Security

- Flatpak
- Qubes OS
- Intel ME Cleaner

Flatpak

A list of permissions to whitelist applications using flatpak

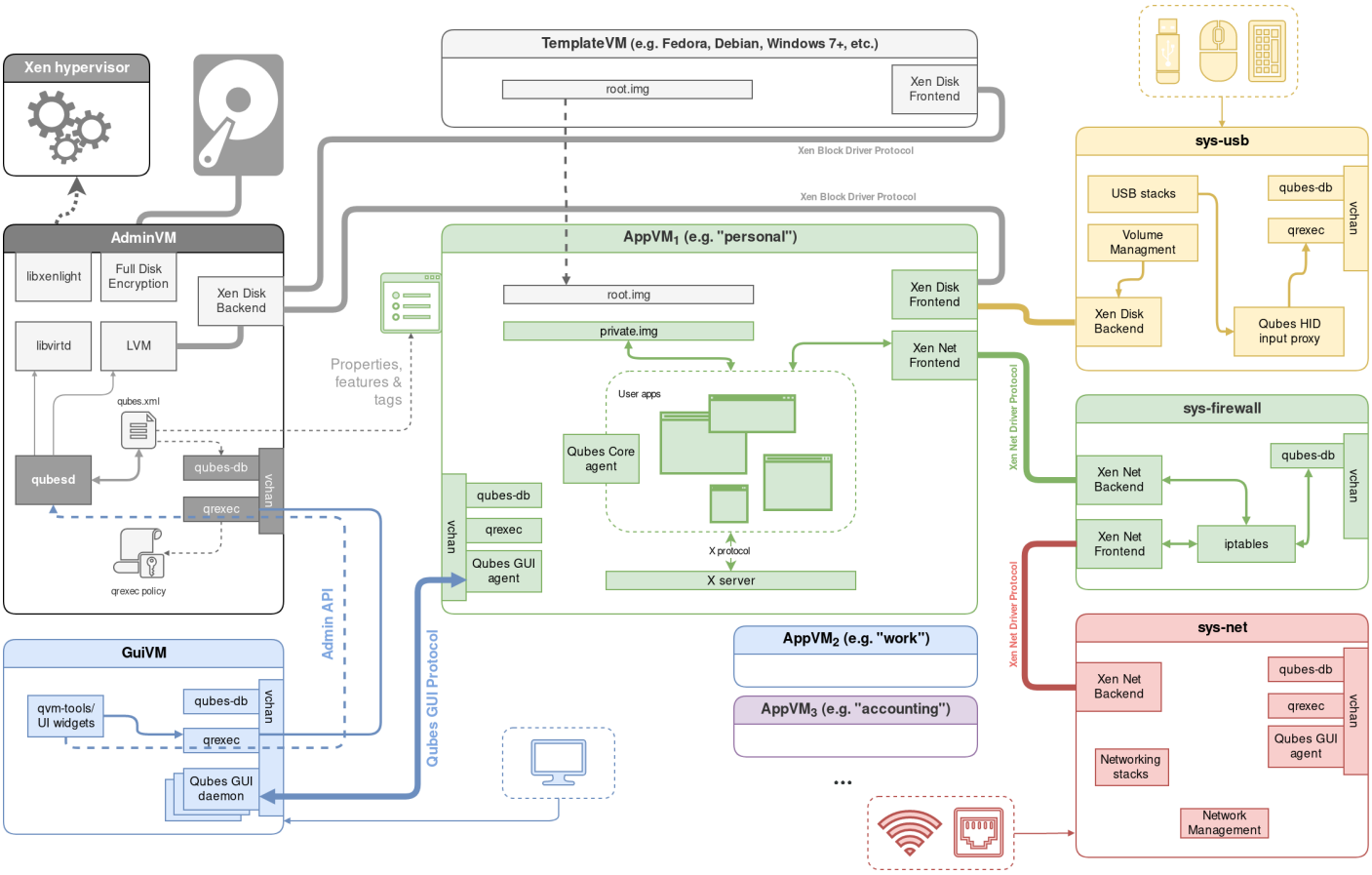
- Android Studio

```
~/Android:create

~/.android:create

~/.gradle:create

~/.java:create
```

Qubes OS

QubesOS is essentially a hypervisor that you can use as a desktop OS. Qubes allows you to run all of your programs inside of VMs, and streamlines the experience. Pretty neat, huh?

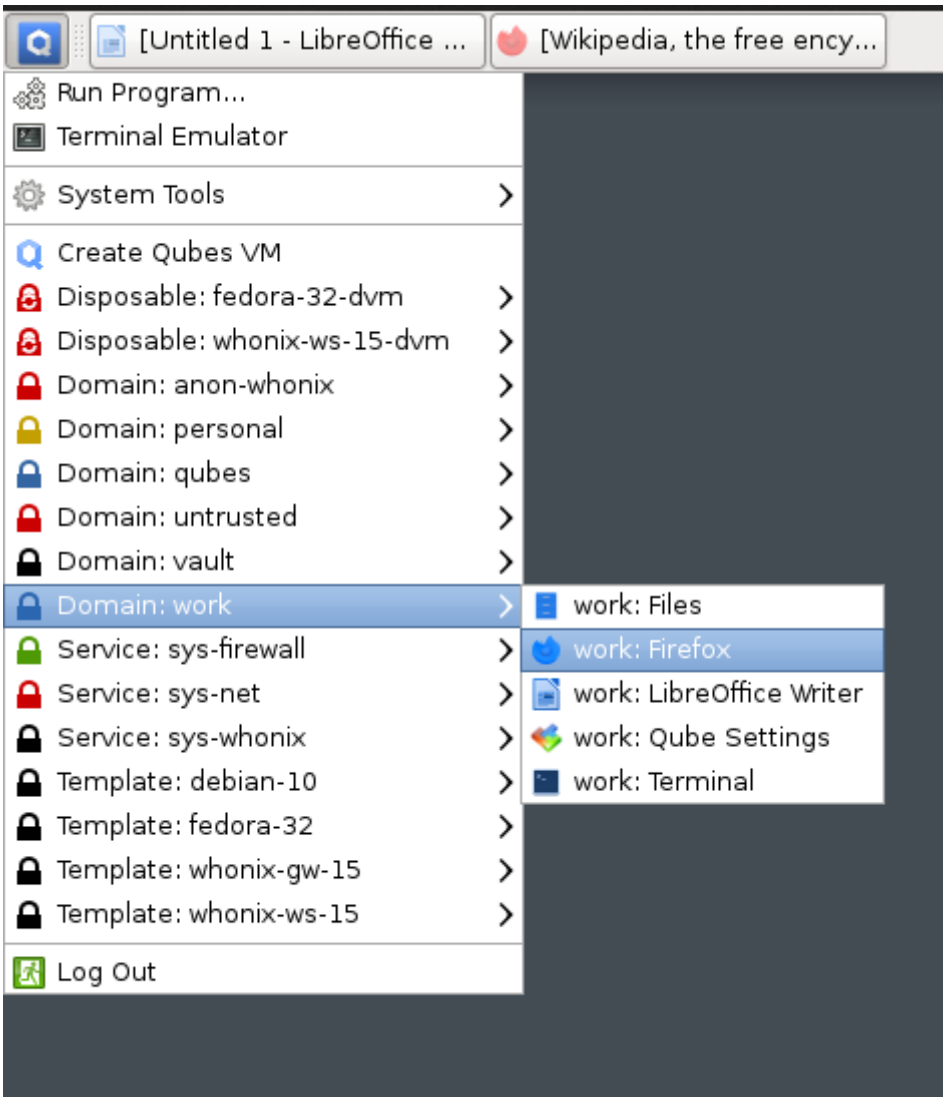
The beauty of Qubes is, besides being the a perfect OS for self-prescribed OCD people like myself, it is very functional. Qubes lets you perform tasks that may necessitate installing some of that undesirable closed-source software, such as TeamViewer or Citrix, within disposable virtual machines (VMs).



Every box with rounded corners represents a virtual machine. Glorious!

But how?

Qubes builds on top of the Xen hypervisor and presents it's own UI in the form of a desktop fedora installation. There, just like a normal Linux desktop, you can launch your applications.



Hmm, this looks strange

The fact that Qubes runs all of your applications in a VM means that you actually need to first select the *virtual machine*, then select the *application* you would like to open. By isolating each application within its own VM, QubesOS prevents malware or vulnerabilities in one app from affecting others or the underlying system. It's like having separate, secure rooms for each activity, reducing the risk of a compromise spreading throughout your digital life.

Ideally, having every application be isolated into its own VM would be the best for security. However, this can be quite resource intensive especially for slower hardware. What Qubes does, then, is isolate applications into different *security domains.*

Qubes also does some fancy, arguably over-the-top things such as isolating usb and network devices from the host (seen as AdminVM aka dom0 in the diagram), in an effort to contain would-be exploitation attempts safely inside VMs.

Solutions to Qubes Quirks

- If your input devices are not directly connected to your computer's USB controller, you will need to remove the `usbcore` device filter from your grub options:

```
# /etc/default/grub
GRUB_CMDLINE_LINUX= ... usbcore.authorized_default=0 ... # Remove the usbcore definition
grub2-mkconfig -o /boot/efi/EFI/qubes/grub.cfg # Reconfigure the grub bootloader
```

Intel ME Cleaner

Using me_cleaner

> Needless to say, there is a certain amount of risk of bricking your system when it comes to this procedure. Proceed with caution!

Make sure that any kind of battery is unplugged from the computer

- This includes your laptop battery as well as your RTC (coin cell) battery!

Clip an SPI reader that is compatible with flashrom onto your motherboard's bios chip

- Supported programmers
- Make sure to consult the datasheet for the specific EEPROM that your motherboard uses when wiring the pins to the programmer. For example, on a

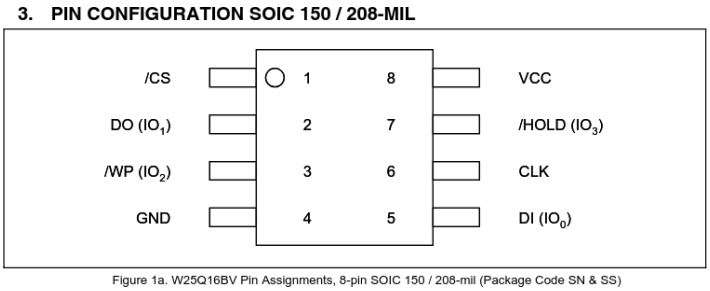


Figure 1a. W25Q16BV Pin Assignments, 8-pin SOIC 150 / 208-mil (Package Code SN & SS)

On a separate computer, interface with the SPI programmer and try reading off of the EEPROM

```
flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=1000 -r original_dump.bin
```

- May need to mess around with SPI speed
  - If you are using a Raspberry Pi, make sure to enable SPI using `raspi-config`!
- Dump the firmware multiple times and store them safely.

Use me_cleaner to modify the original firmware dump

Consult the project's wiki. Essentially, you can do one of two things:

```
# Set HAP bit as well as attempt to remove portions of the ME firmware
python me_cleaner.py -S -O modified_image.bin original_dump.bin

# Only sets HAP bit. Used if the above command fails
python me_cleaner.py -s -O modified_image.bin original_dump.bin
```

Write to the EEPROM

```
flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=1000 -w modified_image.bin
```

You may also want to, after writing to the EEPROM, *read the firmware again* and use me_cleaner to verify that the firmware has been cleaned properly.

```
python me_cleaner.py -c firmware_to_check.bin

# You can also use intelmetool
sudo intelmetool -m

# Output:
# Can't find ME PCI device
```

You can also check in your bios. On my T480 and L390, the `ME Firmware Version` is blanked out.

Cheers!

Other Notes

Updating the BIOS

For whatever reason, my T480 *really* does not want to update its own bios after using me_cleaner. In contrast, my L390 does not have this issue. Anyways, if you choose to update the bios, make sure that me_cleaner is still active afterwards using the aforementioned methods. My L390 for example keeps its HAP bit set even after a bios upgrade.