Project

Create a Policy Manager GUI for QubesOS
Resource touchpoints: GitHub Issue, GitHub Policy.md, Docs pages one, two, three, four

Problem

In order for a reasonably secure hypervisor system to also be usable by folks in the real world, end users need to be able to manage a host of inter-VM permissions. These permissions exist under-the-hood in Qubes OS as *Policies*.

Unfortunately however, users are today only able to reap the benefits of various Policy *Services*, if they both know about and can work with the *qrexec policy system*. To do this they must:

1. To know that such a thing called Policies exists;
2. Be comfortable with using the Command Line;
3. Be capable of abstracting basic "I need this…" thoughts into computer jargon; and
4. Be curious enough about Policies to learn about the complete qrexec system in the Qubes docs.

All of those requirements exclude a number of both technical and non-technical Qubes OS users, from taking advantage of this essential capability in making the most of the Qubes experience.

Users Targeted

Primary: non-technical, high risk folks with a lot to do and who "just need things to work."

- Journalists
- Human rights defenders
- Academics
- Workers in enterprise environments (Gov or Corporate)
- Folks currently dependent on Macs, Ubuntu, or Windows for "daily-driver" machines

Secondary: technical high-risk folks and developers. Early adopters with the patience to live with quirks, troubleshoot in community forums, and are comfortable with a CLI.

- The majority of today's users
- Folks with a need for a hypervisor that extends beyond just security stuff
- Security researchers whose "needs" include breaking things
- Libre and Linux fans

Why?

Consumer products have set the bar for patience and willingness of folks to use new tools. If Qubes OS is not intuitive enough for folks other than early-adopters to use on a consistent basis over time, they'll abandon it to use less-secure options; trading-off increased vulnerability to threats, for decreased friction when using digital tools. That's not ok. For the Qubes OS project to sustain, its user-base needs to grow.

To minimize user attrition and to attract new users, a balance needs to be struck between CLI-only and developer-friendly capabilities, and GUI access to rich information & functionality needed by everybody—whether they know it or not. Robust capability, with minimal fanfare.

As such, many Policy Services currently unavailable in the UI and that may deliver value to at-risk users, need to be accessible in simple, intuitive GUI controls. Controls either unified in a "Permissions Manager" type of preferences panel, or as settings contextually located in relevant widgets.

Design Principles

- **Plain Language.** Temper the mental gymnastics required by folks, today, to make the most of Policies for their own needs.
- **Promote Discovery.** Of advanced security or permissions functionality that does not exist in the GUI, today, and most users will not expect.
- **Intuitive.** Present controls and concepts that call upon common mental models for everyday users, so that referencing (technical) documentation is optional.
- **Value.** Surface only policies in the GUI that have clear value to a majority of users. Ensure that they can be presented as simple form controls with options for choices that make clear the value of all configuration options. As of this writing, that may only be ~20% of all Policy Services. The other ~80% of Services (such as *StartApp* and *ResizeDisk*), exist to support "Backstage Actions" across Qubes, and would only confuse users in a permissions GUI—so should remain hidden.
- **Empowering.** Instil confidence and agency in users to make informed hardening or customization decisions with ease, and to develop the curiosity to learn more about opportunities for security hardening through isolation permissions, in the process.
- **"Speed" per OG Google Design team.** Enable users to act upon and/or to meaningfully consume as much information, as quickly as possible. Form follows function.
- **Light.** On graphics, any perceived bulk, and supportive aesthetics. Utility is key. Icons to use sparingly to communicate simple semiotics to minimize translation and character-count bloat. Detailed Service controls should remain hidden, unless a user chooses to act upon a relevant service.

Problem — *the "have a tea with it" version*

In order to serve the advanced security needs of at-risk users, Qubes OS was created as a multi-environment system to compartmentalize different protection domains with virtual machines (VMs). However, if each VM had the access porosity—internet, external and internal devices, and other VMs—of a sieve, the isolation benefits of compartmentalization would be rendered moot. Conversely, if each qube were entirely enclosed in their own protective bubbles, none would be of much use to anybody. Nothing could ever come in or go out.

To enable a reasonable balance between security and usability for a variety of folks in a world with a variety of threats, a system of access permissions among qubes and with the outside world was formed, called *Policies*. Without *Policies,* Qubes OS would be little more than a giant collection of walled gardens, and effectively useless.

Policies

Policies today are written as text files that follow a set syntax and format to make arguments within a function, and are saved to a directory on each Qubes OS device.

Policies are composed of a *Service* (or "Operation"), and establish permissions for that Service between a *Source VM* and a *Target VM*. Many services additionally have *Argument* and *Parameter* values to facilitate richer detail around what parts of the Service are being impacted. The basic permissions parameters today, are *Allow*, *Deny*, and *Ask*. Finally, all individual policy statements are made between either an individual *Source* and *Target* VM, or All VMs via @anyvm.

In standard single-environment systems, similar systems of programmatic *argument* based rulesets also exist—but are surfaced in the UI as simple "Do you want x? (y) (n)" value propositions to users; typically under an OS' "Preferences" or "Settings."

Each instance of Qubes is installed with a set library of basic "Qrexec Services," that folks can use to establish permissions between specific qubes around. Upon installation, a wizard asks users a few simple questions so that it can translate those answers into a Policy file for one of those Services (how updates are downloaded). Outside of that experience, however, Policy visibility, creation, and management are restricted to use of CLI tooling. Furthermore, users must be comfortable with abstracting their own use and security needs most naturally modeled as "I need this to do that," into *arguments* and Qrexec syntax.

Users today need to have a reasonable fluency with Bash, and the mental processing skills to abstract human thinking into computer functions. That excludes a lot of users, while further excluding users with those skills who simply don't have the time to learn about this system outside the context of daily use (so, reading docs).

TL;DR, it's time to do that for Qubes' qrexec capabilities.