Multi-factor Login

14-18 minutes ⋮ 12/7/2024

Multi-factor authentication within particular qubes

Most use cases for the hardware tokens can be achieved exactly as described by the manufacturer or other instructions found online. One usually just needs to attach the token (e.g. YubiKey) to the corresponding app qube to get the same result (see the documentation on how to use USB devices in Qubes OS accordingly). The recommended way for using CTAP in Qubes is described here.

Multi-factor login for Qubes OS

By default Qubes has two protection mechanisms against attackers. The first is full disk encryption and the second the user login screen / lockscreen. This article section concerns only adding multi-factor authentication to the second one.

Time-based One-time Password (TOTP)

As the name implies, this generates authentication code that is time-dependent. You can save the TOTP secret in a mobile app like FreeOTP and then use it as an additional factor to login to your Qubes system.

> **Warning**: remember to keep backup access codes.

1. Download `google-authenticator` in dom0:

   ```
   sudo qubes-dom0-update google-authenticator
   ```

2. Run google authenticator:

3. Walk through the setup instructions 2 which will also generate your QR code for your auth app of choice:

   ```
   Do you want me to update your “/home/
   user/.google_authenticator” file (y/n) y

   Do you want to disallow multiple uses of the same
   authentication token? This restricts you to one login about
   every 30s, but it increases your chances to notice or even
   prevent man-in-the-middle attacks (y/n)
   ```

```
By default, tokens are good for 30 seconds, and to
compensate for possible time-skew between the client and the
server, we allow an extra token before and after the current
time. If you experience problems with poor time
synchronization, you can increase the window from its
default size of 1:30min to about 4min. Do you want to do so
(y/n)

If the computer that you are logging into isn't hardened
against brute-force login attempts, you can enable rate-
limiting for the authentication module. By default, this
limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting (y/n)
```

Warning: in the next session if incorrectly performed, there is the risk of locking yourself out. Before procedding ensure that you have an up-to-date backup.

For advanced users, to make sure you can quickly recover, you can also open another loging session in a tty. To do this, you do `ctrl+alt+F2` and login normally. Should anything go wrong, as long as you don't shut the computer down, you can still access this tty by entering the same key combination and reverting the changes. After you've opened this “backup” login, you can get to your graphical desktop with `ctrl+alt+F1`.

Now we are going to add the authenticator as a login requirement:

1. `sudo authselect create-profile mfa --base-on sssd`
2. Edit the custom system authentication template with `sudo nanois encouraged /etc/authselect/custom/mfa/system-auth`.

   Add the following line right after `auth required pam_faildelay.so delay=2000000`:

   ```
   auth required pam_google_authenticator.so
   ```

   After the change, the top of the file should look like this:

   ```
   {imply "with-smartcard" if "with-smartcard-required"}
   auth required pam_env.so
   auth required pam_faildelay.so delay=2000000
   auth required pam_google_authenticator.so
   ```

3. Lastly, activate this authentication method with:

   ```
   sudo authselect select custom/mfa
   ```

Now you can test by locking the screen with `ctrl+alt+l`. If it was successful and you are pleased with the results, restart your computer.

Note: When logging in. the first thing you put is the TOTP secret and then the password. This is true in the screen locker and as well as the session manager (the login window that shows right after you put the disk encryption passphrase).

After this is done, its recommended to do a backup. This is because as long as you incude dom0 in the backup, your authentication code will be backed up as well.

Troubleshooting

The following assumes you haven’t restarted your computer since setting up TOTP secret.

1. Switch to TTY2 with `ctrl+alt+F2`.
2. Revert to the original policy with:

   ```
   sudo authselect select sssd
   ```

3. Switch back to the graphical desktop with `ctrl+alt+F1`. You should be able to login normally (without multi-factor authentication).
4. Change the mfa custom policy and apply it again.

Lost TOTP / authentication device?

In case you’ve lost your TOTP authentication device, you have two options.

The first option is backup codes. When generating the TOTP secret you must have saved some recovery codes. Those can be used in place of the TOTP code, but they’re discarded after use. So make sure you redo the multi-factor authentications intructions.

The second option is recovery from a backup. It will work as long as you included dom0 in said backup. After restoring the dom0 backup, open a terminal in dom0 and the file should be located in `/home/<USER>/home-restore-<DATE>/dom0-home/<USER>/.google_authenticator`.

Login with a YubiKey / NitroKey3

The YubiKey / NitroKey3 is a hardware authentication device manufactured by Yubico / NitroKey to protect access to computers, networks, and online services that supports one-time passwords (OTP), public-key cryptography, and authentication, and the Universal 2nd Factor (U2F) and FIDO2 protocols[1] developed by the FIDO Alliance.

You can use a YubiKey / NitroKey3 to enhance the user authentication in Qubes. The following instructions explain how to setup the YubiKey / NitroKey3 as an *additional* way to login.

After setting it up, you can login by providing both - a password typed in via keyboard *and* the YubiKey / NitroKey3 plugged in. Someone eavesdropping your login attempt would not be able to login by only observing and remembering your password. Stealing your YubiKey / NitroKey3 would not suffice to login either. Only if an attacker has both, the password and the Yubikey / NitroKey3, it would be possible to login (it is thus called Multi-factor authentication).

The following instructions keep your current login password untouched and recommends to define a new, additional password that is used in combination with the YubiKey / NitroKey3 only. This ensures that you a) do not accidentally lock yourself out during setup and b) you do not need to fear shoulder surfing so much (i.e. by not using your standard login password in public).

Setup login with YubiKey / NitroKey3

To use the YubiKey / NitroKey3 for multi-factor authentication you need to

- install software for the YubiKey / NitroKey3,
- configure the YubiKey for the Challenge-Response mode or the NitroKey3 for HOTP mode,
- store the password for YubiKey / NitroKey3 Login and the Challenge-Response / HOTP secret in dom0,
- enable YubiKey / NitroKey3 authentication for every service you want to use it for.

All these requirements are described below, step by step, for the YubiKey and NitroKey3. Note that setting up both a YubiKey and a NitroKey3 is not supported.

1. Install YubiKey / NitroKey3 software in the template on which your USB VM is based. Without this software the challenge-response / HOTP mechanism won’t work.

   YubiKey

   For Fedora.

   For Debian.

   ```
   sudo apt-get install yubikey-personalization
   ```

   NitroKey3

   Follow the installation instructions on the official NitroKey website.

   WARNING: *as of April 2024 the official instructions involve using pipx to install the pynitrokey package and its dependencies without any GPG verification! This is not a recommended practice, but will soon be fixed by NitroKey when they start providing release artifacts with detached signatures on their GitHub. Proper packaging and distribution for Debian and perhaps Fedora is also planned for the mid-long term.* **Installing packages using pip or pipx is not recommended!**

   both

   Shut down your template. Then, either reboot your USB VM (so changes inside the template take effect in your USB app qube) or install the packages inside your USB VM as well if you would like to avoid rebooting it.

2. Install qubes-app-yubikey in dom0. This provides the program to authenticate with password and YubiKey / NitroKey3.

   ```
   sudo qubes-dom0-update qubes-yubikey-dom0
   ```

3. Configure your YubiKey / NitroKey3:

   YubiKey

   Configure your YubiKey for challenge-response `HMAC-SHA1` mode. This can be done on any qube, e.g. a disposable (you need to attach the YubiKey to this app qube though) or directly on the sys-usb vm.

   You need to (temporarily) install the package “yubikey-personalization-gui” and run it by typing `yubikey-personalization-gui` in the command line.

   - In the program go to `Challenge-Response`,
   - select `HMAC-SHA1`,
   - choose `Configuration Slot 2`,
   - optional: enable `Require user input (button press)` (recommended),
   - use `fixed 64 bit input` for `HMAC-SHA1 mode`,
   - insert the YubiKey (if not done already) and make sure that it is attached to the vm,
   - press `Write Configuration` once you are ready.

   NitroKey3

   Set up a new NK3 Secrets App HOTP secret by attaching the NitroKey to your USB qube and running the following commands in it:

   ```
   AESKEY=$(echo -n "your-20-digit-secret" | base32)
   nitropy nk3 secrets register --kind hotp --hash sha256 --
   digits-str 8 --counter-start 1 --touch-button loginxs
   $AESKEY
   ```

   Note that the 20 digit sequence can contain any printable ASCII character, e.g. letters, numbers, punctuation marks. The actual `Secret Key (base 32)` is the base32 encoded form of that sequence.

   both

   We will call the `Secret Key (20 bytes hex)` (YubiKey) or `Secret Key (base 32)` `AESKEY`.

   - It is recommended to keep a backup of your `AESKEY` in an offline VM used as a vault.

- Consider keeping a backup of your `AESKEY` on paper and storing it in a safe place.
- If you have multiple YubiKeys for backup purposes (in case one gets lost, stolen or breaks) you can write the same settings into other YubiKeys. For YubiKeys you can choose “Program multiple YubiKeys” in the program; make sure to select `Same secret for all keys` in this case. For NitroKeys you can set up the secret for multiple of them, but you must always use the same NitroKey, because the HOTP counter will be incremented in dom0 as well as the used NitroKey whenever you make use of this method. If you want to switch to a different NitroKey later, delete the file `/etc/qubes/yk-keys/nk-hotp-counter` in dom0 first to make it work with a fresh NitroKey 3. Do the same if for some reason your counters get desynchronized (it stops working), e.g. due to connectivity issues (NitroKey3A Minis are known to wear out quickly).

4. **YubiKey**

   Paste your `AESKEY` into `/etc/qubes/yk-keys/yk-secret-key.hex` in dom0. Note that if you had previously used a NitroKey3 with this package, you *must* delete the file `/etc/qubes/yk-keys/nk-hotp-secret` or its content!

   NitroKey3

   Create the file `/etc/qubes/yk-keys/nk-hotp-secret` in dom0 and paste your `AESKEY` (in base 32 format) into it.

5. As mentioned before, you need to define a new password that is only used in combination with the YubiKey / NitroKey3. You can write this password in plain text into `/etc/qubes/yk-keys/login-pass` in dom0. This is considered safe as dom0 is ultimately trusted anyway.

   However, if you prefer you can paste a hashed password instead into `/etc/qubes/yk-keys/login-pass-hashed.hex` in dom0.

   You can calculate your hashed password using the following two commands. First run the following command to store your password in a temporary variable `password`. (This way your password will not leak to the terminal command history file.)

   Now run the following command to calculate your hashed password.

   ```
   echo -n "$password" | openssl dgst -sha1 | cut -f2 -d ' '
   ```

6. To enable multi-factor authentication for a service, you need to add

   (same for YubiKey and NitroKey3) to the corresponding service file in `/etc/pam.d/` in dom0. This means, if you want to enable the login via YubiKey / NitroKey3 for xscreensaver (the default screen lock program), you add the line at the beginning of `/etc/pam.d/xscreensaver`. If you want to use the login for a tty shell, add it to `/etc/pam.d/login`. Add it to `/etc/pam.d/lightdm` if you want to enable the login for the default display manager and so on.

It is important, that `auth include yubikey` is added at the beginning of these files, otherwise it will most likely not work.

7. Adjust the USB VM name in case you are using something other than the default `sys-usb` by editing `/etc/qubes/yk-keys/vm` in dom0.

Usage

When you want to authenticate

1. plug your YubiKey / NitroKey3 into an USB slot,
2. enter the password associated with the YubiKey / NitroKey3,
3. press Enter and
4. press the button of the YubiKey / NitroKey3, if you configured the confirmation (it will light up or blink).

When everything is ok, your screen will be unlocked.

In any case you can still use your normal login password, but do it in a secure location where no one can snoop your password.

Optional: Enforce YubiKey / NitroKey3 Login

Edit `/etc/pam.d/yubikey` (or appropriate file if you are using other screen locker program) and remove `default=ignore` so the line looks like this.

```
auth [success=done] pam_exec.so expose_authtok quiet /usr/bin/yk-auth
```

Optional: Locking the screen when YubiKey / NitroKey3 is removed

You can setup your system to automatically lock the screen when you unplug your YubiKey / NitroKey3. This will require creating a simple qrexec service which will expose the ability to lock the screen to your USB VM, and then adding a udev hook to actually call that service.

In dom0:

1. First configure the qrexec service. Create `/etc/qubes-rpc/custom.LockScreen` with a simple command to lock the screen. In the case of xscreensaver (used in Xfce) it would be:

   ```
   DISPLAY=:0 xscreensaver-command -lock
   ```

2. Then make `/etc/qubes-rpc/custom.LockScreen` executable.

   ```
   sudo chmod +x /etc/qubes-rpc/custom.LockScreen
   ```

3. Allow your USB VM to call that service. Assuming that it's named `sys-usb` it would require creating `/etc/qubes-rpc/policy/custom.LockScreen` with:

In your USB VM:

1. Create udev hook. Store it in `/rw/config` to have it persist across VM restarts. For example name the file `/rw/config/yubikey.rules`. Add the following line:

   ```
   ACTION=="remove", SUBSYSTEM=="usb", ENV{ID_SECURITY_TOKEN}
   =="1", RUN+="/usr/bin/qrexec-client-vm dom0
   custom.LockScreen"
   ```

2. Ensure that the udev hook is placed in the right place after VM restart. Append to `/rw/config/rc.local`:

   ```
   ln -s /rw/config/yubikey.rules /etc/udev/rules.d/
   udevadm control --reload
   ```

3. Then make `/rw/config/rc.local` executable.

   ```
   sudo chmod +x /rw/config/rc.local
   ```

4. For changes to take effect, you need to call this script manually for the first time.

If you use KDE, the command(s) in first step would be different:

```
# In the case of USB VM being autostarted, it will not have
direct access to D-Bus
# session bus, so find its address manually:
kde_pid=`pidof kdeinit4`
export `cat /proc/$kde_pid/environ|grep -ao
'DBUS_SESSION_BUS_ADDRESS=[[:graph:]]*'`
qdbus org.freedesktop.ScreenSaver /ScreenSaver Lock
```