Secure Desktops with Qubes: Introduction

Learn about next-generation desktop security with Qubes.

KYLE RANKIN

Kyle Rankin is a Sr. Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *The Official Ubuntu Server Book*, *Knoppix Hacks* and *Ubuntu Hacks*. He is currently the president of the North Bay Linux Users' Group.

PREVIOUS
Dave Taylor's Work the Shell

NEXT
Shawn Power's The Open-Source Classroom

THIS IS THE FIRST in a multipart series on Qubes OS, a security-focused operating system that is fundamentally different from any other Linux desktop I've ever used and one I personally switched to during the past couple months. In this first article, I provide an overview of what Qubes is, some of the approaches it takes that are completely different from what you might be used to on a Linux desktop and some of its particularly interesting security features. In future articles, I'll give more how-to guides on installing and

configuring it and how to use some of its more-advanced features.

When it comes to Linux security, server security tends to get the most attention. When you are hardening servers, you generally try to limit what any individual server does and use firewalls to restrict access between servers to only what is necessary. In a modern environment where a server is running only SSH plus maybe one or two other networked services, there are only a few ways for an attacker to get in. If a particular server does get hacked, ideally you can detect it, isolate that server and respond to the emergency while the rest of your environment stays up.

Desktop Linux security is a completely different challenge because of just how many *different* things you do with your desktop. Each action you take with your desktop computer opens up a new way to be compromised. Web browsing, especially if you still have certain risky plugins like Flash installed, is one major way a desktop can be compromised. E-mail is another popular attack vector since you need to open only one malicious e-mail attachment or click on one malicious phishing link for an attack to succeed. Linux desktops also often are used as development platforms, which means users might be downloading, building and executing someone else's code or running services directly on their desktop to test out their own code. Although some Linux users are smug when they think about all of the malware on other platforms, the fact is that the days when Windows was the only desktop OS in town are over, and these days, much of the malware is written in a cross-platform way so that it can run on many different operating systems.

The biggest issue with desktop Linux security is what's at risk if you do get hacked: all of your personal data. This could be anything from user names and passwords to important accounts like your bank or credit-card accounts, your social-media accounts, your domain registrar or Web sites you shopped at in the past that have your credit-card data cached. An attack could expose all of your personal photos or access to private e-mail messages. Attackers could leave behind a Remote Access Trojan that lets them get back into your machine whenever they want, and in the meantime, they could snoop on you with your Webcam and microphone. They even could compromise your SSH, VPN and GPG keys, which opens up access to other computers.

The core idea behind how Qubes provides security is an approach called

security by compartmentalization. This approach focuses on limiting the damage an attacker can do by separating your activities and their related files to separate virtual machines (VMs). You then assign each VM a certain level of trust based on the level of risk that VM presents. For instance, you may create an "untrusted" VM that you use for your generic, unauthenticated Web browsing. You then might have a separate, more-trusted VM that you use only to access your bank. You may decide to create a third highly trusted VM that has no network access at all that you use to manage off-line documents. If you also work from your personal computer, you may create separate VMs for personal versus work activities, with the work VM being more trusted. If you browse to a malicious Web site with your untrusted Web browser, the attacker won't have access to your banking credentials or personal files since you store those on different VMs. Qubes even provides disposable VMs: one-time-use VMs that are deleted completely from disk after the application closes.

How Qubes Works

Although you certainly could use any of the virtual machine technologies out there to set up multiple VMs on your regular Linux desktop, that kind of arrangement can end up being pretty clunky, especially if you don't want multiple desktop environments running inside their own windows. There also are all kinds of mistakes you could make with that kind of set up that would eliminate any security benefits you might get. For instance, how should you share files or copy and paste between VMs securely, and how do you keep all of those VMs up to date with security patches?

Where a traditional Linux distribution made it easy for you to get all of the software you wanted to use without having to download and compile it all, Qubes provides a number of extra tools that makes it easy to manage a desktop full of different virtual machines all with different levels of trust. Qubes also approaches all aspects of the desktop with security at the forefront and uses secure defaults throughout the OS. In doing so, Qubes makes it more difficult (but not impossible) for you to shoot yourself in the foot.

Qubes uses Xen to provide all of its virtualization (if you want to know why Qubes chose that over other technologies, see the FAQ on the Qubes site). Instead of each VM having its own complete desktop

environment, Qubes uses the more-privileged dom0 Xen VM as a host for the desktop environment (currently Qubes gives you the choice of KDE or XFCE, although the community has contributed others), and the other VMs display individual application windows within dom0's desktop environment.

So, launching Firefox in Qubes behaves much like you would expect in any other desktop distribution. The main difference, however, is that Qubes lets you color-code each of your VMs based on level of trust ranging from red (untrusted) to black (ultimately trusted) with a number of different rainbow colors in between.

When you launch an application from an application VM (appVM, in Qubes parlance), the VM starts up if it wasn't started before, then the application appears with a window border that is colorized based on the color you assigned its appVM. So, if you have two instances of Firefox on your desktop at the same time, you can tell your untrusted Web browser from your banking Web browser, because the untrusted one might be colored red while your banking browser might be colored green. Figure 1 provides a screenshot from Qubes' documentation that

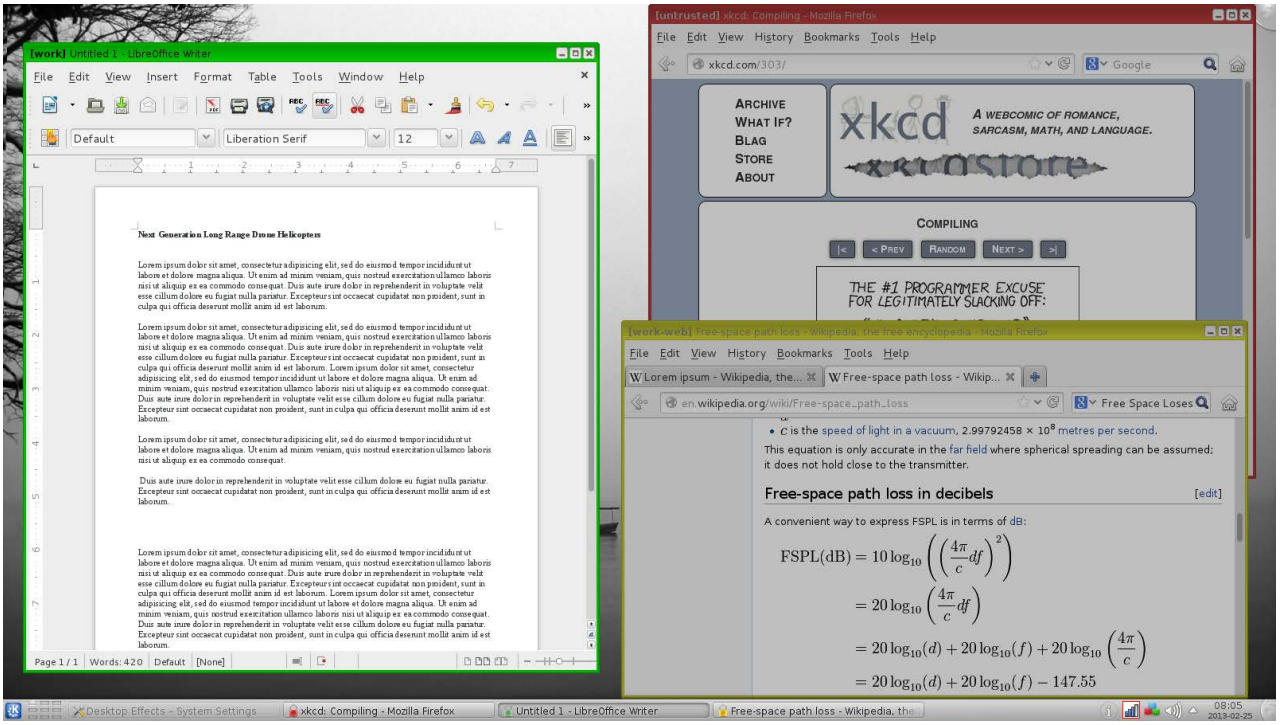

Figure 1. Multiple Windows with Different Colors

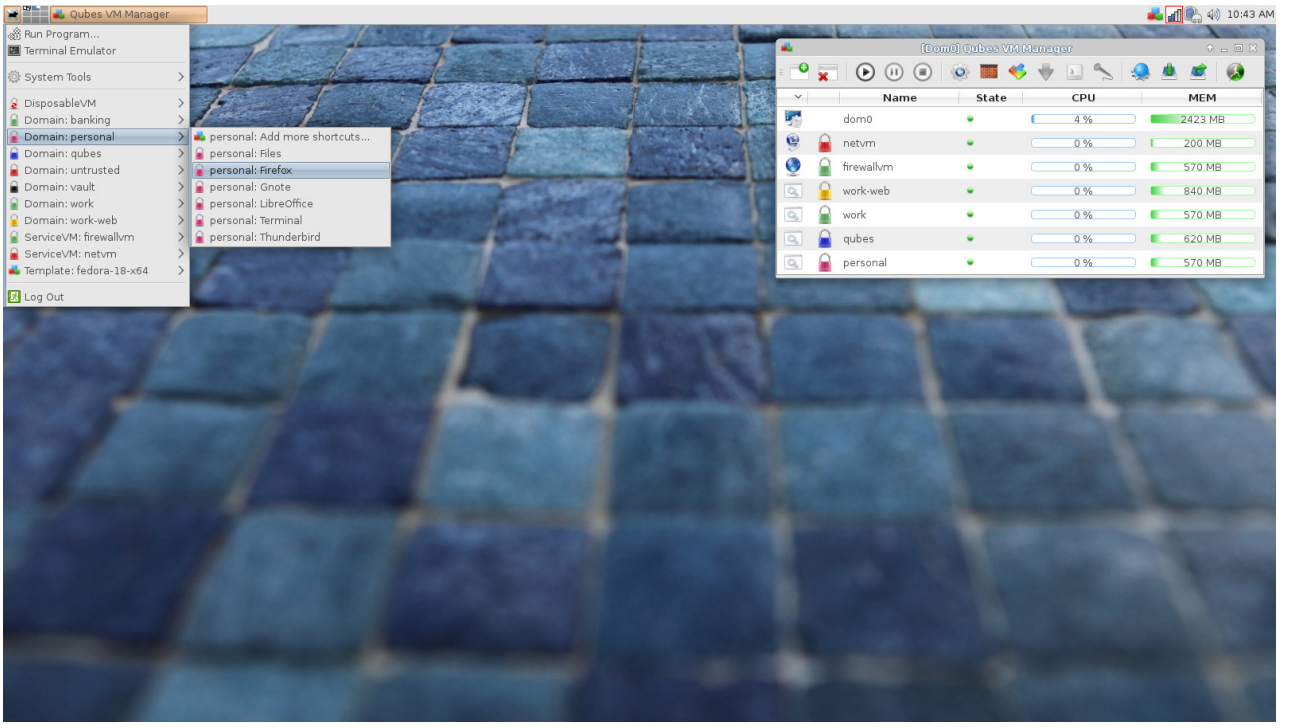

Figure 2. Qubes Application Menu

demonstrates the point.

Since the dom0 VM has privileged access to data about the other VMs in Xen, Qubes goes to extra lengths to protect it by having only the desktop environment run from it and by removing all network access from dom0. You are encouraged to do as little as possible in dom0, and instead, you should use appVMs for any applications you want to run. Qubes even intentionally makes it more difficult to copy files to or from dom0 compared to copying them between appVMs. In the dom0 desktop environment's application menu, each VM has its own submenu where you can launch each of its applications (Figure 2). Qubes provides tools so all of those submenus don't become too unwieldy, and you can select which applications appear under which appVM's menu.

Sharing Information between AppVMs

When you have multiple windows open, however, that raises the question of how do you copy and paste? An insecure approach might be to share the clipboard between all windows, but then the risk

would be that if you logged in to a Web site in a trusted Web browser by copying and pasting from your password manager, that password would be readable by any other appVMs that happened to be running. Instead, Qubes provides a two-tier approach to clipboards. Each appVM has its own clipboard, and you can copy and paste within that appVM as normal. If you want to copy from one appVM and paste to another, once you have put the data in one appVM's clipboard, you press Ctrl-Shift-c to put it in the global clipboard, then highlight the window you want to paste into and press Ctrl-Shift-v to paste that data into that VM's clipboard and wipe it from the global clipboard. Then you can paste inside that application as normal. It's definitely an extra cumbersome step, but you would be surprised at how quickly you adapt to: Ctrl-c, Ctrl-Shift-c, change window, Ctrl-Shift-v, Ctrl-v. It definitely helps you prevent accidentally pasting information into the wrong window.

Qubes also provides a command-line tool and right-click menu options in the GUI file manager so you can copy or move a file between appVMs. When you attempt this, you get a prompt in a black-bordered window the appVM doesn't control so you can accept this file transfer. Even then, the file doesn't appear wherever you want on the destination VM (otherwise an attacker could overwrite important files with backdoored versions). Instead, the files show up in a QubesIncoming directory inside your home directory.

TemplateVMs, Persistence and Backdoor Protection

Another area where Qubes provides an extra level of protection for a desktop user is in how it handles persistence. If attackers compromise a normal desktop, they can install backdoored versions of utilities, like ls or bash, or add extra programs that are triggered to start at boot. With Qubes, appVMs are based off templateVMs that have base installs of Fedora, Debian or Whonix by default (the community has provided templates for other popular distributions). When you create a new appVM, you choose which template it is based from, and when you start it, that appVM gets a read-only version of that template's root filesystem. Although the user inside the appVM still can install software or change the root filesystem, when that appVM shuts down,

This means your browser history and settings will stick around, but if an attacker did compromise your browser and tried to install a backdoor into bash or Firefox, the next time you rebooted that appVM, the backdoor would be gone.

all of those changes are erased. Only the /rw, /usr/local and /home directories persist. This means your browser history and settings will stick around, but if an attacker did compromise your browser and tried to install a backdoor into bash or Firefox, the next time you rebooted that appVM, the backdoor would be gone.

Also by default, appVMs do not launch any common init services like cron automatically. That means attackers also couldn't just add a user cron entry that launched the backdoor. Although it's true that attackers could store a malicious program in your appVM's home directory, the next time you reboot the appVM, the program no longer would be running, and they would have no way to launch it again automatically.

So, how do you install software? Because each appVM uses a root filesystem based on its templateVM, when you want to install new software, you launch the software manager from the templateVM and install the application with yum, apt-get, the GUI equivalent or whatever other method you normally would use to install the software. Qubes then detects any new application menu items you've added and makes them available to the appVMs based on that template.

The only gotcha is that those newly installed applications are unavailable to appVMs until those appVMs restart. Because compromising the templateVM compromises every appVM based on it, Qubes generally encourages you to leave templateVMs off, to not run general applications from them and to turn them on only when you add trusted software. Although this does add an extra bit of work when you want to install software, it also provides a nice benefit

in that when you need to apply a security update, you just need to update the templateVM, and when you restart each appVM, it will get the update.

Network Security with netVMs

Another way Qubes provides security is by compartmentalizing the network. Upon installation, Qubes will create a few special system VMs called network VMs (netVMs), named sys-net, sys-firewall and sys-whonix. The sys-net netVM is assigned any networking hardware on your host, so it's unavailable to any other VM. Because this netVM is the only one with an IP on the external network, it's considered untrusted and colored red. You use Network Manager to configure this netVM with any credentials it needs to connect to wireless networks, and its Network Manager applet shows up on your desktop as normal. The sys-firewall VM (technically classified as a proxyVM) is colored green and connects to sys-net for its network access. By default, any appVMs you create then use sys-firewall for their network access.

Why all this complexity? First, sys-firewall acts as a true firewall for all of your appVMs. Although by default all appVMs can talk to the Internet unrestricted, Qubes provides a GUI tool that makes it easy to lock down individual appVMs so that they can access only certain hosts on the network. For instance, you could restrict your banking appVM so that it can talk only to port 443 on your banking Web site or restrict an e-mail appVM to talk only to your remote mail server. You even could restrict other VMs so that they could talk only to hosts on your internal network. Anyone who wants to attack one of your appVMs has to go through sys-net and sys-firewall. This also means if attackers do compromise an appVM, they don't have direct access to network hardware, so they can't, for instance, automatically connect to a different wireless access point.

The sys-whonix VM acts like sys-firewall except that it automatically sets up a secure Tor router. Any appVMs that use sys-whonix instead of sys-firewall or sys-net for their network have all of their traffic routed over Tor automatically. Qubes also provides an anon-whonix appVM by default that uses the security and anonymity-focused

distribution Whonix, and it includes the Tor browser and routes all traffic through sys-whonix by default.

I'm sure you already can see a number of areas where Qubes provides greater security than you would find in a regular Linux desktop. Hopefully, you have a sense of what a different approach Qubes takes from what you might be used to. With Qubes, you find yourself thinking much more about how you should isolate files and information and what attackers could get if they successfully compromised one of your appVMs. Even the extra copy-and-paste and file-copy steps force you to confront whether you are transferring information between an untrusted VM to a trusted one and think through the implications. I've found the extra security measures actually let me relax a little bit more than I would otherwise, because for instance, I know an e-mail attachment I open in a disposable VM can't do me much harm, or a malicious Web site in my untrusted Web browser can't access anything of value.

I've touched on only some of the higher-level security features in Qubes with this article. In my next article, I will describe how to download and install Qubes, explain how to use Qubes as a desktop OS, including some of the basic features of the Qubes VM Manager and other Qubes-specific tools, and give some examples for how you might organize your day-to-day desktop use across appVMs. I'll follow up with an article describing more-advanced Qubes features, including split-GPG (a method that allows appVMs to use your GPG private key without having direct access to it), how to manage links more securely with default application handlers, how to open e-mail attachments automatically in disposable VMs, and how to create a usbVM that isolates all of your USB devices for you (and why you would want to do that).■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS