How do I implement cascading encryption with dm-crypt+LUKS?

JTeller3 ⋮ 43-54 minutes ⋮ 1/21/2021

January 22, 2021, 12:32am 1

Hello Qubes Community,

I am currently wanting to implement a custom configuration of dm-crypt+LUKS that involves cascading encryption during the Qubes installation process. This will allow a setup roughly similar to what VeraCrypt provides on Windows for full-disk encryption, enabling one to use cascading ciphers in the form of something like AES-Twofish-Serpent. I am struggling to figure out how exactly to accomplish this or where to even start, however, and so I am here seeking some guidance.

From what I have figured so far, I will need to nest the LUKS volumes inside each other, perhaps offset to make room for their respective headers, and mount them as loop devices. I understand that `cryptsetup` can set up loop devices for me, but I suspect that I will have to use something like `losetup` to manually set up the volumes as loop devices. If cascading encryption can be implemented without the need for `losetup` or any manual loop device creation, that will be great, but I do not know how that can be achieved or if that is even possible.

I do know that cascading encryption in LUKS should be possible, though, at least in theory and with enough manual configuration. I want to make that a reality, but unfortunately I have found no guides or tutorials describing how to do so.

In particular, I am stumped by what exact options I will need to pass to `cryptsetup` and what their values will be. For example:

- Will I need to `--offset` the volumes to make room for the LUKS headers? Will that still be necessary if they are detached onto an external drive?
- Will the `--offset` need to be at `4096` and `8192`, or something else? If I keep the headers attached, will the offsets be different for each layer?
- Will I need to `--align-payload` at all, as I see some users do on other (different) custom configurations in Arch?

At a more general level, I am unsure about the order of operations:

- Do I start from the innermost volume and work outward, or do I start from the outermost one and work inward?
- If I need to use `losetup`, when should I?
- Where do I start in the whole process?

I admit that this configuration is very complex and definitely beyond my expertise, but I am stubborn about wanting to implement this, all from a `tty` during the Qubes installation process. I have already spent a week working to figure this all out, pouring over `man` pages and familiarizing myself with manual partitioning and LUKS encryption, all with little success. The Qubes Community is my last stop on this journey, however, because I cannot think of any group of people better equipped to help me than you all, and I do not want to bother the Qubes developers themselves with my pet project because they are already too busy performing miracles as it is.

Any help that can be provided to accomplish this goal will be greatly appreciated. If this project is a success and we can document the process of setting this up here, too, this may serve as the first guide on how to do so, which may be a godsend for others in my position who are seeking to implement a strong form of encryption.

I frankly do not want to spend another week fumbling in the dark about this, though, so hopefully you all give me some light at the end of this long tunnel.

Regards,
John

P.S. Apologies for the length. I don't mean to rant. I just want to cover all the bases here so that I don't waste any of your time with an unclear picture of the technical problems I am facing. If you made it down here but don't want to read all the rest, basically all you need to know is how to answer the question in the title and any advice you can give on steps needed to implement it. Again, thank you for your time.

JTeller3 January 22, 2021, 1:55am 2

I noticed that you mentioned cascading encryption in one of your comments recently, @whoami. I hope you don't mind me pinging you, but if you can provide any input about my situation, I will greatly appreciate it.

whoami January 22, 2021, 5:57am 3

Wow, you selected an interesting topic

Well, my cascading comment was more adding a "manual additional encryption" method or software on top. i.e. when you have an encrypted volume or file and you put this again into a VeraCrypt contain or on a hardware encrypted USB device etc. So, nothing which help you here. I am sorry.

Let the geeks pick up your issue.

tripleh January 22, 2021, 3:58pm 4

If I understand you correctly, you want two different encryption algorithms for full disk encryption?

So you distrust a single algorithm for disk encryption?

Please keep in mind:

- There are much more likely attacks than breaking e.g. AES out there.
- Many jurisdictions have laws to force you to reveal the key/passphrase (IIRC e.g. UK and U.S.), some will torture you.
- Doing crypto wrong is often worse than using the common stuff.

So if you're really under state surveillance or so, your time is likely better invested in reading about OpSec (operational security).
Under such circumstances it's e.g. better to not have a laptop at all at border control and leave the data you care about somewhere on the Internet.

Anyway if you're still ineterested in the topic:

- Qubes OS doesn't have full disk encryption at the moment. /boot/ remains unencrypted, i.e. you'd have to solve that first.
- Then you'd have to modify the initramfs with the cryptsetup & loop device method you mentioned.

Actually if you don't insist on full disk encryption, you can simply use cryptsetup inside a VM and you'll enjoy the disk encryption of Qubes OS plus the additional layer from your container. So if you keep all your super secret data in there, an attacker would have to break OS encryption *and* your container encryption. You just need to use `cryptsetup` for that.

If you need your double encrypted data in multiple VMs, there's qcrypt [1] for that (I wrote it).

[1] GitHub - 3hhh/qcrypt: multilayer encryption tool for Qubes OS

JTeller3 January 22, 2021, 9:24pm 5

Hello @tripleh, thank you for your thoughtful input.

Specifically, I want to implement full-disk encryption with a cascading encryption scheme similar to what VeraCrypt provides on Windows. For example, I want to be able to encrypt the drive with a triple encipherment cascade, such as `(AES(Twofish(Serpent)))`. This will also allow the choice of using different hash functions, such as Whirlpool.

An added wrinkle in this is that I want to be able to accomplish this all from a `tty` during the Qubes installation process, which limits me to only the tools made available by the ISO. This should be achievable, since I do not see any reason why a custom configuration such as this is not possible with dm-crypt+LUKS, however manual and difficult the implementation may prove to be. It has proven difficult for me already, which is why I am here, throwing myself at the mercy of the community.

What this setup accomplishes is that it allows one to distrust a particular cipher and hash function combination as a single point of failure. Instead of relying on AES+SHA or Serpent+Whirlpool to protect you, why not rely on both in the event that one of them is successfully cryptanalyzed but not the other? By distributing risk among the algorithms, one can benefit from the standardization and unremarkability that AES+SHA provides even while distrusting it to secure your data. This holds true even when assuming that multiple encryption using different algorithms may be vulnerable to a meet-in-the-middle attack just like multiple encryption using the same algorithm is.

I understand that easier attack vectors than cryptanalyzing a fully encrypted drive exist, such as keylogging or modifying the bootloader or imprisoning you for contempt or even just beating you with a wrench. Cascading encryption with different algorithms is not meant to protect against any of that, but instead to mitigate risk of eventual cryptanalysis, distrust a given algorithm set, and increase the cost of unauthorized decryption. Toward those ends, I believe cascading encryption with different algorithms is an effective security practice.

I also understand that security is only as good as its weakest link, but let us assume that the drive encryption *is* currently the weakest link for which cascading hashing and encipherment is a remedy. Ultimately, it does not matter if it really is for me; what I am asking for is help implementing a proof-of-concept encryption scheme during the Qubes installation process. If successful, the process can be documented as a guide for others to use, who may be in more dire need of such a scheme than any of us ever will be.

Regarding `/boot`, I intend to detach it onto a separate drive, along with the LUKS header for either the outer layer or all layers (depending on how much more difficult the latter is than the former). Qubes OS could thus be installable on a main drive as plausibly deniable full-disk encryption. I did not include that in my original topic post because those are extraneous to the problem. While additional difficulties will likely develop from trying to combine a detached `/boot` and LUKS header setup with manually implemented cascading LUKS encryption, I do not want to further confuse an already complicated problem when a proof-of-concept for cascading LUKS encryption during Qubes installation has not been achieved.

If I am mistaken that `/boot` and the LUKS header for Qubes OS can be detached from the main drive, then perhaps full-disk encryption with Qubes OS is indeed not possible at this time. But even so, my problem of implementing cascading encryption with dm-crypt+LUKS while installing Qubes OS remains. If that is actually not possible either, for some specific reason, then I want to know why it is not so that I do not continue hitting this brick wall.

If you have any guidance on how to accomplish that, I will run with it as best I can. And if you can explain to me why, specifically, implementing cascading encryption with dm-crypt+LUKS during the Qubes installation process is **not possible**, then perhaps I can finally stop wondering why I cannot seem to figure it out.

Thanks again,
John

P.S. Thank you for the link to the interesting tool you created. It may prove useful to me (and others, of course) once Qubes OS is installed. I will bookmark the link for future use.

unman January 23, 2021, 4:29am 6

The Qubes installer is just a somewhat modified Fedora installer, so you
should look in Fedora forums for help with this.
You can detach /boot and LUKS headers - I'm not clear why you think this
will give you plausible deniability.
As to your rationale, I think you are mistaken - the weakest link is
not drive encryption - it's the user. I live in a state where State
officers have the right to demand access to a computer, and such
legislation is increasingly common. Unless you are worried about someone
having access to your drive separately from you, in which case you have
other problems.

If you intend to set up a hidden encrypted volume, then you should be
able to do this from the command line, install a cover system to the
outer volume, and then install Qubes to the inner volume. You shouldn't
need to do anything special here.
An alternative workable approach is to copy a working Qubes install on
to the inner volume - you'll need to do some work in mapper and fstab,
but again, that's standard.
Then on your bootable encrypted drive, you have grub2 primed to
decrypt the inner volume and boot it: again, standard stuff.

It's somewhat difficult to help you because you haven't given any detail
on what you have tried, and how it failed. Nor explained *why* you have
to do this during the Qubes installation process.

JTeller3 January 23, 2021, 10:10pm 7

Hello @unman and company,

I recognize that this is a very, very long reply. I apologize for the length, but feel it is necessary to adequately address all the points raised so far. It probably is not, which is my fault alone. I have split the following into two sections, the first of which clarifies some my attitudes and motivations toward the purpose of this topic and the project it proposes, and the second of which directly addresses how far I have gotten (not very far) and why I want this implementable during a Qubes installation (strongly preferred, but reluctantly optional). If all you care about is the latter, skip the first section and begin after the second section break.

I am aware that the Qubes installer is based on Fedora, but was unable to find anything that can help me from Fedora forums and communities. My best luck has been with the Arch Wiki, whose article and subarticles on dm-crypt have provided me the most insight. Unfortunately, there is not even any mention of cascading encryption, much less documentation or guides, which is why I am here. The closest I have found to that is a brief suggestion of how to do so *in Zulucrypt* for LUKS *containers*, which is found on the Whonix Wiki's "Full Disk Encryption" article. To the best of my knowledge, guides or tutorials for implementing cascading encryption in LUKS using `cryptsetup` simply do not exist. If we are able to accomplish doing so, perhaps one will.

I believe that detached `/boot` and LUKS headers provide plausible deniability *for the primary drive* because the entire drive will be encrypted without any evidence or indication that it is. For anyone inspecting the drive without foreknowledge, it will simply appear to be filled with random data, which can be explained away as having just been wiped in preparation for installing an operating system. This requires no knowledge of the separate drive containing the `/boot` and LUKS headers. I understand this to be a typical method of implementing plausibly deniable full-disk encryption. Regardless, its deniability is not the primary purpose of this project, just a potential effect of it. The main goal, as I stated above, is to accomplish an implementation that does not trust a given algorithm set (such as AES+SHA) as a single point of failure, that provides a method of distrusting AES+SHA while benefiting from its status as a standard ("gray man" encryption), and that can increase the cost of successful cryptanalysis. This practically assists in preventing the unauthorized decryption of the data in the event that it is exfiltrated without the necessary keys or boot drive, such as in the event of theft, or confiscation, or imaging, or even the death of the owner.

I did not state that drive encryption is the "weakest link" and I recognize that the user is often that; you misunderstand me. What I proposed was a scenario in which we assume that it is for a particular security model, which was in response to being reminded that weaker links than drive encryption exist.

Frankly, however, I do not think I need to justify why I am attempting to accomplish this encryption scheme. That is ultimately not relevant to whether such an implementation is possible and successful. As a matter of courtesy, I am entertaining tangential discussions about the security implications of this setup and the threat models to which it responds in order to illustrate the practical relevance of cascading full-disk encryption in certain security scenarios. I have already extensively explored such considerations in my own time and with others, as well, so I can explain its practical relevance to security and identify entire problems and scenarios it helps one solve. None of that aids in accomplishing a proof-of-concept implementation, though, and does not actually get me any closer to figuring out how to do so.

The reality is that VeraCrypt already provides cascading encryption as a matter of due course, at least on Windows, and it can be enabled by simply selecting it as a feature during setup, all without any prerequisite knowledge of how it works beneath the hood or much work beyond that. This allows anyone, even someone with absolutely zero technical expertise, to benefit from extremely strong encryption with minimal change in how they use their devices.

dm-crypt has never had anything remotely comparable to that, which is a crying shame. It is long overdue for a successful, *documented* proof-of-concept implementation of cascading encryption with LUKS. While this may prove to be very difficult and confusing to figure out (it has for me), actually implementing it should be a breeze once a complete guide on how to do so is available. At that point, the whole process can even be automated and integrated into the installation process as a feature that automagically implements it for you, with as little required expertise as is demanded by VeraCrypt on Windows, to the benefit of all. Automating and integrating it can be a project for someone far more brilliant than me, though, such as @tripleh or even the Qubes/Whonix/dm-crypt+LUKS developers.

In my opinion, it is unacceptable that the most popular and common implementation of drive encryption on Linux systems (dm-crypt+LUKS) does not have feature parity with VeraCrypt on Windows. Part of what made TrueCrypt (and its spiritual successor, VeraCrypt) a game changer in my eyes is that it offered cascading encryption at the touch of a button, something that had never been done before in any publicly available, easy-to-use security tool. In doing so, it brought cascading encryption to the masses and set a new standard for state-of-the-art encryption. In light of this, the continued lack of support for cascading encryption in dm-crypt+LUKS—if only in documentation and implementation, and not possible configuration—derives from the security model a previous era, when cascading encryption was not available for anyone with a compatible device to use.

You may disagree with this opinion. I entirely understand why one might; many, including many professional security researchers, probably consider cascading encryption to be "excessive" or only providing diminishing returns. But that is not the point, and discussing the pros and cons of that perspective brings us no closer to it becoming a reality, which is my goal and the reason why I created this topic.

Hopefully now, you and anyone else reading this understand some of the motivations for why I am on this seemingly quixotic journey toward cascading encryption with dm-crypt+LUKS. If you or anyone else here can help me continue it, then I will be forever grateful and will do my best to pay it forward by producing documentation and guidance for how it can be implemented. If not, for whatever reason, then that is unfortunate but understandable and I appreciate the consideration nonetheless.

Now, with that out of the way, let me address your concluding remarks.

Regarding what I have tried, my answer is not much at all because—as I indicated in the original topic post—I do not even know where to begin or how to approach this. I do understand, generally, what tools will be necessary to accomplish the task and how the commands are likely to be crafted; but beyond that, I am stumped. Maybe I am overthinking it. I already understand how to accomplish full-disk encryption with dm-crypt+LUKS and detached encrypted `/boot` and LUKS headers, since all of that is documented in the Arch Wiki in a sufficiently clear format that it can be adapted and followed during the installation of nearly any Linux operating system.

The hard part is figuring out how to implement cascading encryption along with this and where to fit that part in the order of operations, especially since this appears to be uncharted territory without any corresponding guide or tutorial to consult. Thus far, I have surmised that I will need to manually set up multiple LUKS volumes nested inside each other and probably manually mount them as loop devices using `losetup` (unless `cryptsetup` can handle it for me, like it can in non-cascading setups); but I lack the expertise to understand how exactly that will look as a series of commands, and in particular what options may need to be included in them (such as `--offset` and `--align-payload` in `cryptsetup`). I will also need to generate a custom `initramfs` and perhaps more beyond that to implement the full-disk encryption part, but that is ultimately extraneous to the main goal of cascading encryption.

Once mapped out and defined, the whole process should be pretty straightforward (and appears to be from a high-level overview), but at the moment it is hopelessly confusing to me. In response, I am pouring over any information I can find to familiarize myself with the fundamentals and intricacies of these tools, such as the `man` pages of `cryptsetup`, `losetup`, and `lvm`; the aforementioned Arch Wiki documentation; conference presentations and tutorials on how to use these tools; guides on how to automate decryption of multiple LUKS volumes and set up LUKS-encrypted DVD/BD disks using loop devices; and even complaints about LUKS in the comments section of a Schneier blog article on TrueCrypt's audit! And yet, I am still here, seeking guidance.

I want to implement all this during the Qubes installation process because:

1. It should be possible to implement using only the tools provided by the ISO.
2. It should not require any prior setup through a different device, which introduces unnecessary complexity and risk, especially if the threat scenario involves access to only one trusted device, an empty SSD and USB, and a DVD+R burned with a known safe copy of the Qubes ISO.
3. It should not involve any custom tools from outside the Qubes ISO, whose trustworthiness and efficacy are unknown or in doubt (in contrast with reliable tools such as `cryptsetup`, `losetup`, and `lvm`), whose compatibility and interoperability with Qubes OS are unknown, and whose involvement may block any chance of it being included as an automated feature in future installation processes (including in Qubes OS).
4. It should be relatively easy and straightforward to implement by simply following the instructions on a guide (such as one I intend to create out this project), much like a standard Arch installation.
5. It should be a setup anyone installing Qubes OS can implement if they want as an advanced example of a custom installation, whose documentation page can house the guide on how to do so.

Although I appreciate how limiting it may seem to do this entirely from a `tty` during the Qubes installation process, I believe that working within that constraint will ultimately simplify the process and ensure the greatest compatibility with existing Qubes installation procedures. If a custom tool is ever designed to automate this process, it can be included in future Qubes ISOs and integrated as an advanced feature, but the chances of that happening probably depends on an initial manual implementation of it as proof that it can be done.

If we remove this constraint, however, does that appreciably ease the difficulty of accomplishing this? At this point, I will even work toward a proof-of-concept implementation that requires additional tools and setup prior to initiating the Qubes installation process. Whatever it takes to securely implement cascading encryption using dm-crypt+LUKS is fine by me because at least that it a start.

I am under no illusion that I am requesting a lot here (and typing a lot, too), and that my goal is very ambitious (to put it nicely). But I would not be here unless I thought this community could lend me the expertise I am sorely lacking and needing to accomplish this. I just hope I am right, and that there is enough patience to go with it, because otherwise I may just have to shelve this project entirely as a incomplete failure. I can always try the PrivacyTools forum, as well, but if the wizards here lack the magic (or maybe just the will), then I worry those wizards will too.

If there is any further information I can provide, please do not hesitate to ask.

Weary regards,
John

The first thought is, make sure the kernel has the right crypto modules loaded…I doubt something like serpent is compiled-in, so you'd need to `modprobe serpent; modprobe twofish; etc`.

At the basic level to triply-encrypt a file container:

```
truncate -s 32MB encrypted.container
truncate -s 2MB encrypted.header
cryptsetup luksFormat encrypted.container --header encrypted.header
[configure first passphrase]

sudo cryptsetup luksOpen encrypted.container container1 --header
encrypted.header
[enter first passphrase]
[now you get /dev/mapper/container1 as a device]

cryptsetup luksFormat /dev/mapper/container1
[configure second passphrase]
sudo cryptsetup luksOpen /dev/mapper/container1 container2
[enter second passphrase]
[now you get /dev/mapper/container2 as device]

cryptsetup luksFormat /dev/mapper/container2
[configure third passphrase]
sudo cryptsetup luksOpen /dev/mapper/container2 container3
[enter third passphrase]
[now you get /dev/mapper/container3 as device]
```

…now use /dev/mapper/container3 as the physical volume for a new LVM volume group to install on, sample output:

```
$ lsblk -f
NAME                FSTYPE   LABEL UUID
MOUNTPOINT
loop1
└─encrypt1          crypto_        a9f7dfa6-f210-4e9e-914c-ca0ea2d53987
  └─encrypt2        crypto_        20356cc3-eb74-4ea1-bfe5-00d9264a35ba
    └─encrypt3      LVM2_me        HkpCwV-Z9og-XvE7-nCHm-Qu1v-Gyk1-b1Tnxm
      └─triplelvm-rootfs
        ext4          727787ab-d334-4b25-b7d8-b7847cd4db76              /
tmp/yah
```

Things you'd still have to do:

1. Give `cryptsetup luksFormat` the right `--cipher` option for each step to get something other than AES
2. Above uses passphrases, and potentially three separate ones. Do you want to type 3 different passphrases? Or 3 same shared passphrases? Or keyfiles? If keyfiles, then you have to think about filesystems in each layer or doing some absolute offsets working on a raw disk container.
3. Maybe /etc/crypttab will handle this so that an initramfs generation does the right thing; I don't know.

Seems overly complex with little benefit, but perhaps this is a starter. I bet there are many other gotchas hiding.

tripleh January 24, 2021, 9:08am 9

Hi again John,

understanding your goal is key to providing support about the right methodology. That's why @unman was asking.

E.g. you threw in plausible deniability which is not necessarily related to cascading encryption algorithms.

Veracrypt supports "hidden volumes" which appear as random data at the end of another encrypted volume.

Denying the outer volume as being encrypted is unplausible as it is random data to the observer and that indicates encryption. Denying the inner volume is somewhat plausible as the outer volume was initialized with random data. So it might be there, but also might not. If you however don't actively use the outer volume for anything, it becomes unplausible from the metadata that this is the only volume.

This approach is indeed pretty hard to achieve with dm-crypt at the moment.

Anyway you said that you only care about cascading encryption, which @icequbes1 already explained. Essentially cryptsetup just opens a plaintext device from the ciphertext device and you use the plaintext device as next cascaded ciphertext, decrypting it to another plaintext device and so on.

You can probably do that during installation on the tty and install to the last plaintext device. I'm not sure whether the installer will manage to get fstab & initramfs generated correctly, but ideally it should.

BR
David

unman January 24, 2021, 12:34pm 10

I should point out that this isn't cascading encryption in the way that
veracrypt has it - in veracrypt each 128 bit block is encrypted with one

cipher, then *that* is encrypted with another, and so on.
What's offered here is encrypting one *volume*, then encrypting the encrypted volume and so on. That's doable, but somewhat different.

Also, you can (fairly) easily use hidden volumes with cryptsetup, if that's what you are looking for.
Both of these are well documented approaches.

cryptsetup does support truecrypt and veracrypt encrypted volumes, so I suspect that you can more readily encrypt the volume using *their* cascading ciphers, and then use cryptsetup to map those volumes.

icequbes1 January 24, 2021, 6:08pm 11

Yes, I should have made that clear. This "cascading" encryption is actually 3 block devices enveloping one another, so it incurs more overhead as device mapper/kernel has to map an inner block to the outer block, then encrypt, and it does this 3 times to encrypt a single block if there are 3 layers.

Veracrypt's cascading encryption encrypts a single block with 3 different ciphers and that is written to the physical disk, something like `aes(serpent(twofish(block)))`. Compare that to what I mentioned which does something like `dm-remap1(aes(dm-remap2(serpent(dm-remap3(twofish(block))))))` where each mapping might be passing various sector boundaries on the underlying block device and not to mention the LUKS headers for the inner containers also take up space, etc.

I also thought this would be the best approach if "cascading encryption like Veracrypt" was truly sought, but it did not appear that `cryptsetup` could perform initialization of a Veracrypt-encrypted disk from a quick look at the man page which appears to be one of @JTeller3's requirements - something that can be done natively without an external toolset.

But the question asked for dm-crypt and LUKS implementation. I could not find information of LUKS' ability to supper cascading ciphers in the same manner as Veracrypt. A very crude and likely bad implementation (I am not a cryptographer) might be to write a kernel module that acts as a meta cipher and performs the cascading encryption itself by executing the appropriate inner ciphers. Veracrypt mentions its cascading encryption uses 3 separate keys for each cipher, but keys are derived from the same passphrase; not sure how much control you'd have over that if you went the "kernel module that implements cascading ciphers" way.

Whatever the case, I reinforce @tripleh and @unman - make sure your goals and needs are met by the approach you seek; deniability isn't so easy to achieve and the cascading encryption likely makes that even harder to achieve.

JTeller3 January 24, 2021, 7:09pm 13

(Note: I deleted this post because I accidentally sent it as a reply to another user, rather than as a general topic post. Upon resubmission, I am told that the body is too similar to my previous post. I hope this note redresses that and allows it to go through, and that it does not cause duplicate notifications.)

Before I begin, I want to apologize to everyone for my previous reply. Although I did not intend to appear gruff, I suspect I came across that way. I want to apologize in particular to @unman and @tripleh, whose thoughtful commentary I have appreciated but who may have not received my response as showing it.

@icequbes1:

Thank you very much for the suggestions. After marveling at its simplicity, it has given me some cause to believe I have been overthinking this. For example, `losetup` is likely not necessary, since `cryptsetup` can probably handle it for me if I let it. I suspected as much, but my lack of experience with custom configurations with `cryptsetup` gave me pause.

I also find your speculations about how to achieve actual cascading encryption feature parity between VeraCrypt and dm-crypt+LUKS to be illuminating. The care with which you have given toward my specified constraints has been humbling. Thank you.

Regarding passphrases, I intend to implement a cascading full-disk encryption using dm-crypt+LUKS on the main drive, whose encrypted `/boot` and headers are detached onto an external flash drive, which is itself encrypted with dm-crypt+LUKS except with *attached* headers (to avoid an infinite regress of header detachment). The encryption for the main drive will be handled through keyfiles, which will be sequentially passed to the volumes during boot, and the only time I will enter a passphrase is to unlock the LUKS volume encrypting the boot drive (e.g. "Gold Solution" in this article).

The boot drive could itself be encrypted in a cascade, but I cannot think of an implementation of this that does not require entering a passphrase for each layer, since otherwise it will either be an infinite regress of keyfiles or they will be stored unencrypted with the header (which defeats the purpose). Regardless, a single encryption of this drive, perhaps with algorithms different from the rest, should be more than sufficient when the main drive is a triply-encrypted full-disk encryption.

I anticipate that automating the decryption cascade can be accomplished with `/etc/crypttab` and/or a custom `initramfs`, as is described here and here.

Is this "overly complex"? It depends on your threat model. For most purposes, yes it is and probably "excessive", too. Nonetheless, I think strong encryption—including cascading encryption—should become a new standard because it helps to practically address problems such as eventual cryptanalysis and algorithm trust while practically facilitating security resilience and post-mortem privacy in offline scenarios. I appreciate your willingness to entertain this complexity, even though you may not consider it a worthwhile pursuit.

@tripleh:

You are right about methodology. I have tried to be clear with my motivations and goals, without confusing it further with other moving parts that I hoped could be put aside for the purposes of this topic, but I have not been very successful.

I did not want to present the full scope of the task I am trying to accomplish—triply hashed and ciphered cascading full-disk encryption using dm-crypt+LUKS and with detached `/boot` and LUKS headers on an encrypted boot drive, as implemented during the Qubes installation process in a `tty`—because I considered some of those parts to get in the way of establishing a clear and basic implementation of cascading encryption with dm-crypt+LUKS. Once that has been established, integrated it into the rest of the setup should prove to be somewhat easier.

I also did not want to make this topic be about others helping me implement an extremely specific custom configuration, but instead to define a general guide on how to implement cascading LUKS encryption while installing Qubes. While I will welcome any guidance on completing the rest of my project, I intentionally restricted the scope of this topic to avoid muddying the waters. Maybe that too was a mistake, especially if I run into any further problems during the process, but there is not much I can do about that now.

The main result I want to accomplish with this is a proof-of-concept that can be documented and provided to the community as an advanced custom installation, complete with a step-by-step guide on how to do so. While I may use this setup myself if it proves to not be too much of a burden, my intent has been to enable users of Qubes OS (or Linux distributions) to use it themselves as they see fit, whether that includes me or not.

I agree that plausible deniability through a detached `/boot` and LUKS headers is irrelevant to cascading encryption with dm-crypt+LUKS. I intended to leave it out, for the reasons described above, but considered it appropriate at the time to include it once you brought up the lack of full-disk encryption support in Qubes OS above. In retrospect, that may have been the wrong approach.

(Sidenote: Is this actually true? I interpreted that to mean no feature support, but that it is still implementable by detaching `/boot` and the LUKS header. If FDE for Qubes OS is actually not possible even then, then that is an important obstacle in the way for my larger project.)

An example of plausibly denying a fully encrypted internal drive, in scenarios where the boot drive is not known but your ownership of the drive is assumed, may be to claim that the drive was recently wiped in preparation for installation, but installation has not yet occurred. (To my knowledge, many companies issue their employees devices with similar setups when traveling internationally.) This can be claimed at an airport or border crossing, where the justification is that you do not bring your data with you during travel and part of your travel security includes travelling with an empty drive on which you will install an operating system once you arrive. You can then have a live USB of, say, Ubuntu or Linux Mint or Windows 10 that you can plausibly claim will be installed, but that you have not yet done so. This may be because you did not have the chance and you were in a rush for your flight, or because you understand that device inspections like this can occur and you fail to see the point in showing off a freshly installed operating system.

Perhaps that is not plausible, though, and I am mistaken about that.

The only way I know that this can be verifiably disproven is by comparing multiple images of the drive over time, thereby revealing activity (and potentially defeating XTS mode), but that is not the likely scenario in which you attempt to deny it and is generally not the threat model deniable encryption seeks to address.

Anyway, thank you for all your input. It has been clarifying for me and for this thread.

@unman:

You are correct that VeraCrypt implements it differently and it was lazy of me to omit that fact. Such an implementation of dm-crypt+LUKS would potentially require modifying `cryptsetup` itself or otherwise designing an extensively modified fork of it, however, and so I quickly abandoned that as well beyond the scope of any personal pet project, especially for an amateur non-cryptographer like me. I suppose that if we define cascading encryption in the way TrueCrypt/VeraCrypt implements it, which properly distinguishes it from multiple encryption, then what I am trying to accomplish is not even that. But it is the closest I imagine one can achieve with the tools available, even if it is a poor man's version of it.

Given these differences, do you think that is what most significantly limits the utility of this custom configuration? Or, put differently: Since it is not even like VeraCrypt's cascading encryption, does that invalidate its proposed security benefits by comparison to the point that nothing is gained? I still maintain that even volume-level "cascading" encryption is an improvement for the reasons I have already stated, but maybe you disagree. I would love to read your thoughts on the matter. @icequbes1, if you have any on this, too, I'm all eyes.

I already knew coming into this that I was in over my head and likely possessed some misunderstanding, but I thought I could figure it out along the way and rectify those errors. So far, I have (somewhat), but I did not anticipate just how far in over my head I was and how mistaken I may have been. For that reason, I regret creating this topic, since it seems to have been based on some shaky premises that may not be able to withstand scrutiny from those with far more expertise than I have.

This topic was originally intended to explore the implementation of a type of cascading encryption that can improve upon the security and address some problems I have identified with the current security model of standard single encryption. It was intended to be implementable without much difficulty during Qubes installation, and documented as an advanced type of custom installation, which may prove to be helpful for anyone who wants such a setup up or believe they need it to mitigate their threat model. It was **not** intended to be a mere exercise in extreme patience for others in the wordy process of disabusing me of my befuddlement. But I feel that is all it has been.

I still thank you all for your help and will still try to implement this, even though I may have approached this all wrong. It has been a learning experience for me, at least.

Plausible enough to do what? Pass through airport security? In a country where decrypting your drives is mandatory on demand this will get you through, but they will know anyway. That's assuming you are not targeted. If you are targeted personally they might as well take even actually empty disk with some random data on it and claim they have reasons to suspect encrypted contraband. Use it as pretext to screw with you. In which case your best bet is not to take any disks with you at all, or fill one with zeros and fresh ubuntu installations.

This will only work in a country that can legally force you to reveal your keys, but only after they can successfully prove the encryption have taken place, and they have to have no other intel except for the drive AND they have to actually play by all the rules. That's a loooot of boxes to tick. The amounts of thought you seem to be putting into this would suggest you have actual data to protect and not just doing all this for sport. In which case you really might want to reconsider the suggestions in this thread people made and maybe some other ones too. Like maybe having only one ordinary LUKS encrypted disk with a whole bunch of only slightly sensitive data on it that you'd be more than happy to unlock if under duress.

I know this is probably not what you would like to read and it's something that I struggled for a very long time personally, but in the end I had to accept that random looking data on a disk is just not a practical form of plausible deniability in all but the most ideal scenarios.

Apart from that however I do agree that LUKS could be way more flexible. At the same time I would not expect anybody to put so much extra work for something so few people would actually use. That's not to say I do not believe in security benefits of such nested encryption.

What you are trying to do is (imho) better than veracrypt's way of doing things in all aspects I can think of except for convenience (the type of convenience you sacrifice security for). With vera you have one password that unlocks one 256 bit key that unlocks all your data. The OS you unlock it in have the access to both the keys and the data for the whole duration of operating on the data. In contrast, with your solution you have let's say three layers of encryption, each with different cipher, but also with different keys and different passwords to unlock them (if you want). So especially in Qubes what you can do is you attach your disk to some vm and unlock it with the first password, then take the resulting block device and attach it to another vm, repeat the process. Not only none of the vms will ever know all of they keys or passwords, you can even try to achieve multiple layers of "plausible deniability" in the sense of having random blobs of data or seemingly empty space containing your lower level containers. There is another advantage in that you can segregate your data in sensitivity levels. Say you have 100GB disk and 70 GB of data, 5 of which is extremely sensitive data that you only need to access seldomly, 15 is slightly less sensitive data that you need to access every couple of days and 50 is still sensitive but not critical data that you want to use daily. In that case you can have first (one or two) layer(s) of encryption spanning the whole disk, but then partition the resulting decrypted block device and place the least sensitive 50GB of data on it and the rest of the space goes for the next layer of encryption. At the next level you place the 15GB of more sensitive data and leave some space for the last layer of encryption when the critical 5GB will live. This way you only unlock the sensitive data if you really need to, you can also copy the still encrypted containers in lower layers to other systems without decrypting them at all. If you are using luks headers (detached or otherwise) you also get all their advantage on every layer, meaning you can change/add passwords for any of the containers independently of each other, even when they are decrypted and mounted.