1. DISPOSABLE VM

I found some interesting ideas on Reddit, Qubes OS Official Documentation, Qubes OS mailling list, Qubes OS unofficial documentation, and I wrote some. The main goal of this document is to provide examples of what is possible to be done with disposableVM.

1.1 Introduction

1.1.1 References

- disposablevm
- disposablevm-customization

1.1.2 What is a disposable VM?

DisposableVM are created for a specific task, so it doesn't contains data from a previous use. Strong points:

- It doesn't contains any previous data — unless you added personnal data in the templateVM, but that is a very bad idea —, so only the data from the current task can be exploited by a malware.
- If it get infected by a malware, the malware cannot persist.

You want to use DisposableVM everytime you can.

1.2 Playing online video

The following bash function will:

1. Create a new disposable VM
2. Run the "youtube-dl" on the given URL parameter
3. Launch the downloaded video with the default video player
4. Destroy the disposable VM once the video is over

```
yt(){
 qvm-run-vm --dispvm youtube-dl "$1" -q --exec 'xdg-open {}'
}
```

Examples:

```
# Download a youtube video
yt e07q7afCakc

# Download any video
yt https://tube.neowutran.ovh/videos/watch/b66d6a30-3d8e-41b7-a⌟
 ↪  a74-43e462bcb643
```

If you want to download/play video that weight more than 1.8Go, you need to increate the Private Storage Max Size of the DisposableVM Template.

1.3 Web browsing

Like nearly everyone, I have some plugins installed on my browser. So it require to have the plugins / config installed directly in the TemplateVM.

1. Create a new ‘web’ disposableVM Template
2. In the ‘web’ template, configure your browser and install the plugins you need.

And now you can use your newly create DisposableVM to browse the web. However, you probably open web page in many VM (from email, rss, chat room, ...). So you need a configuration to automatically open web page in a disposable VM.

Create the file ‘ /.local/share/applications/browser-dvm.desktop’ with the following content:

```
[Desktop Entry]
Encoding=UTF-8
Name=BrowserVM
Exec=qvm-open-in-vm @dispvm:web %u
Terminal=false
X-MultipleArgs=false
Type=Application
Categories=Network;WebBrowser;
MimeType=x-scheme-handler/unknown;x-scheme-handler/about;text/h⌟
 ↪  tml;text/xml;application/xhtml+xml;application/xml;applicat⌟
 ↪  ion/vnd.mozilla.xul+xml;application/rss+xml;application/rdf⌟
 ↪  +xml;image/gif;image/jpeg;image/png;x-scheme-handler/http;x⌟
 ↪  -scheme-handler/https;
```

Due to what I count as a bug, you also need to edit dom0. Add the following line to the file ‘/etc/qubes-rpc/policy/qubes.OpenURL’

```
# Syntax information:
# @dispvm means "The default disposableVM Template"
# @dispvm:XXX means "The disposableVM Template XXX"
@anyvm @dispvm:web allow
```

Set it as default and test it.

```
xdg-settings set default-web-browser browser-dvm.desktop
xdg-open https://www.qubes-os.org
```

If you have a lot of VM and want to apply this configuration to most of them, it can be automated instead of doing that manually for every vm. In dom0, create a new bash file 'browser.sh'

```
cat <<EOF > ~/.local/share/applications/browser-dvm.desktop
[Desktop Entry]
Encoding=UTF-8
Name=BrowserVM
Exec=qvm-open-in-vm @dispvm:web %u
Terminal=false
X-MultipleArgs=false
Type=Application
Categories=Network;WebBrowser;
MimeType=x-scheme-handler/unknown;x-scheme-handler/about;text/html;text/xml;application/xhtml+xml;application/xml;application/vnd.mozilla.xul+xml;application/rss+xml;application/rdf+xml;image/gif;image/jpeg;image/png;x-scheme-handler/http;x-scheme-handler/https;
EOF

xdg-settings set default-web-browser browser-dvm.desktop
```

Then adapt and execute the following command:

```
# For every VM, except "web", "vpn-internal" and "anon-whonix", execute the content of the file "./browser.sh"
qvm-run -all --exclude vpn-internal --exclude web --exclude anon-whonix "$(cat ./browser.sh)"
```

I know there is a tool more "specialized" for this kind of task, called "Salt", but I found it quite complex to use for what I want to do. For anyone reading this with knowledge about Salt, what is the correct way of doing that?

Security notice Like every template VM, if the template VM is compromised, every disposableVM / appVM based on it are compromised too.

1.4 Manipulating untrusted files/data

Same idea as for the web example, set the default application for XXXX to be: create a new disposableVM and open it inside.

Create the file ‘ /.local/share/application/pdf-dvm.desktop’

```
[Desktop Entry]
Encoding=UTF-8
Name=PdfVM
Exec=qvm-open-in-dvm %u
Terminal=false
X-MultipleArgs=false
Type=Application
Categories=PdfViewer;
MimeType=application/pdf
```

Set it as default and test it.

```
xdg-mime default pdf-dvm.desktop application/pdf
xdg-open ./test.pdf
```

1.5 Mounting LVM image

Source

```
#!/bin/bash

#########################################################
# Make a LVM image appear in qvm-block
# Create a disposable VM
# Attach the image to the newly created disposable VM
# Wait until the disposable VM is destroyed
# Remove the LVM image from the qvm-block list
#######################################################

image=${1?Image file is required, example
 ↪  "/dev/qubes_dom0/vm-debian-tmp-root"}
dvm=${2?DVM template name is required, example: "fedora-dvm"}
dev=$(basename $(readlink "$image"))
qubesdb-write /qubes-block-devices/$dev/desc "$image"
list_before=$(qvm-ls | cut -d " " -f1 | sort)
qvm-run -v --dispvm=$dvm --service qubes.StartApp+xterm &
sleep 5
list_after=$(qvm-ls | cut -d " " -f1 | sort)
diff=$(comm -3 <(echo "$list_before") <(echo "$list_after"))
qvm-block attach $diff dom0:$dev
wait
qubesdb-rm /qubes-block-devices/$dev/
```

1.6 Replace sys-* VM

Using static DisposableVMs for sys-*

1.7 Replace some AppVMs

For any VM that doesn't need to persistantly store new data, you can replace it with a static disposableVM. That way, every time you launch it, it is completly clean.

One example from my setup: A qube that have a specific settings (browser configuration, net-vm, firewall and some other), but never need to store new data.