# Qubes Duress - Deniable Qubes Instalation - General Discussion - Qubes OS Forum

17-21 minutes ⋮ 3/11/2021

Undeniably, state can be harmful.
Across the world we see inquiring of data and encryption keys from users and failing to do so promptly, results in serious problem. This example prohibits “{{security,encryption,privacy}}-by-default” approaches.
Qubes can offer reasonable security, but could it be enhanced to hide it’s presence (at least from the common eye).
Duress passwords on login where instead of logging in to the normal user session would take you to a W10 vanilla environment (similar to kali) or even delete the entire state of the system (!) could be a solution.
Deniable encryption in the form of luks volumes without identiefiable headers could be another solution.
HEADs enhancing paradigms where instead of only system integrity, checks for user integrity could also be applied would be best.
A combination of the above, reasonable.

Have anyone attempted similar approach? Do anybody see problems in such approachs?

Interesting. I must admit my ignorance. Leaving here a link on duress codes for someone in the same situation.

I’ve added “Deniable Qubes Instalation” to the title to help users who also didn’t know what Qubes is. Hope that’s ok.

I don’t have much to add, but I’m looking forward to this conversation.

This is an interesting and old, but very complex topic that will, i hope, produce some lively debate and input.

> Duress passwords on login

In terms of how this may play out - it depends on who is making the machine owner act under duress. I can see many facets - some legal, some nation state, some criminal etc. Legal side - some jurisdictions seeing a headerless, crypted partition will be saying “no legal proof, we cant do a thing,” through to some who will be saying “we see what looks like an encrypted drive” and compelling the owner into revealing the decryption keys (Regulation of Investigatory Powers Act 2000 part III in the UK for example). Nation state level, I wouldnt even like to postulate. And Criminal elements after the encryption key? Well, thats the old “heres my friend Mr lead pipe” approach, surely?

Bear in mind any IT forensics team who have any form of - uhm - common sense, will first image the drive and work from that image. For example, law enforcement or similar state services are unlikely to power up your machine and ask kindly for the passwords. They will

have an image made of the drive, then return the drive to the original hardware and seal as evidence (or return the drive to the hardware if it was obtained clandestinely). The work will then be done on that image (or one of a few copies made of it). A duress password would be absolutely useless in such a situation as once that has been used and the data destroyed, any forensics team would be aware by performing a simple comparison of the images.

There is really only a very small, finite “im under duress” window where you may be actually in front of the computer and the only copy of the data is the one on the drive in the machine - some nations can ask you to log into your laptop at the border, so they can inspect the contents before granting entry, for example. Or perhaps there was some adversary performing a home invasion (legal or not) hoping to find the machine in a powered up, non encrypted state (I could see in that scenario that a lock screen duress password may be an interesting conept, as long as the machine user has enough time to actually hit lock)

> Deniable encryption in the form of luks volumes without identiefiable headers could be another solution.

This would be easily visible upon analysis. Lets say i have a 512G drive, and 64G of that is non encrypted, randomOS. Then the rest of the drive is non partitioned/filled with gibberish. It would be pretty clear theres “something” there. Even containers within the partitions are not foolproof (along the lines of truecrypt hidden volumes). There is an old but reasonable paper on the concept here 9

> HEADs enhancing paradigms where instead of only system integrity, checks for user integrity could also be applied would be best.

interesting concept, I am a heads fan. Heads already has the GPG public key of the user embedded in CBFS. Requiring the private GPG key to boot could provide user integrity checking but only as far as “someone has my physical usb key.” Also, the TPM passphrase is required to unseal the luks keys from the TPM (If the user installed that way) or the luks passphrase is needed to boot if the key is not sealed in the TPM… You could go extreme and say “if the machine boots without my GPG key in the slot, then burn it” - but remember what I said above about most situations being where an image of your drive is being worked on. Imagine a PBKAC where the user forgets to put the key in and kill their own data one day. The situations where you would get the opportunity to, under duress, type in a duress password are likely less than you think.

im sure theres plenty more others may comment on this topic. At least I hope so.

“Even containers within the partitions are not foolproof (along the lines of truecrypt hidden volumes).”

Random data at the end of an decrypted drive (with FDE) or somehwere in its middle can be plausible as decrypting random data usually results in random data and FDE is usually initialized with random data. So one would need at least 2 layers of encryption for some plausibility (IIRC that’s what the Veracrypt hidden volumes do).

It’s hard to not accidentally overwrite your hidden data then though.

I think we talked about this somewhere on the forum 1-2 months ago already.

Thanks for the input so far.

Basic goal would be deniability in front of a court of law. There are numerous conteporary examples of authoritative control over citizen. I believe that it is a global phenomenon that even suspicion of violating a law would lead to the seizure of all electronics devices from suspects. Along these lines, imagine passing through the Chinese borders with a laptop running Qubes. Not on my personal experience but a few hours of questioning seem to be the norm (if not worse) and with the current state of things that model remains to be seen where else will apply.
I am aware of the GB law. It's actually scary. Activists are on the most danger. These kind of laws can easily apply to political enemies, and in most case are not even necessary.

Obviously not the duress target, since most users will quickly get outclassed in terms of resources, coersion, expertise vs such forensics teams. But(!), if all I have on my drive is a vanilla qubes (or any other "front-end" vanilla system) and the rest of the drive contains my encrypted data (qube vms) with detached headers all they can know is that there is a high entropy pattern on which an individual can claim that they fresh-installed without populating with zeroes.

If you could provide a link to the discussion would be great. I was unable to spot it.
Adding to the initial question, the matter at hand is one's ability to encrypt without snoopers knowing of the encryption. More of a security/encryption by default approach.

For consideration of existing tools:

- CryptoGrub 7
- Cryptsetup De-LUKS 16
- Denibale Encryption example 12
- BSD login Duress 1
- Linux PAM Duress 5

Building upon these some ideas directly inspired by the above:

- qubes-vm-boot-protect, duress passwords would be desirable for distinct qubes vms(high security ones)
- DeLUKS encrypted primary or secondary pool drive
- Simply disguishing the whole boot process until the point you are logged in on your qubes desktop would be nice.
- the simple kali trick with the transformation to w10. For the purposes of deniability and duress, obscurity would suffice.

Could the qubes community packages infrastructure provide a basis for work on bringing these ideas to qubes?

An idea: a USB device that holds the key in a secure element
(specialized crypto chip), which itself is secured with a password so in
normal usage it would work like a standard 2FA … you need the USB key
and the password to decrypt the laptops hard drive.

The twist: when entering the duress password the USB device irreversibly
deletes the key (secure elements already have this function when they

detect attempts to access the silicon).

So unless the adversary can clone the secure element, which by design shouldn't be possible: it's game over.

I guess the problem here is that all those chips are proprietary and in order to trust it, we would need an open hardware version of it.

there is i recall a device referenced as the Yubi Key. it has similar function as to what you have referenced here. as you have said @sven the firmware is of the largest concerns. from what i remember i have seen chatting about Yubi Key on this forum. is anyone known to have success with the Yubi Key approach? another idea could be implementing veracrypt in a form with the hidden volume style. perhaps some modifications could be added to allow for a system deletion with the
duress passcode entered. a very good consideration would be to add such a feature to the official 4.1 release. i do believe it would draw great usefulness for the community.

to add as i research more now- a plausible cure be containing a qubes distribution within a hidden veracrypt volume. if the character enters a stressful circumstance they would be possible to unlock only the hidden volume which could contain arbitrary but pertaining information. is anyone attempted to try this idea? would it be possible to contain the distribution within a volume such as this? have you seen such a concept completed @sven @deeplow?

Did you consider Librem Key, which has free firmware?

ive just researched librem key and it is very appealing. thx for your insights. do you know if anyone has had success integrating this with qubes?

Looks like it does not work out of the box, but it's a software problem in Qubes:

I would like to see this feature available. To approach this, we need to answer the following:

1. Are there any existing tools that provide deniable encryption?
2. Can these be integrated into Qubes?
3. If 2=yes: What work needs to be done to integrate them?
4. If 2=no: How much work needs to be done to create a solution?

@adw Has created a github issue already on this matter, see:

▼ *Further to this topic*

I would like to see some additional improvement in the current encryption offerings.

Specifically:
Support for layered/encapsulated encryption (option to choose different algorithms/ implementations).
Further, support for hardware keys (ideally U2F challenge response support) - with the option to assign the algorithm also.
The exact same disk encryption functionality offering for the VMs.

AFAIK functionality regarding disk encryption isn't great across the OS-world, so this may

require some dev. work - but it's something I'd really like to see, and I think many would value.

I have seen this, also:
Integrate 2FA/YubiKey into LUKS · Issue #2712 · QubesOS/qubes-issues · GitHub 2

As the separate issues raised here on the thread and on github are all related, I think that these ought to be combined into a unified github project. (That is, of-course, once the above questions have been answered and the Core Qubes Team can allocate some time and set a release target).

As someone who has been forced to surrender their electronic devices at a border crossing for several hours (and wasn't allowed to be present while they were inspecting them), I can definitely say this would be useful for this use case, under the following assumptions:

- The parties performing the inspection aren't too tech-savvy, or are not necessarily interested in a "deep" inspection.
- There is no reasonable threat of torture or physical harm.

If I was threatened with torture or physical harm, I don't know how useful it would be.

I will point out that I have been asked to do this on more than one occasion, at multiple border crossings, and the methodologies and approaches/procedures have differed wildly.

I don't need to imagine this, unfortunately… 😎
As of 2017, there is at least one person at every border checkpoint that is trained in basic Linux Terminal commands (mostly Ubuntu) and Windows Powershell (but thankfully the variance in skill of the officers remains very broad). One guy didn't delete his history logs, and typed in Ubuntu commands in dom0 (`apt` and `lsb_release` were entered 30+ times 😅). There is also a remote unit that can supposedly "remotely" inspect devices at every border crossing (I don't know anything more than this).

They also are aware to some extent of the existence of Qubes OS, however your average border agent still appears to have some difficulty detecting it (thankfully).

They're usually looking for things like Tor Browsers, Crypto Wallets, VPN configs. That sort of stuff.

**SOURCE:** The border guards tell me things.

Indeed, this may be the case, and would be useful when the actors aren't exactly sure of exactly what it is they're looking for.

Would indeed be incredibly handy. "I don't know what that is. It was just there when I installed the OS, officer…"

YES!

I couldn't think of a better place for them 🙂

Not any more 😉

I like. If I ever have enough money, I'd so make this.

I have managed to get it working in Debian templates as a 2FA. Will attempt it with Fedora and dom0 when I have the time.

---

## Usefulness

I seriously doubt duress passwords would be helpful in cases of "I know those files are on this computer, and you're going to give them to me, whether you like it or not *sharpens knife, prepares some sort of venomous animal, and removes hot coals from furnace*", but it would definitely be useful for "Sir/Madam, please step this way, this is just a random inspection…"

True, but I would love the ability to have it, though. 🙃

As long as the users fully understand its limitations and use cases, I think it would be very "peace of mind" feature to have of Qubes OS.

## Potential Other Applications

- Alternate password for remote wipe, "monitor/honeypot mode", masqueraded boot process (the Windows logo and spinny thing, booting to a Windows-looking desktop), removing PCI devices via the kernel on boot, etc. (Or even custom other things)
- Anyone who can be subject to "random inspections" that are superficial (school teachers, boyfriends/girlfriends [not necessarily infidelity, maybe you are trying to hide a birthday surprise? Just saying…], low-level law enforcement, etc.)

In any case, I'd love to help out in making this a reality.

I still don't see much advantage over OpSec on border controls.

E.g. why not hide your data somewhere on the Internet and either take an empty laptop or none at all with you on border controls?

And if you fear tough controls asking you to show that Internet volume, you may still keep your data on the Internet inside a hidden trucrypt volume or use other steganogrophy.

If the Internet is blocked inside that country, go with some super small data storage. SD card? Maybe use Truecrypt hidden volumes/steganography on it. Make sure you can destroy it "accidentally".

So why does Qubes need all that, if you can achieve a better solution by changing your behaviour?

If you're checked on and think that the checkpoint modified your hardware, you'll have to buy new one anyway.

To avoid that the checkpoint notices you mounting that super secret storage volume from some forensically restorable Qubes OS logs (assuming they are hyper sensitive and can

read them), you should probably do that with disposable VMs only and maybe even deploy the disposable VMs in RAM only.
There's an open Qubes issue for the in-RAM thingy, but even that is currently already possible for those people with a bit of work & technical knowledge. If the checkpoint then notices you to have that setup, they might become more interested. But it's always a cat & mouse game…

P.S.: As a first step one would probably just disable the logs monitoring your mounting behaviour…

Touché… 😗 All I can think of is the "inconvenience" of loading an empty machine afterwards. But I suppose that would have been taken into account when considering your threat model…

So maybe a Qubes `contrib` steganography package, then?
*(Just brainstorming…)*

Hide it inside a "smart light bulb" or anything with a microchip in it (toaster, hairdryer, fridge, prosthetic leg, etc.)
*(definitely agree here)*

---

All valid points. Agreed.

Checkpoints aren't actually the only places duress passwords would be useful.

- `xscreensaver` passwords that would put you into a vanilla environment.
- Duress passwords for individual qubes (particularly for work qubes) might be useful for some people, particularly anyone who is *not* particularly tech-savvy

---

I've tried to find the exact clip, but I haven't (yet)…

Basically, James Bond is held at gunpoint, and an attacker has his phone, and asks for the password. Bond then tells him his duress password, and the phone tazers the attacker.

I'm not saying we should make Qubes OS tazer anyone (although, in fairness, that would actually be pretty cool, and totally possible with the right hardware).

But I do think that there would be use cases, even as a `contrib` package…

"All I can think of is the "inconvenience" of loading an empty machine afterwards."

For the tech-savy people Qubes offers [salt] for the basic setup (I need these VMs on my machine configured in that way).

If you're tech-savy & super lazy on loading Internet storage into your VMs, you might want to look into [qcrypt] (disclaimer: I'm the maintainer.).
But it can be observed to be installed on your machine incl. the fully automated configuration of loading your remote volumes (assuming you use qcryptd), i.e. if you don't want to raise

any suspicions at all on checkpoints, it’s probably better to not have it installed and do all of that manually inside a disposable VM or just install it later when you feel safer.

[salt] Salt (management software) | Qubes OS 2
[qcrypt] GitHub - 3hhh/qcrypt: multilayer encryption tool for Qubes OS 11