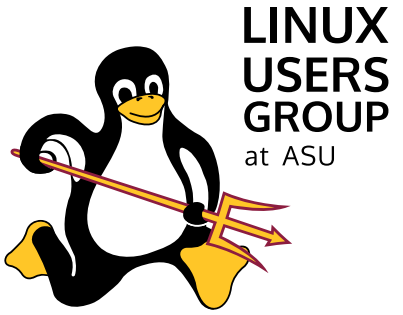
https://www.qubes-os.org/doc/gui/

Updating Windows



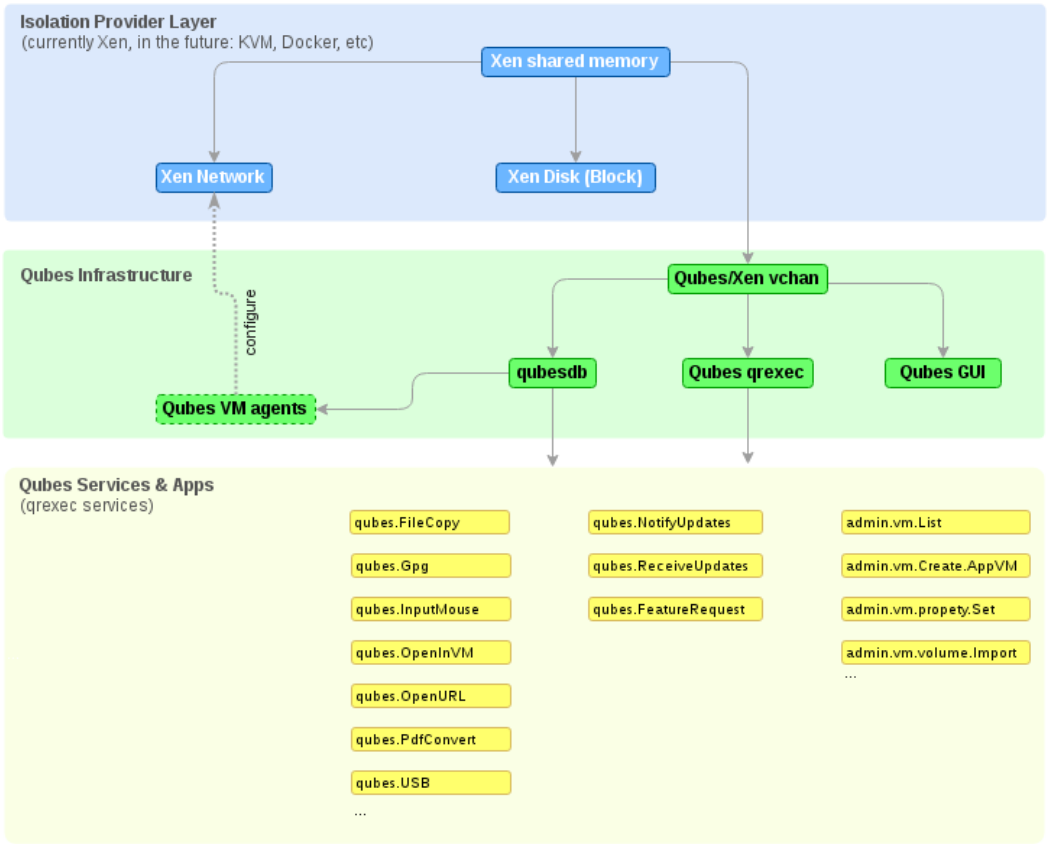
1. qubes_gui
    1. Asks the qubes_drv driver for a list of physical memory frames that hold the composition buffer of a window.
    2. Passes on this information to the qubes_guid driver in dom0 via a MFNDUMP message.
2. qubes_guid
    1. Xorg server is started with a LD_PRELOD-ed library named shmoverride.so which hooks all functions calls related to shared memory.
    2. qubes_guid creates a shared memory segment and then instructs Xorg to attach it via the MIT-SHM extension.
    3. When Xorg tries to attach the segment via glibc shmat, shmoverride.so intercepts the calls and instead maps AppVM memory via xc_foreign_pagers
    4. We can then use MIT-SHM functions to draw onto a dom0 window.

https://www.qubes-os.org/doc/gui/

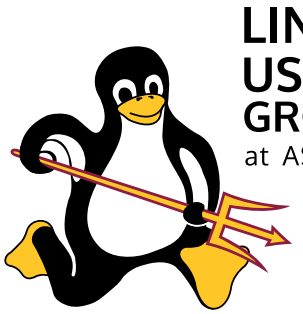


Section 4.4 – General Overview

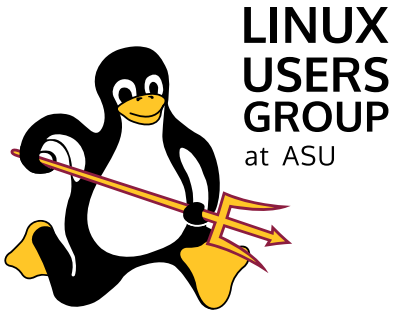
Overview



LINUX USERS GROUP at ASU

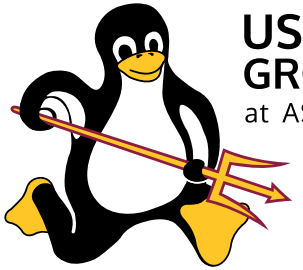


Section 5 – Security and Attack Surfaces



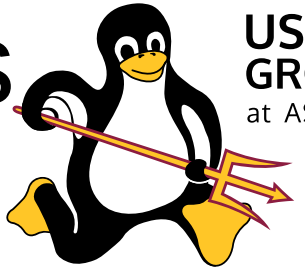
Section 5.1 – Project Security

Security Team



- Handles disclosed security issues.
- Evaluates the importance of security advisories of dependencies.
- Applying security patches.
- Publishing security bulletins and canaries.
- Managing the project’s PGP keys.

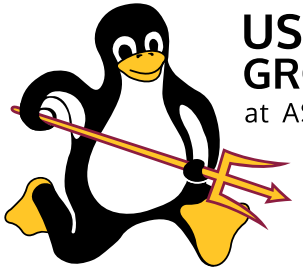
Security Team Members



LINUX USERS GROUP at ASU

- Marek Marczykowski-Górecki – Project Lead
- Simon Gaiser – Project Developer
- Joanna Rutkowska (canaries only) – Project founder

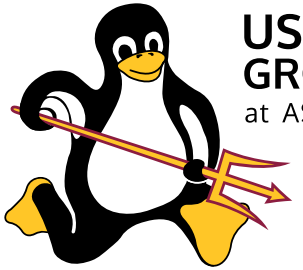
Signing and PGP Keys



LINUX USERS GROUP at ASU

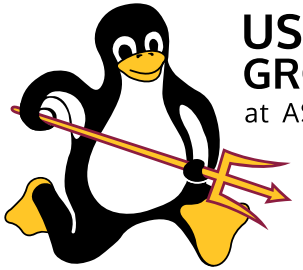
- Qubes has signing keys for each major release.
- Every Qubes team member’s PGP keys are published on the Qubes key site.
- Every member of the security team must sign a security release.

Canaries

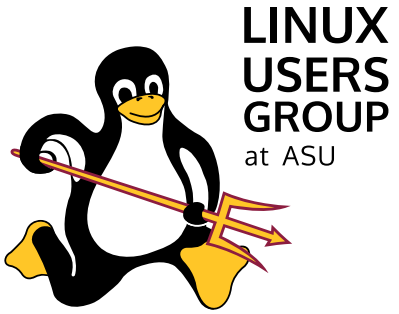


- Canaries are periodical security announcements by the Qubes security team.
- Requires every member sign them.
- Includes specific information such as the date, its number, when the next one plans to be, etc.
- Includes proof of time by news articles and the hash of a Bitcoin block mined in the last 24 hours before the release.

Security Bulletins

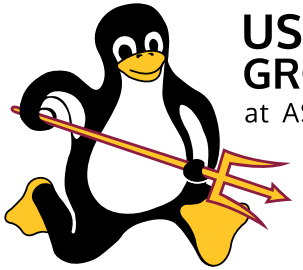


- Special announcement issued by the security team.
- Contains summary and analysis of a specific set of vulnerabilities and their patch.



Section 5.2 – Hardware Mitigation

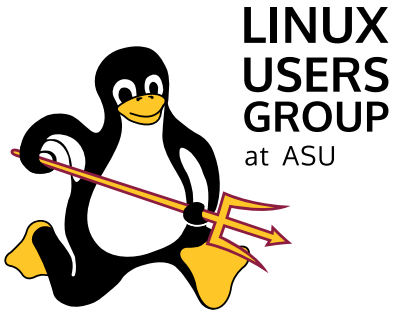
Disk Encryption and MFA



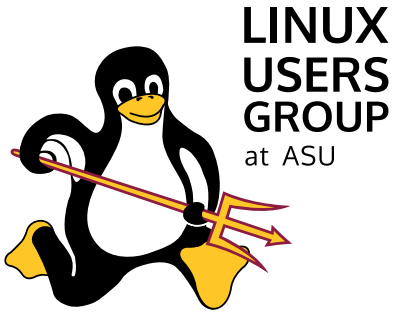
LINUX USERS GROUP at ASU

- LUKS 2 full disk encryption.
- U2F Authentication – Only an alternative
- Multi-factor Authentication – Only for restricting access to external services, not the login.

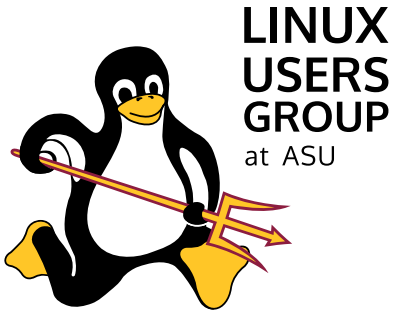
Anti Evil Maid



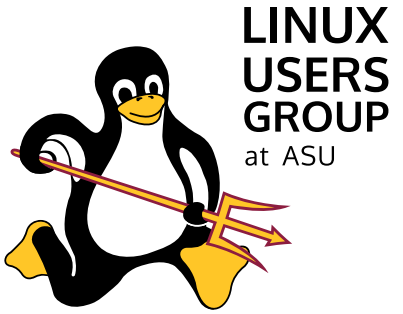
- External TPM trusted boot.
- If the whole chain is valid and properly executed, a message will be loaded validating it.
- Requires USB devices to be able to attach directly to dom0.



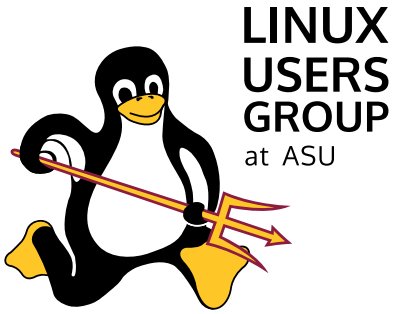
Section 5.3 – Attack Surfaces and Risk



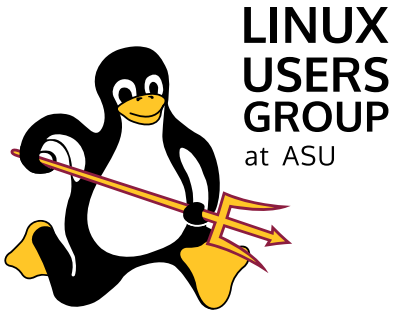
Physical Security of the Device



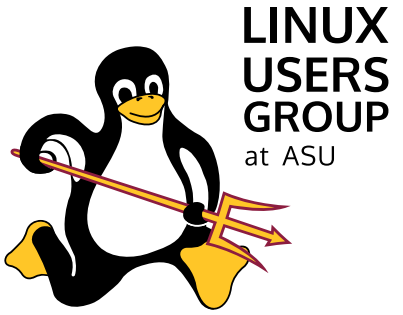
Hypervisor



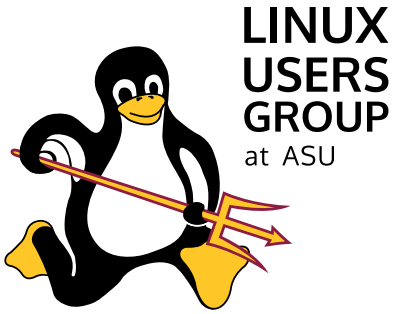
dom0 Drivers



Custom-built Malware

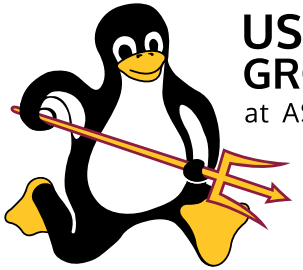


Real World Risk?



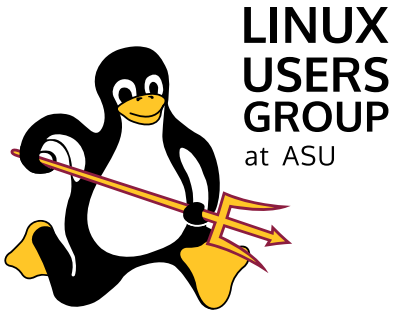
Conclusion

Check out the Project



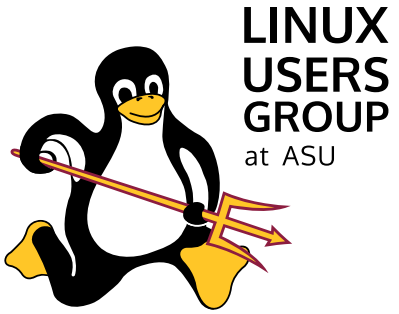
- QubesOS official website: https://www.qubes-os.org/
- Download QubesOS: https://www.qubes-os.org/downloads/
- QubesOS Forums: https://forum.qubes-os.org/
- QubesOS GitHub: https://github.com/QubesOS

Support the Project

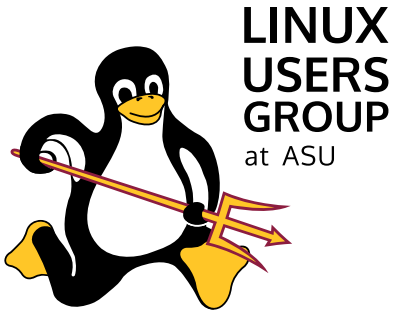


- QubesOS Donation: https://www.qubes-os.org/donate/
- Contribute: https://www.qubes-os.org/doc/contributing/

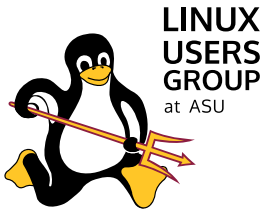
Resources



- Qubes Website: https://www.qubes-os.org/
- Qubes Documentation: https://www.qubes-os.org/doc/
- Qubes GitHub: https://github.com/QubesOS
- Xen Project: https://xenproject.org/
- Xen Project Wiki: https://wiki.xenproject.org/wiki/Main_Page
- Invisible Things Lab: https://invisiblethingslab.com
- Tarsnap website and documentation: https://www.tarsnap.com/



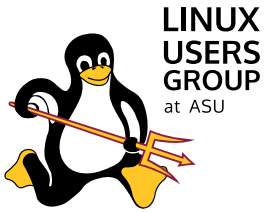
Questions?



Qubes OS
A Technical Introduction

Daylam Tayari

Technical?

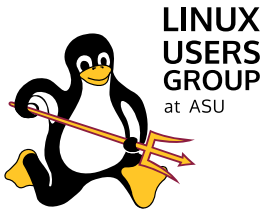


Don’t worry – This won’t be exclusively technical

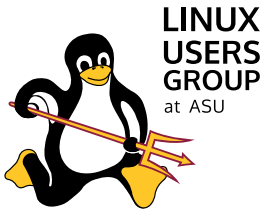
Overview → Specificities → Technical Details

Questions at any point?
Ask!

Talk Layout

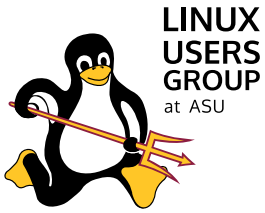


1. Overview
2. Usage and Security Implementations
3. Drivers, Interfaces and Policies
4. Technical Analysis of the Architecture
5. Security and Attack Surfaces



Section 1 – Overview

What is Qubes?



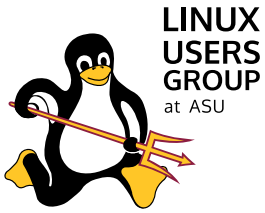
- An operating system based on a hypervisor.
- Everything compartementalised.
- Separation at every level – Hardware does not interact directly with the end usage (normally).
- All your regular actions are in VMs.
- Entirely open source.

Based on the Xen hypervisor.

Compartementalised at every level – The core, the hardware facing VMs, template and app VMs, disposable VMs.
Why normally? PCI devices can still be passed through.

All the work is done in the AppVMs.

History

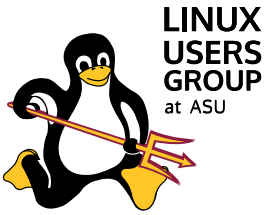


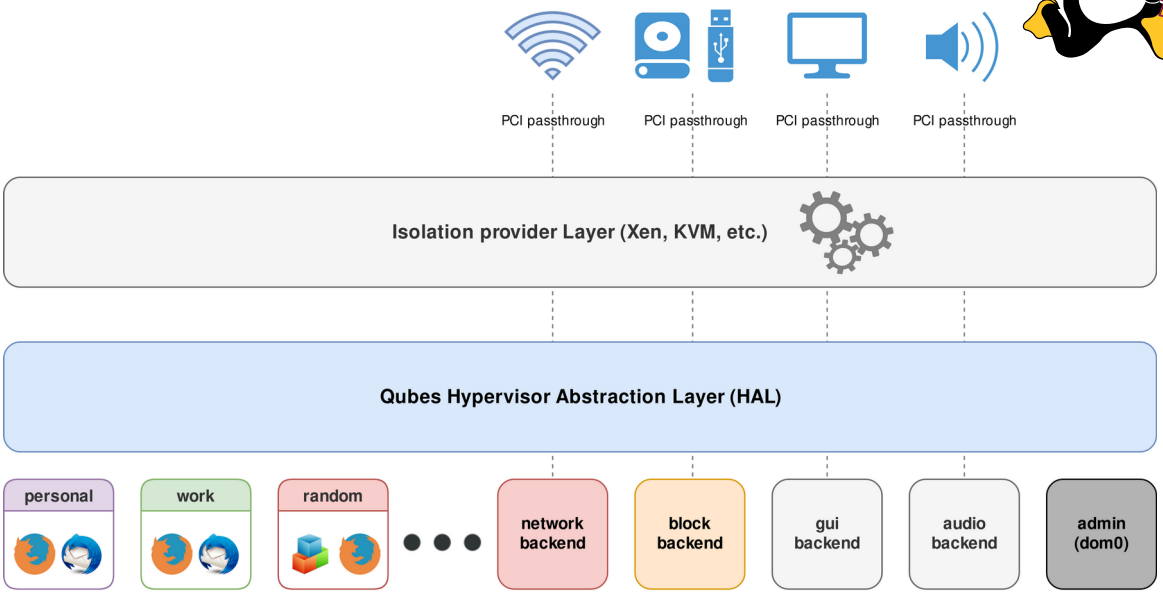
- Introduced in January 2010.
- Founded by Joanna Rutkowska.
- Compartementalised approach for security.
- Currently led by Marek Marczykowski-Górecki.
- Latest version is 4.1.0 released 4th of February 2021.

Rafal Wojtczuk was part of the initial release.
Through Invisible Things Lab.

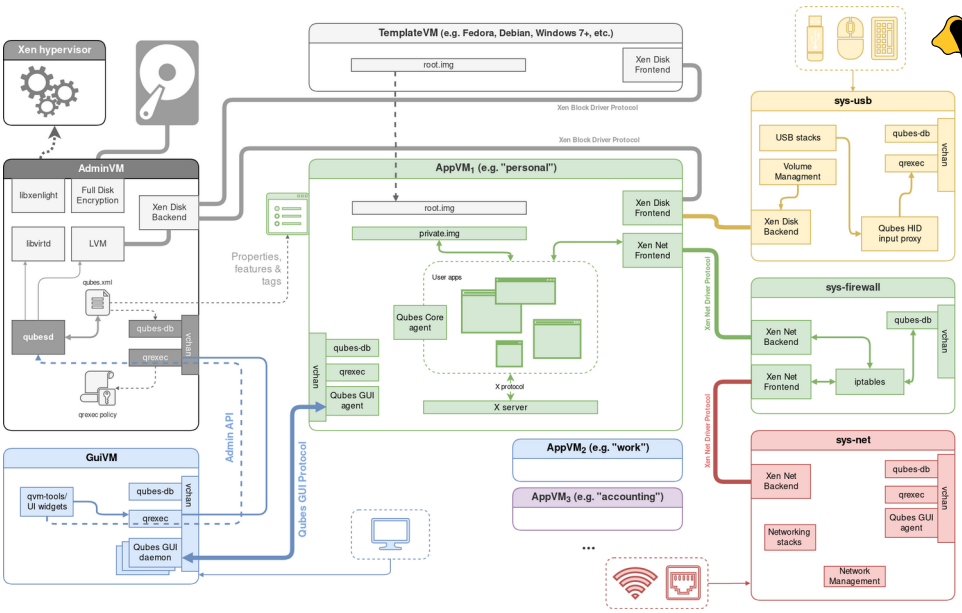
Work on it had started a couple months prior.

Architecture

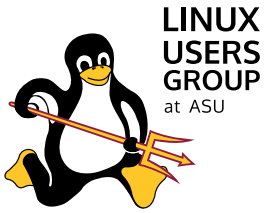




Core Stack


Xen hypervisor
TemplateVM (e.g. Fedora, Debian, Windows 7+, etc.)
root.img
Xen Disk Frontend
Xen Block Driver Protocol
AdminVM
libxenlight
Full Disk Encryption
Xen Disk Backend
libvirtd
LVM
qubes.xml
qubesd
qubes-db
qrexec
vchan
qrexec policy
Properties, features & tags
AppVM1 (e.g. "personal")
private.img
Xen Net Frontend
User apps
Qubes Core agent
Qubes GUI agent
X protocol
X server
AppVM2 (e.g. "work")
AppVM3 (e.g. "accounting")
sys-usb
USB stacks
Volume Managment
Qubes HID input proxy
sys-firewall
Xen Net Backend
iptables
sys-net
Networking stacks
Network Management
GuiVM
qvm-tools/ UI widgets
Qubes GUI daemon
Admin API
Qubes GUI Protocol
Xen Net Driver Protocol

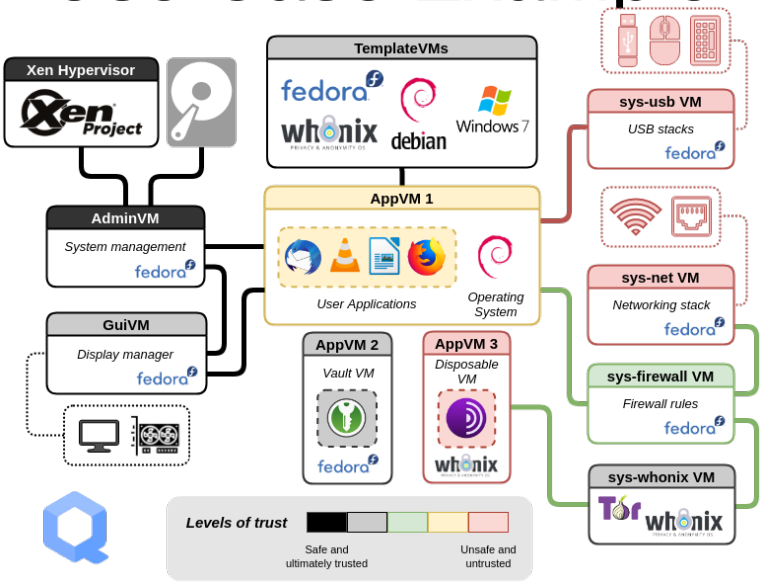
Working in AppVMs



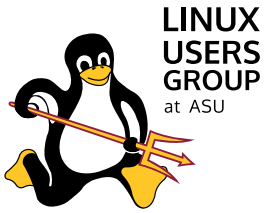
- AppVMs are where you do all your work.
- They can run everything – even Windows
- Connected to the hardware through other VMs.
- Interact with the core and other VMs with drivers.

Mention how I will often refer to VMs as a Qube or as Qubes.

Use Case Example



Why?



- Security by Isolation Approach
  - Virtualisation
  - Compartementalisation.
  - Need to Have approach.
  - Disposable VMs.
- Privacy
  - Whonix and Tor
  - Preventing Fingerprinting.
- Compartmentalisation of your Workspace.
- Changing Systems Often

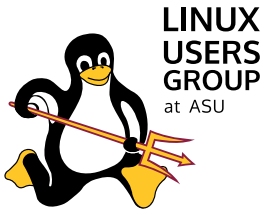
Need to have – Only give VMs what they need. Like need to know.

Fingerprinting – Just over 40,000 people use Qubes to any capacity. (fingerprinting based off update server connections: https://www.qubes-os.org/statistics/)

Like compartementalising everything (personal anecdote)

Changing systems often – Cross system migration with Qubes backups. Also practicality of temporary concealment (border crossing example – privacy rights, i.e. Australia but also the US).

Why Not?



- Technical experience occasionally required.
- Resources
  - Overhead
  - Sharing
  - Device Specifications
- Thinking in a Qubes way
- Technical Restrictions

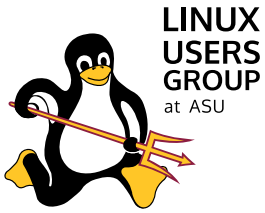
Technical experience includes being able to troubleshoot, know what something means to pinpoint the error and go from there.

Overhead around 4 Gb of RAM and very low CPU wise for Dom0.
Resources are shared between VMs but can also be limited. Also means each VM will have some level of overhead.

Requirements:
- 64 bit Intel or AMD processor with virtualisation.
  - Intel: VT-x with EPT and VTT-d
  - AMD: AMD-V with RVI and AMD-Vi/AMD IOMMU
  The latter is I/O MMU (Memory Management Unit) virtualisation which allows VMs to access PCI devices (PCI passthrough).
- 6Gb of RAM (16gb recommended)
- 32Gb of storage (128Gb recommended)

Thinking in the Qubes way – Thinking in the way of compartementalising everything not just using a central VM or two.
Technical restrictions – Screensharing, full size screenshots, USB devices can only be passthrough a single VM at a time (i.e. audio output or mic).
Certain customisation which include Dom0 can risk both security and be a technical challenge

Why do I?

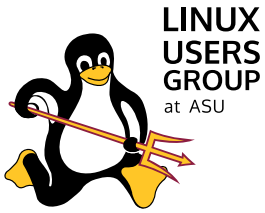


- Compartementalisation
- Security
  - Isolation
  - Need to Have
- Mirroring across devices
- Whonix

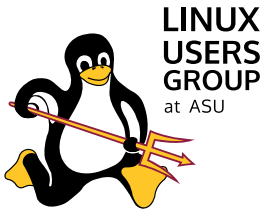
Always compartementalised growing up.

Using various operating systems for different uses.

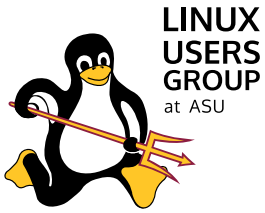
Compartementalisation also at the network level: Ethernet, WiFi, VPN, Tor, etc.



Section 2 – Usage and Security Implementations

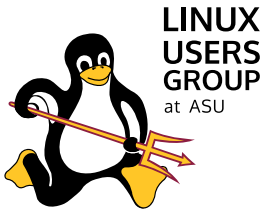


Section 2.1 – Virtual Machines in Qubes



Section 2.1.1 – Virtual Machine Types

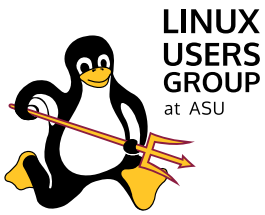
Types of VMs



- DispVMs – Based off template and no persistance.
- AppVMs – Based off a template and limited persistance.
- TemplateVMs – Root file system for AppVMs.
- StandaloneVMs – ‘Traditional’ VM.
- AdminVM – The core, dom0.

Everything is essentially a VM but which are for what.

DispVMs



- Disposable VMs are temporary disposable systems.
- Based off a template.
- Inherits the entire template including `/home/`.
- No persistance whatsoever, fully wiped on exit.
- Has a temporary volume though can be set to only use RAM.
- Files can be easily opened to be viewed.
- Randomly numbered between [1-10000).
- **Demo!**

Each DispVM uses volatile.img file that is used to create R/W illusion for its root fs via COW (same mechanism as for any other AppVM based on a template). This file is located in /var/lib/qubes/appvms/<name-of-dispvm-template/volatile.img.

(BTW, an interesting peculiarity, which you, like myself, might be wondering about: when another DispVM is started this volatile.img, used by the previous DispVM is unlinked, and so the new DispVM uses a new copy of the file, inode-wise).

Theoretically, one could modify the conf file used for dispvm creation, specifically this line (e.g. in /var/lib/qubes/appvms/fedora-18-x64-dvm/dvm.conf):
'script:file:/var/lib/qubes/appvms/fedora-18-x64-dvm/volatile.img,xvdc,w',

... and change the path to point to /dev/shm/qubes, where the DispVM savefile is kept, and which is stored in RAM entirely (the whole /dev/shm is a ram-based fs).

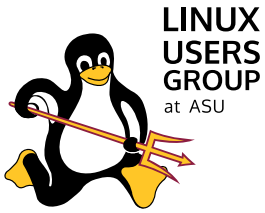
But, as Marek pointed out, a DispVM which might got to the dark side of the force, might then generate write to its whole root fs and make this vaoltile.img file to swallow up to 10GB in size, making the whole system unusable if we kept it in /dev/shm.

Unless we could setup some quote on /dev/shm, can we?

Anyway, regarding your forensics concerns -- a rather easy and convenient way to remove any records of a particular AppVM's activities is to simply shred its private and volatile images:

shred private.img volatile.img

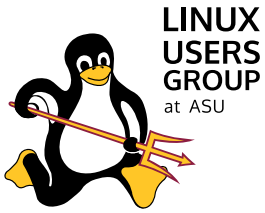
AppVMs



- Based off a template.
- Entire `/rw/` directory persists which includes:
  - `/home/`
  - `/usr/local/`
- Inherits part of its `/home` and `/usr/local`.
- Is provided an image of the template VM which was **generated at its last shutdown**.
- Where you will do all your work.
- **Demo!**

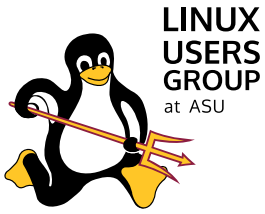
Inheritance from template includes the `/etc/skel` directory which points to `/home/` and the `/usr/local.orig/` direcotries in the template.

Template VMs



- Templates have no inheritance, recursive templates is not a possibility.
- Everything persists so **be careful**.
- No network by default.
- Its image is updated at shutdown.

Standalone VMs

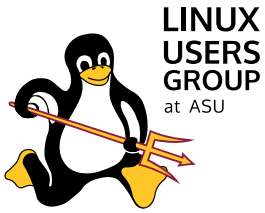


- Traditional VMs.
- Either independent clone of template or seperately installed system.
- No integration though possible with drivers.

You can install a VM using an ISO.

By adding the Qubes drivers to the system, it can have integration with the rest of the system.
Qubes Windows Tools.

AdminVM

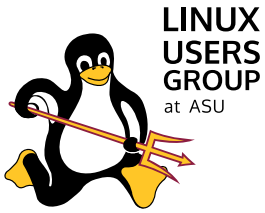


- AdminVM is dom0.
- Is the backbone of the operating system.
- Where the entire backend of the OS is ran.
- No network connectivity.
- Updates can only be made to specific repositories.
- Runs Fedora 32.
- Isolated from the rest of the system – No clipboard integration.

GUI VM is planned to be moved out into its seperate VM in a future release.

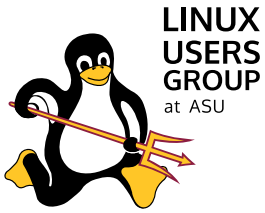
Explain logic behind Fedora 32.

GUI VM is in its own Qubes.



Section 2.2 – Template VMs

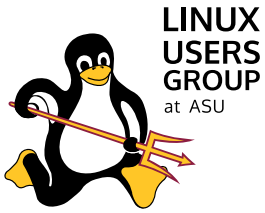
Template Repositories



- Official Templates – Built and released by the Qubes team.
- Community Templates – Designed, built and maintained by the community.

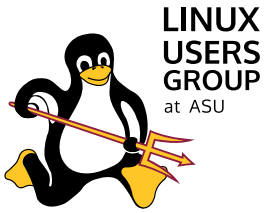
Community Templates are only those that are acepted and added to the community template repository, not just any.

Official Templates



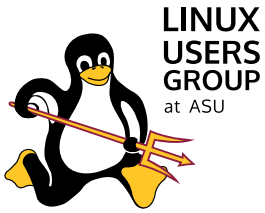
- Fedora 34
  - Minimal Template
  - Xfce Template using Xfce instead of GNOME.
- Debian 11
  - Minimal Template
- Whonix 16 Gateway and Workstation

Community Templates



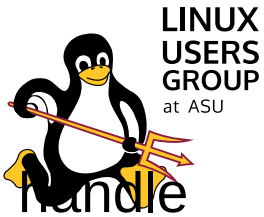
- Whonix
- Ubuntu
- Arch Linux
- CentOS
  - Minimal Template
- Gentoo
  - Minimal Template
- NetBSD

Custom Templates



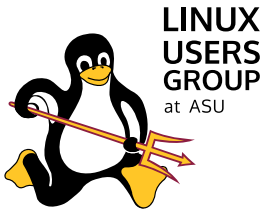
- Templates which aren’t standalone but can be implemented.
- Kali Linux
- Parrot

Updating



- Qubes has an updating utility which will handle proper updating for Templates.
  - Creates a mirror.
  - Updates.
  - If successful → Updates the actual Template
- Manually in the individual TemplateVM.

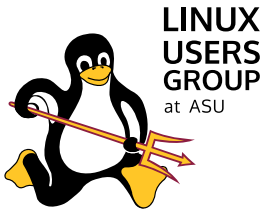
Installing Programs



- Using the package managers.
- Sending over binaries and repositories and installing them locally.
- Adding them manually in directories such as `/opt/` and creating respective desktop files.

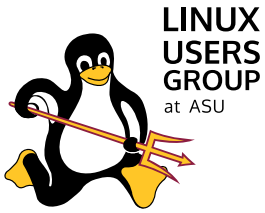
No internet access in Templates by default and they shouldn’t be granted it.

Whenever the contents of the elevated clipboard are retrieved and sent to a VM, they are wiped from the elevated clipboard.



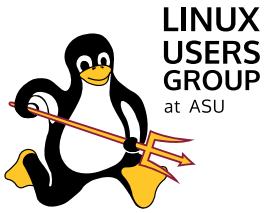
Section 2.3 – Migration and Backups

Cloning



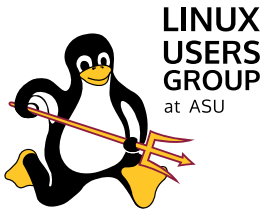
- Any VM, regardless of type can be easily cloned.
- Cloning a VM is accessible directly in the basic settings.

Qubes Backups



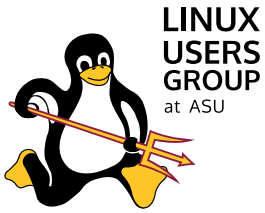
- Qubes official backup utility.
- Packages a set of VMs into a backup.
- Saves the version of the last shutdown.
- Saves specific file systems:
  - AppVMs – private.img
  - TemplateVMs – root.img
  - StandaloneVMs – root.img and private.img
- Encrypts backups with scrypt.

Backup Encryption



- Uses the scrypt encryption utility.
- Initially developed for the Tarsnap online backup system.
- Uses a complex cryptographic system including both signing and encryption, both which are required for decryption.
- Qubes uses their release signing key for the signing.

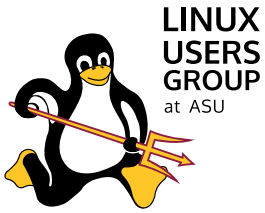
Tarsnap Cryptography



- A 2048-bit RSA key used for signing archives. This is used in combination with SHA256 and a Merkle hash tree to verify the authenticity of stored archives.
  - Cryptographers: RSASSA-PSS is used, with SHA256 as the hash function.
- A 2048-bit RSA key used for encrypting session keys. All the data which the Tarsnap client sends to the server to store is encrypted with per-archive random AES-256 keys; those keys are encrypted with this RSA key and attached to the stored data.
  - Cryptographers: RSAES-OAEP is used to encrypt the session keys, using MGF1 and SHA256, and the padding verification performed when decrypting is carefully written to be free of timing side channels. The AES-256 keys are used in CTR mode, with sequentially incrementing nonces — generating a new AES-256 key for each session ensures that a key-nonce pair will never be used twice.
- A 256-bit HMAC-SHA256 key used to protect each individual block of data from tampering. From a cryptographic perspective, this is unnecessary, since a Merkle hash tree protects each archive; but data is compressed using zlib before being stored, so this provides protection against a theoretical attacker who can tamper with stored data and has found a security flaw in zlib decoding.
  - The zlib decoding process is quite complex — for that matter, the same is true of all decompression algorithms — and such complex code has a relatively large risk of having security vulnerabilities, hence the decision to add an extra layer of security here.
- Two 256-bit HMAC-SHA256 keys used to generate names for blocks of data stored. Tarsnap uses the same reference-by-hash trick as the author's Portsnap and FreeBSD Update utilities; using HMACs instead of raw SHA256 hashes prevents any information from leaking via the hashes.
  - Why two keys? One is used to hash data, and the other is used to hash archive names. Yes, that's right, even the names of archives are stored securely!

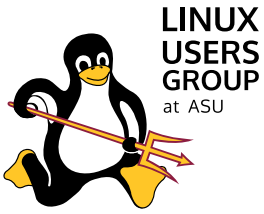
https://www.tarsnap.com/crypto.html

Backup Restore

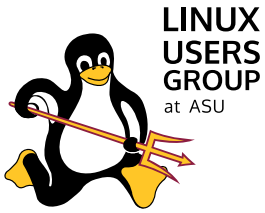


- Official Qubes Restore utility.
- Restores backups without a hitch.
- Has the possibility to solely verify integrity.
- Can either restore alongside or overwrite.
- Restoration has no difference machine to machine as long as they are on the same Qubes release.

Migration to an ISO



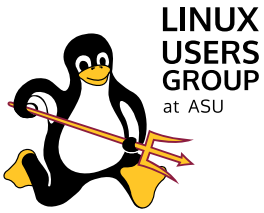
- Official support with Qubes Builder
- Allows you to turn a Qubes image into an ISO
- Theoretically should be flashable onto another computer just like any ISO.



Section 2.4 – Interactions between VMs

This is a short overview of the various interfaces and drivers.
We will go into further detail in section 3.

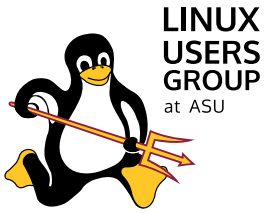
Clipboard Interactions



- A single elevated clipboard.
- Any VM can send their clipboard to the elevated clipboard with `Ctrl+Shift+c`.
- Any VM can retrieve the contents of the elevated clipboard with `Ctrl+Shift+v`.
- Elevated clipboard is wiped on retrieval.
- AdminVM is not integrated with the elevated clipboard.

Explain why dom0 pasting is prohibited. Security.

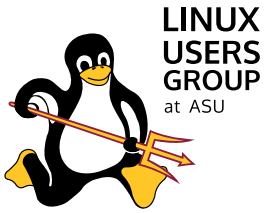
File Interactions



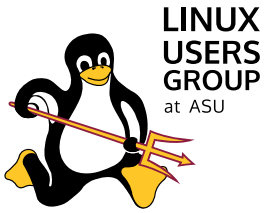
- All VMs can send files to another VM.
- All VMs except the AdminVM can receive files.
- Files will arrive in the `~/QubesIncoming/VM_NAME/` folder.
- AdminVM can only receive files from other VMs by passing the IO from the VM to the AdminVM.
  - i.e. `qvm-run --pass-io QUBE_NAME 'cat NAME_OF_FILE ' > NAME_IN_DOM0`

Explain why dom0 passing is so complicated.

Split Utilities



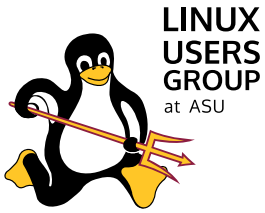
- Split GPG
- Split SSH
- Split Electrum Wallet
- Split dm-crypt
- U2F Proxy



Section 2.5 – Networking

This is a short overview of the various interfaces and drivers.
We will go into further detail in section 3.

Networking Overview

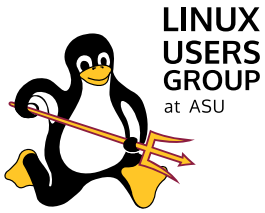


- Any VM can be set to be a networking VM.
- There are two primary VMs responsible:
  - Hardware VM (i.e. WiFi, Ethernet, etc.)
  - Firewall VM
- Other VMs can be configured to handle and adapt a network connection.
- Network VMs can be chained.
- Changing networks is as simple as changing one setting and is applied instantly.
- You select the final VM as the one you want to use.

Highlight the chained part and elaborate a bit on it.

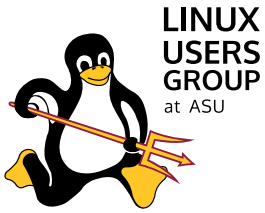
Explain the final selection and bring up the VPN example.

Hardware VMs



- Hardware networking VMs are those that interact with the actual PCI devices.
- Good practice is one per PCI device.
- Lets everything through.
- Uses NetworkManager by default.

Firewall VM

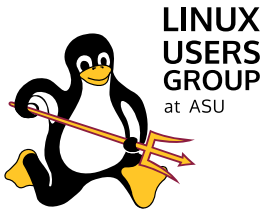


- Firewall rules can be set for each Qube.
- Applies all network-level policies.
- Responsible for blocking inter-VM networking.

Firewall policy can be from allow all/allow none to allow only specific whitelist IPs and only to specific ports and protocols.

Inter-VM networking can be blocked.

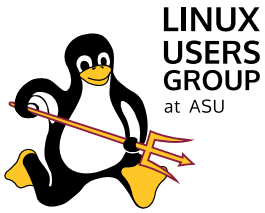
Custom VMs



- No limit in the amount of network VMs that can be created and made.
- You can create a VPN VM and everything that is passed through goes through the VPN.
- You can deploy a full on DNS sinkhole (i.e. Pi-hole) or a firewall if you wanted and chain it.

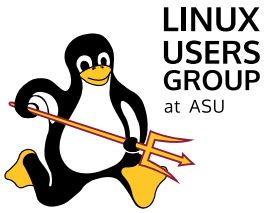
Bring up the chaining aspect and chaining through multiple VMs.

Whonix Networking



- sys-whonix VM running Whonix Gateway.
- All connection passed through it are passed through Tor.
- Tor connection is fully configurable.
- Just like any other VM.

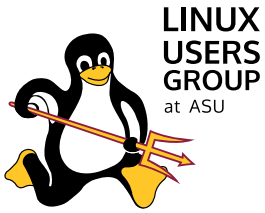
Mention creating multiple sys-whonix for multiple connections.



Section 2.6 – USB Handling

This is a short overview of the various interfaces and drivers.
We will go into further detail in section 3.

USB Qubes

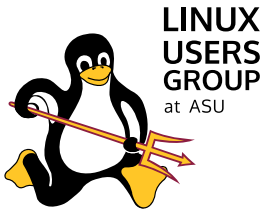


- Separate VM which interacts directly with the USB PCI device.
- Without a USB Qube, the USB PCI device interacts directly with dom0.
- To use or have access to a USB device in a VM, it HAS to be manually passed throught the Qubes Devices taskbar menu.
- **Demo!**

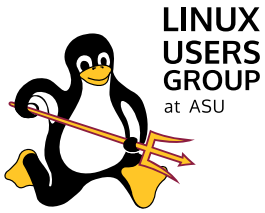
Passing as a USB keyboard security risk.

Camera example.

Complications



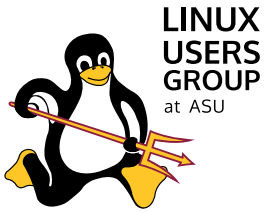
- Necessary passthrough for login
  - Passthrough of USB keyboard device to dom0 on boot through a policy.
- Connection not as stable as if directly passed through.



Section 2.7 – VM Configuration

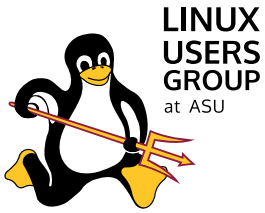
This is a short overview of the various interfaces and drivers.
We will go into further detail in section 3.

Basic Settings



- Name and label colour
- Template
- Networking VM
- Private storage size
- Start on boot
- Include in backups
- Debug mode

Advanced Settings



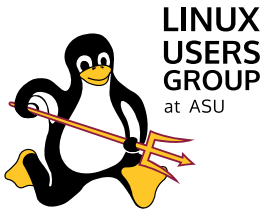
- Initial Memory
- Max Memory (only if memory balancing is enabled)
- VCPUs Number
- Kernel
- Virtualisation
- Disposable VM Template (if not itself)
- Provides Network Toggle
- Boot from ISO

Memory balancing is also preseent

Kernel Options:s
- 5.4.136
- 5.4.143
- 5.4.156
- 5.10 (Qubes 4.1)

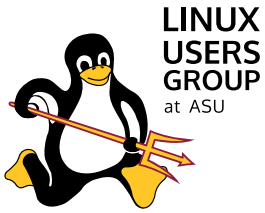
Virtualisation options:
- PVH
- HVM
- PV

Firewall Rules



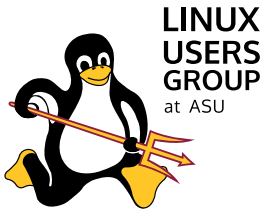
- Configure firewall rules.
- Configurable by:
  - Address
  - Port
  - Protocol
- Temporary Access

Devices



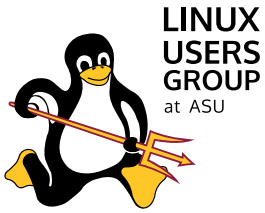
- PCI Passthrough
- Must use HVM or PV virtualisation and have disabled memory balancing for ideal results.
- Limit in types of PCI device that can be passed depending on the VM type.

Applications

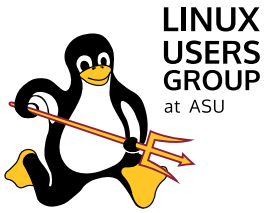


- Select which applications show up in the application menu for that VM.
- Grabbed from all `.desktop` files in the `/usr/share/applications/` directory.
- Refreshed from Template VM.

Services



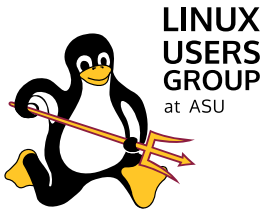
- Add or remove specific services.
- Enable or disable specific added services.



Section 2.8 – Customisation

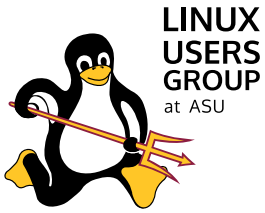
This is a short overview of the various interfaces and drivers.
We will go into further detail in section 3.

DEs and WMs



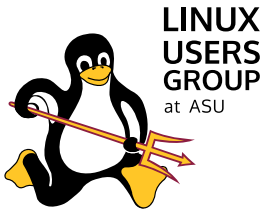
- Desktop Environments officially supported:
  - Xfce (default)
  - KDE
- Window Managers (community support):
  - i3
  - AwesomeWM
  - bspwm (unofficial)

Security when Ricing

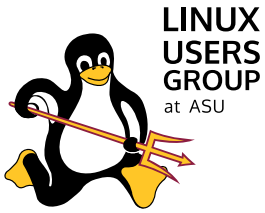


- Passing programs to dom0
- Affecting dom0
- Restriction in availability of programs.

You can’t go around and add any old program to your dom0.
Also maintaining them and security risk.



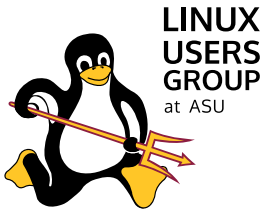
Section 3 – Drivers, Interfaces and Policies



Section 3.1 – Intercommunication Drivers

This is a short overview of the various interfaces and drivers.
We will go into further detail in section 3.

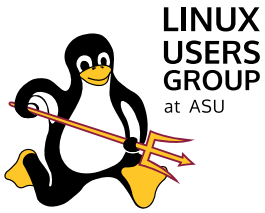
Clipboard



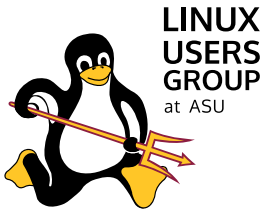
- A single elevated clipboard.
- Any VM except dom0 can copy and paste from it.
- It can only store a maximum of one entry.
- When it is pasted in a VM, it is wiped.

Can also be manually cleared.

File Transfer



- Files can be transferred between all VMs except dom0.
- Files will be sent to the `~/QubesIncoming/VM_NAME/` directory.

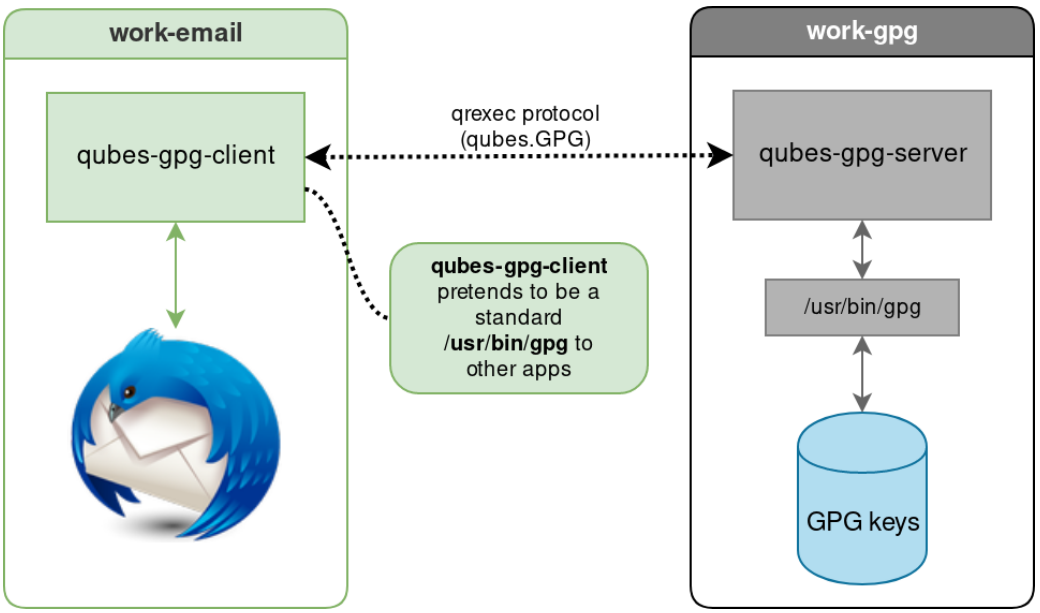


Section 3.2 – Split Interfaces

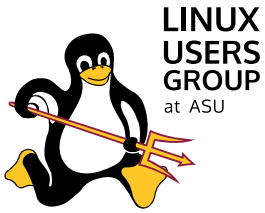
This is a short overview of the various interfaces and drivers.
We will go into further detail in section 3.

Split GPG - 1

LINUX USERS GROUP at ASU



Split GPG - 2



- All keys are stored in a seperate VM which is ideally fully isolated.
- The qubes-gpg-client is what interacts with the client, passing all the commands over to the GPG program on the GPG Qube and sending its output back.
- Two sets of prompts appear at each GPG usage.
- Access can be determined to a set amount of seconds.
- Passwords are entered and validated directly in the GPG Qube.
- You set your GPG client to be the Qubes client in programs.
- **Demo!**

Isolation = no internet connection or devices or anything ever passed to it (such as USB devices)

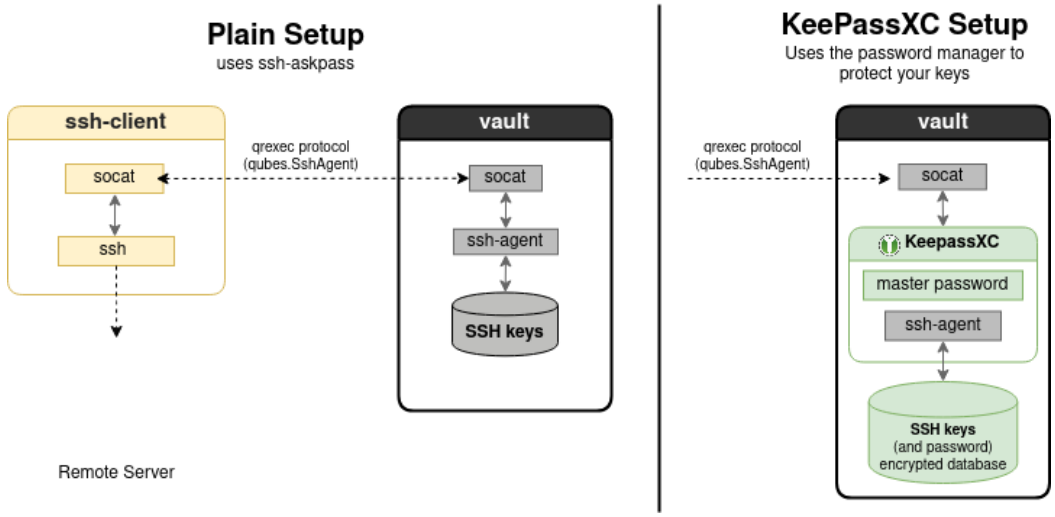
The policy is determined in the `/rw/config/gpg-split-domain` file.

The password for PGP key never touches an AppVM.

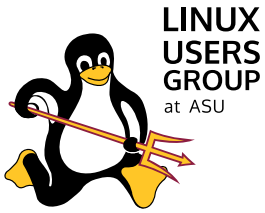
Example for like Thunderbird and Git.

Split SSH - 1

LINUX USERS GROUP at ASU



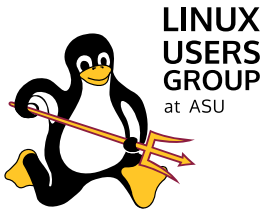
Split SSH - 2



- SSH keys safe in an isolated VM.
- SSH keys are retrieved and passed through when requested.
- SSH private keys are kept in the isolated VM.
- Unfortunately, use can be held on for the entire of the existence of the VM.
- Community made.

SSH agent connection, once done, can be kept active and held on for the entirety of the time the VM stays up which is a big liability and my personal reason why I don’t use it.

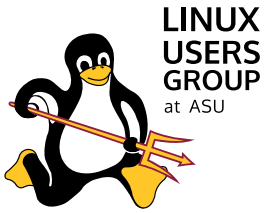
Spit dm-crypt



- Splits up the process of decryption of a LUKS1 partition.
- Encrypted VM is attached to a specifically configured offline disposable VM.
- Prompts for a password and decrypts the partition in the disposable VM.
- Mounts the partition into the destination VM.
- The destination VM does not have access to the password or encryption key.

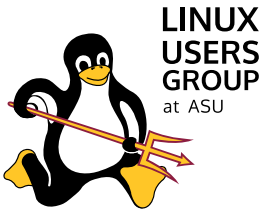
The encrypted partition is attached from the source VM to an offline device DisposableVM configured not to parse its content in any way: The kernel partition scanners, udev probes, and UDisks handling are disabled.

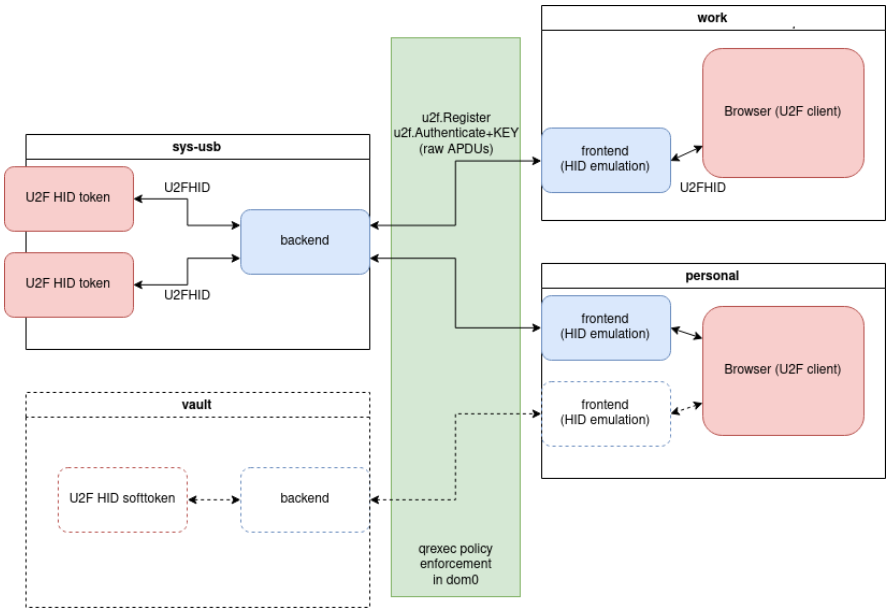
Split Electrum Wallet



- Wallet is stored in one isolated offline VM.
- Wallet is accessed via an online watching-only Electrum wallet which connects to the isolated wallet.

U2F Proxy - 1

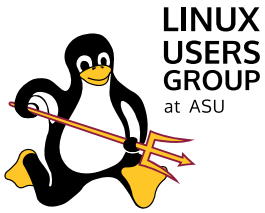




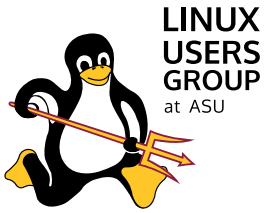
Dashed lines signify upcoming enhancements.

Soft tokens to emulate a smart card.

U2F Proxy - 2



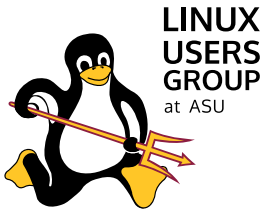
- The proxy only passes the necessary to the actual U2F key, performs the transaction and returns it to the browser without passing the entire U2F key to the device.
- Can be set to only allow specific tokens per VM.
- No direct interaction between the U2F device and the VM that is requesting the token.



Section 3.4 – Policies

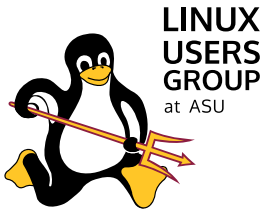
This is a short overview of the various interfaces and drivers.
We will go into further detail in section 3.

Policies in Qubes



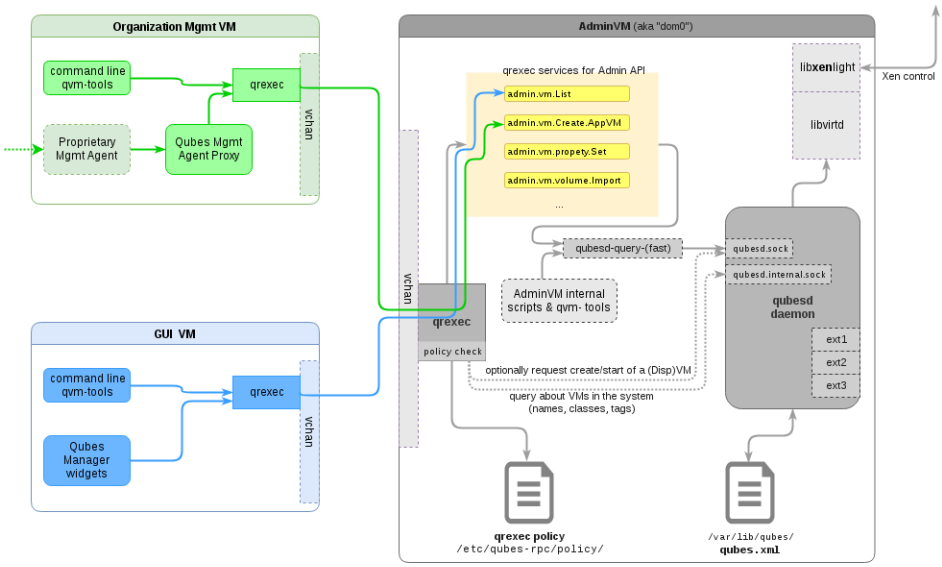
- Each interaction with a driver is controlled by a policy.
- Policies can be set both inside of a VM for its drivers or for the system.
  - `/rw/config/`
  - `/etc/qubes-rpc/policy/` in dom0

Control of Policies

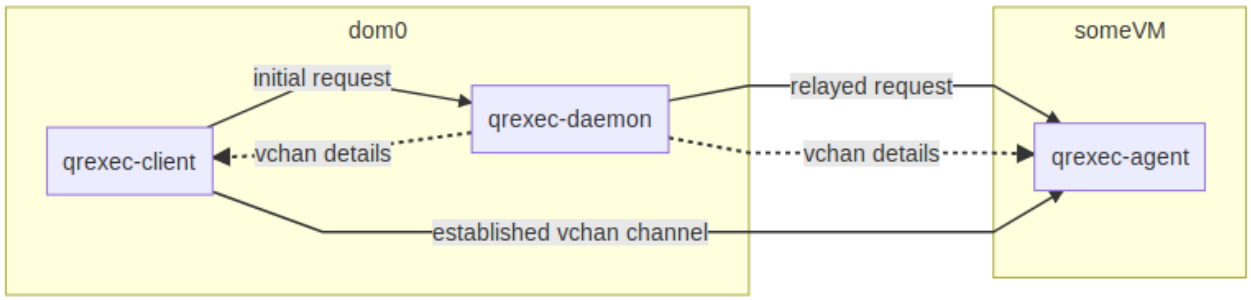


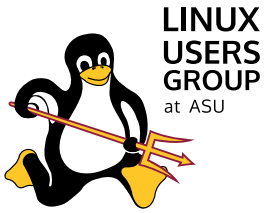
- Which VMs does the policy apply to.
- How to handle any VM not explicitly referred to.
- Handle policies depending on the tag of a VM.
- Allow or deny specific actions.

Policy Overview

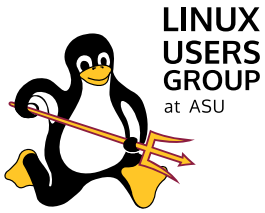


Policy Client Overview



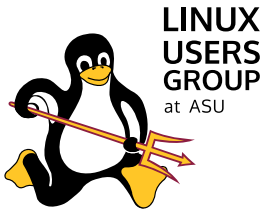


Section 4 – Technical Analysis of the Architecture



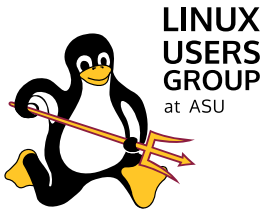
Section 4.1 – Volume System

Volume Types



- root.img – Root device (`/`).
- private.img – Persistant volume of a VM.
- volatile.img – Swap and temporary `/`. Discarded on reboot.
- modules.img – Kernel modules and firmware.

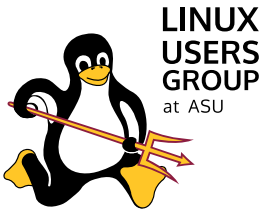
Block Devices



- xvda – Base root (**NOT** root.img)
- xvdb – private.img
- xvdc – volatile.img
- xvdd – modules.img

xvda actually represents a snapshot of the root.img volume, not a direct link to it which is why the differentiation.

Root Implementation - 1



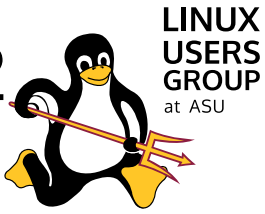
- Two versions of `root.img`:
  - `root.img` – True root filesystem.
  - `root-cow.img` – Snapshot of the root filesystem.
- Everytime an AppVM is started, a `root-cow.img` is made (unless no changes).
- AppVMs are based off the `root-cow.img`.

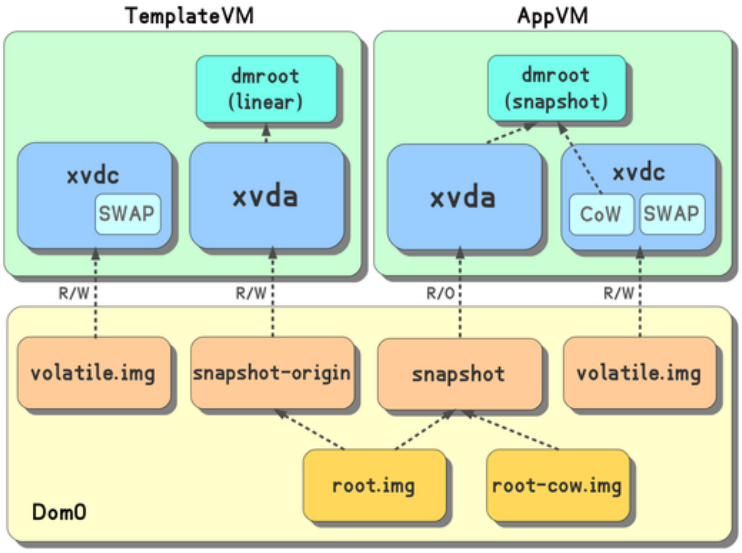
CoW stands for Copy on Write

Snapshot made at the start.

If root.img hasn’t changed, all the AppVMs will be based on the same template.

Root Implementation - 2





When a TemplateVM starts, its provided with the snapshot-orig which it then writes to the root.img during shutdown.

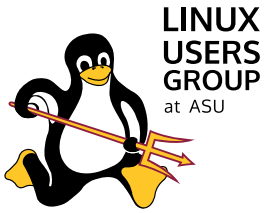
Uses the `qvm-template-commit` tool.

A Xen script moves the `root-cow.img` volume to `.old` and creates a new one with a new inode number.

(AppVMs reference to it by inode so no issues and the old root-cow is deleted when the AppVM shuts down).

Reverting to a past root.img IS POSSIBLE.

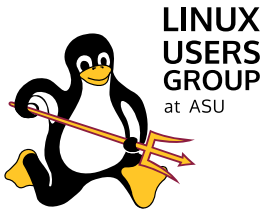
private.img



- Persistent storage for an AppVM.
- What is mounted in `/rw/`.
- Mounted in read-write.
- Contains:
  - `/home/` – Bind mounted on `/rw/home/`
  - `/usr/local/` – Symlinked to `/rw/usrlocal/
  - Qubes configuration files in `/rw/config/`.

Qubes configuration files such as the PGP.
Show/demonstrate.

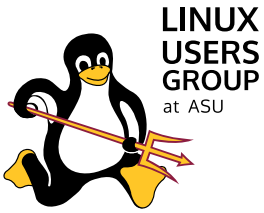
volatile.img



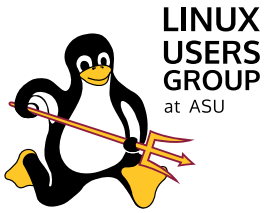
- Contains temporary changes.
- Discarded after AppVM shutdown.
- Contains two primary elements:
  - Changes to the root partition.
  - Swap partition.

Changes to the root partition such as installing programs.

TemporaryVMs

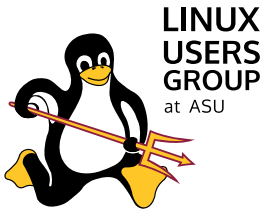


- TemporaryVMs are a unified version.
- Each have their own `root.img` and `private.img` which act in the same way except both are persistent.
- Still use a `volatile.img` for swap.



Section 4.2 – Virtualisation

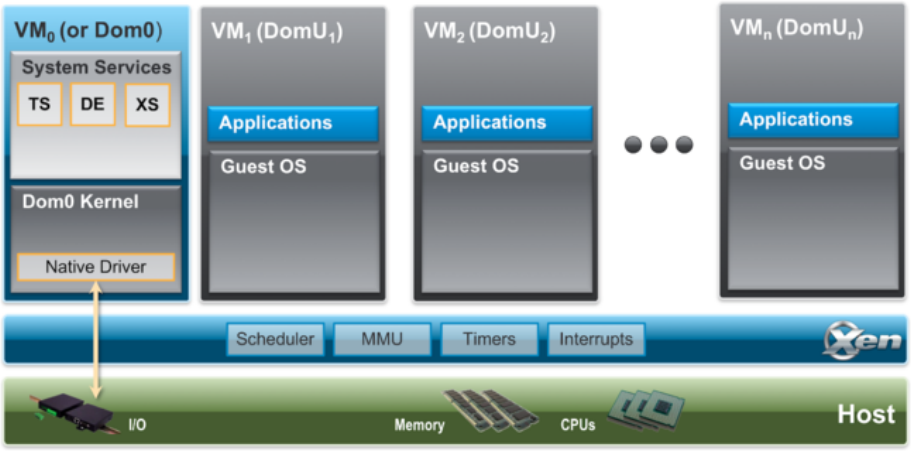
Hypervisor



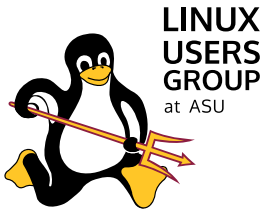
- Qubes uses the Xen Project Hypervisor.
- Xen is an open source type 1 hypervisor.
- Qubes architecture founded on the Xen Project Architecture

Xen Architecture

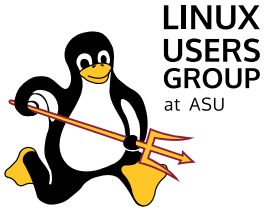
LINUX USERS GROUP at ASU



VM Virtualisation



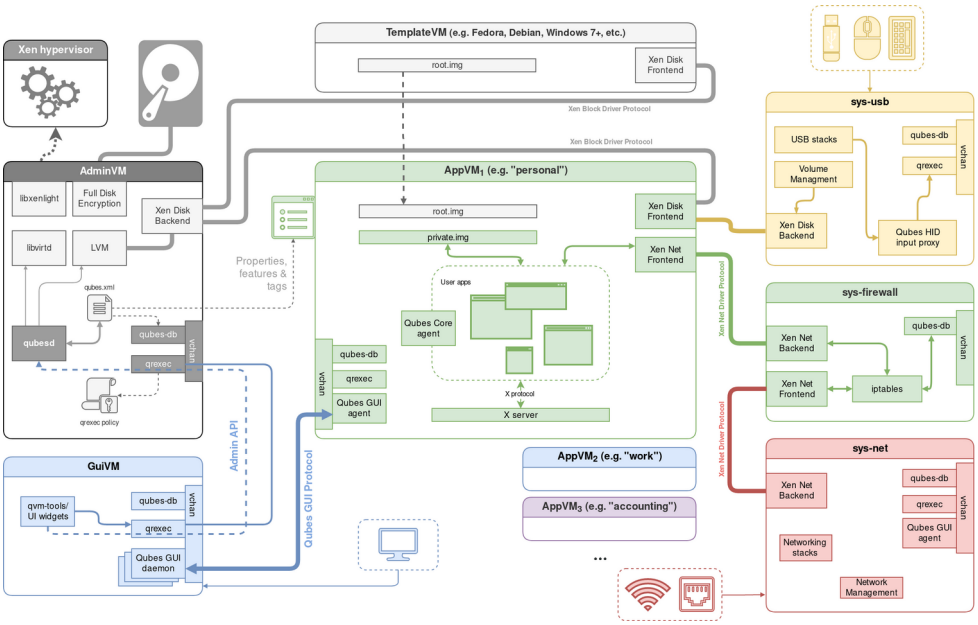
- Virtual Machines can be virtualised in three virtualisation types:
  - PVH (default)
  - HVM
  - PV



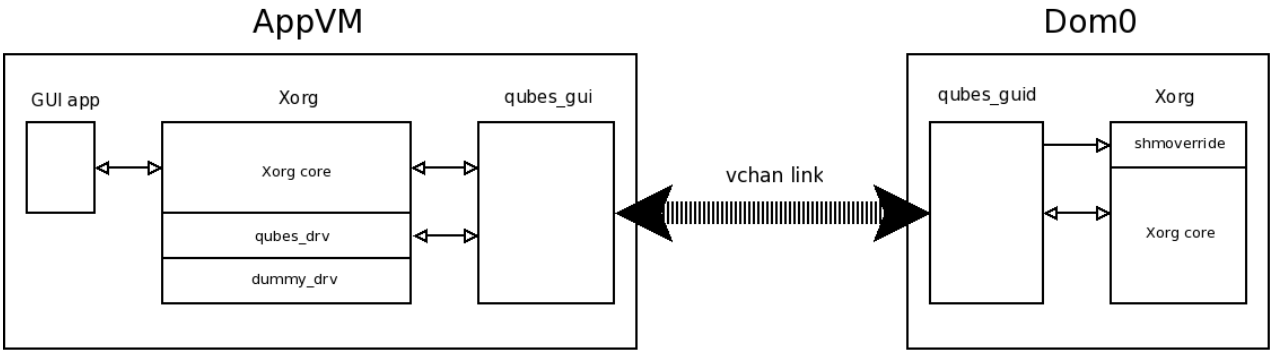
Section 4.3 – GUI

Architecture Overview

LINUX USERS GROUP at ASU



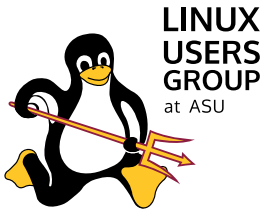
GUI Protocol Overview



The vchan link is the integration into Qubes of the Xen vchan protocol.

The vchan protocol is used to stream data between virtual machines without needing any locks.

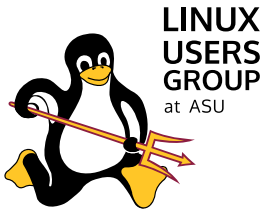
GUI Drivers



- “Hardware” Drivers
  - dummyqsb_drv – Video driver that paints onto the framebuffer in the RAM.
  - qubes_drv – Provides a vritual keyboard and mouse.
- Qubes Drivers
  - qubes_gui – Runs in the AppVM.
  - qubes_guid – Runs in dom0.

Hardware drivers arent actually connected to any real hardware.

qubes_gui Driver

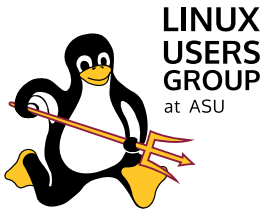


- Call XCompositeRedirectSubwindows on the root window, so that each window has its own composition buffer.
- Instruct the local Xorg server to notify it about window creation, configuration and damage events; pass information on these events to dom0.
- Receive information about keyboard and mouse events from dom0, tell qubes_drv to fake appropriate events.
- Receive information about window size/position change, apply them to the local window.

https://www.qubes-os.org/doc/gui/

This is all of its responsibilities.

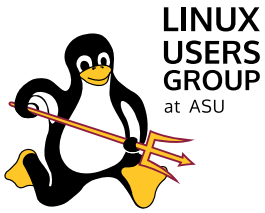
qubes_guid Driver



- Create a window in dom0 whenever an information on window creation in AppVM is received from qubes_gui.
- Whenever the local window receives XEvent, pass information on it to AppVM (particularly, mouse and keyboard data).
- Whenever AppVM signals damage event, tell local Xorg server to repaint a given window fragment.
- Receive information about window size/position change, apply them to the local window.

https://www.qubes-os.org/doc/gui/

Updating Windows

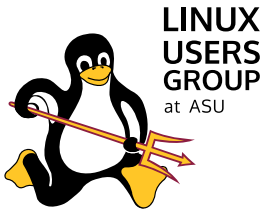


1. qubes_gui
   1. Asks the qubes_drv driver for a list of physical memory frames that hold the composition buffer of a window.
   2. Passes on this information to the qubes_guid driver in dom0 via a MFNDUMP message.
2. qubes_guid
   1. Xorg server is started with a LD_PRELOD-ed library named shmoverride.so which hooks all functions calls related to shared memory.
   2. qubes_guid creates a shared memory segment and then instructs Xorg to attach it via the MIT-SHM extension.
   3. When Xorg tries to attach the segment via glibc shmat, shmoverride.so intercepts the calls and instead maps AppVM memory via xc_foreign_pagers
   4. We can then use MIT-SHM functions to draw onto a dom0 window.

https://www.qubes-os.org/doc/gui/

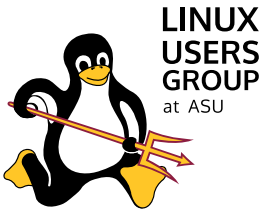
Messages are a set list of messages that the GUI drivers use to communicate.

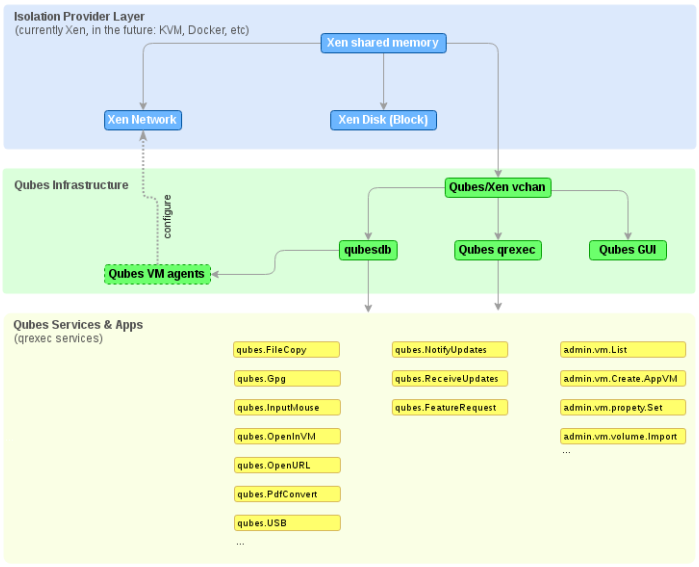
Full list is at the bottom of this: https://www.qubes-os.org/doc/gui/

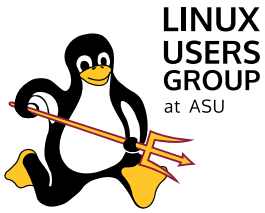


Section 4.4 – General Overview

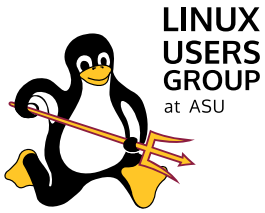
Overview


LINUX
USERS
GROUP
at ASU


Isolation Provider Layer
(currently Xen, in the future: KVM, Docker, etc)
Xen shared memory
Xen Network
Xen Disk (Block)
Qubes Infrastructure
configure
Qubes/Xen vchan
qubesdb
Qubes qrexec
Qubes GUI
Qubes VM agents
Qubes Services & Apps
(qrexec services)
qubes.FileCopy
qubes.Gpg
qubes.InputMouse
qubes.OpenInVM
qubes.OpenURL
qubes.PdfConvert
qubes.USB
...
qubes.NotifyUpdates
qubes.ReceiveUpdates
qubes.FeatureRequest
admin.vm.List
admin.vm.Create.AppVM
admin.vm.propety.Set
admin.vm.volume.Import
...

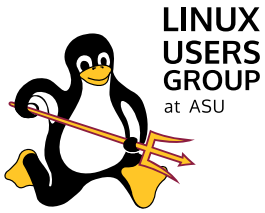


Section 5 – Security and Attack Surfaces



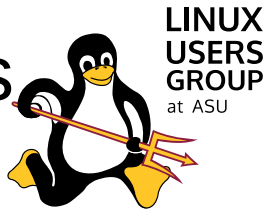
Section 5.1 – Project Security

Security Team



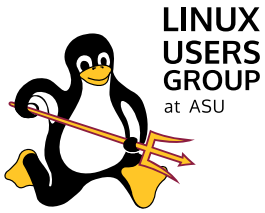
- Handles disclosed security issues.
- Evaluates the importance of security advisories of dependencies.
- Applying security patches.
- Publishing security bulletins and canaries.
- Managing the project’s PGP keys.

Security Team Members



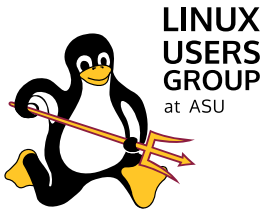
- Marek Marczykowski-Górecki – Project Lead
- Simon Gaiser – Project Developer
- Joanna Rutkowska (canaries only) – Project founder

Signing and PGP Keys



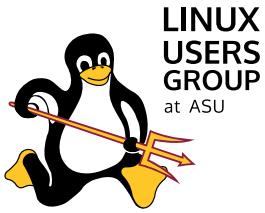
- Qubes has signing keys for each major release.
- Every Qubes team member’s PGP keys are published on the Qubes key site.
- Every member of the security team must sign a security release.

Canaries

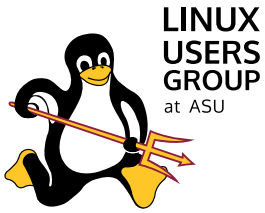


- Canaries are periodical security announcements by the Qubes security team.
- Requires every member sign them.
- Includes specific information such as the date, its number, when the next one plans to be, etc.
- Includes proof of time by news articles and the hash of a Bitcoin block mined in the last 24 hours before the release.

Security Bulletins

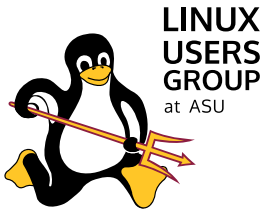


- Special announcement issued by the security team.
- Contains summary and analysis of a specific set of vulnerabilities and their patch.



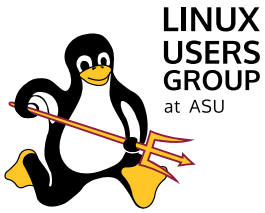
Section 5.2 – Hardware Mitigation

Disk Encryption and MFA

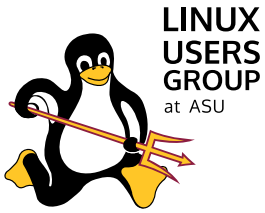


- LUKS 2 full disk encryption.
- U2F Authentication – Only an alternative
- Multi-factor Authentication – Only for restricting access to external services, not the login.

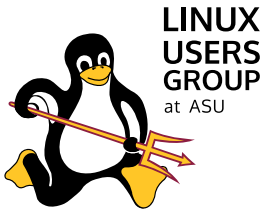
Anti Evil Maid



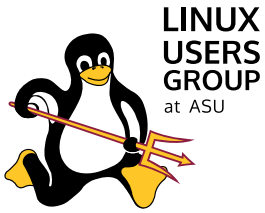
- External TPM trusted boot.
- If the whole chain is valid and properly executed, a message will be loaded validating it.
- Requires USB devices to be able to attach directly to dom0.



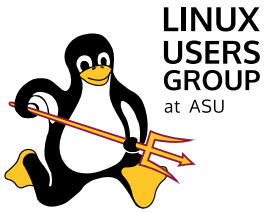
Section 5.3 – Attack Surfaces and Risk



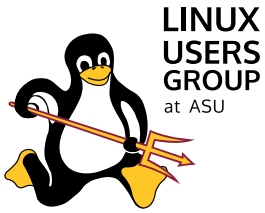
Physical Security of the Device



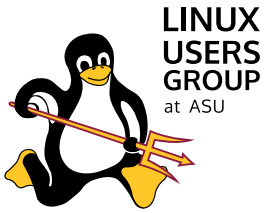
Hypervisor



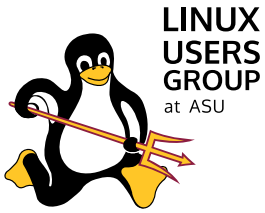
dom0 Drivers



Custom-built Malware

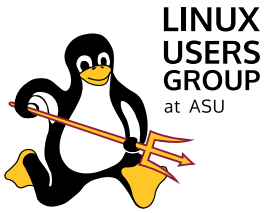


Real World Risk?



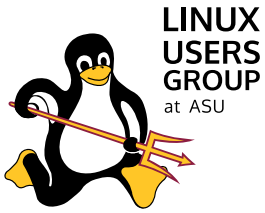
Conclusion

Check out the Project



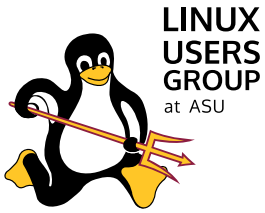
- QubesOS official website: https://www.qubes-os.org/
- Download QubesOS: https://www.qubes-os.org/downloads/
- QubesOS Forums: https://forum.qubes-os.org/
- QubesOS GitHub: https://github.com/QubesOS

Support the Project

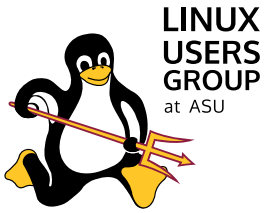


- QubesOS Donation: https://www.qubes-os.org/donate/
- Contribute: https://www.qubes-os.org/doc/contributing/

Resources



- Qubes Website: https://www.qubes-os.org/
- Qubes Documentation: https://www.qubes-os.org/doc/
- Qubes GitHub: https://github.com/QubesOS
- Xen Project: https://xenproject.org/
- Xen Project Wiki: https://wiki.xenproject.org/wiki/Main_Page
- Invisible Things Lab: https://invisiblethingslab.com
- Tarsnap website and documentation: https://www.tarsnap.com/



Questions?