Defeating the Cobo Vault Pro’s Self-Destruct Mechanism

Nick Johnson

Follow

The Startup

--

1

Listen

Share

Cobo recently offered to send me their Vault Pro, a hardware wallet focused on “see what you sign” security, which they’re presently adding Ethereum support to. They didn’t ask for anything in return, and I assume they expected I’d give it a try and maybe tweet about it.

Unfortunately, they sent it to someone who used to run a hobbyist electronics company, so I did a bit more than that.

Packaging and First Impressions

The Vault Pro comes nicely packaged, and includes the Vault itself, a rechargeable battery (via USB-C, inconveniently placed so you can’t charge it while using it), a separate 4xAAA battery compartment, presumably for emergency use, and various instruction cards and quick-start guides.

The package has tamper-evident seals on it, but one of them had come unstuck in the post without leaving its “void” marks behind on the box, so I’m sceptical how effective they are. Cobo tell me they’re looking into this, and improving their seals and their application for future batches.

The device seems solidly made, and the batteries clip on satisfyingly with a magnetic clasp that also makes them easy to remove. As advertised, there are no communications ports, just a fingerprint reader and camera on the back. There’s also an SD-Card slot hidden under the battery compartment, which their site says is for firmware upgrades.

Setup and UX

The setup process will be familiar to anyone who’s owned a hardware wallet before. You’re asked to create a new wallet or import an existing one; in either case there’s the familiar set of 24 mnemonic words to write down or enter. The touchscreen makes this much easier than the Ledger’s two button interface, but the screen is just a bit small for the full keyboard to be fully comfortable. This is the case when entering passwords, too — I always had the feeling that I was going to fat-finger adjacent keys.

The Vault runs a cut-down version of Android, which seems a sensible choice given the capabilities of the hardware. There’s no home screen, though — just Cobo’s built in application, which uses common UX patterns and so is pretty easy to get used to.

Ethereum Support

The big thing Cobo are working on at the moment — and likely the reason they sent this to me — is Ethereum support. They’ve supported Ether transfers for some time now, but are adding web3 integration, via a fork of MetaMask, and support for smart contracts.

Trying this out required me to download the latest firmware and flash it to the device using a MicroSD card, which was fairly straightforward. Cobo provide instructions on how to verify that the update package matches the sources published on GitHub for the APK (Android App) and secure element firmware, which I haven’t yet done — but it’s to their credit that this is something they support.

After updating and installing their fork of MetaMask in my browser, I’m able to configure a “read-only wallet” on the device, which gives me a QR code to scan in MetaMask. After doing so, my Vault account is displayed in MetaMask, and I can interact with web3 apps using it.

Signing is accomplished by MetaMask showing a QR code (or for longer transactions, an animated sequence of QR codes), which I scan with my Vault; then, once I’ve signed the transaction, the Vault shows a QR code with the signature, which I scan with MetaMask. I tested this out with the ENS app and it worked without issue, though most transactions are still just as opaque as they would be on Ledger, not showing the transaction payload:

For some transactions, it's able to decode the payload and show it to the user; for example, this ERC20 transfer:

The Cobo folks tell me that they're working on support for more ABIs, and support for users to load their own set of ABIs via SD card. They're also considering options for including signed ABI data in the transaction QR codes, or other options for transaction transparency. They're also working on getting their changes to MetaMask accepted into the main extension — which will be essential for making this work long-term.

One of the things Cobo advertises is transparent signing — you should be able to examine and decode the QR code data to check it's doing what you think it is. Let's try it. Here's the contents of the MetaMask-generated QR code for one of my signing requests:

Cobo uses an encoding of their own making called UR encoding', designed to make the most of a QR code's text alphabet. Using their JS decoding library gives us this:

'xfp' is the master key fingerprint from BIP-32. The rest is clear enough. Decoding the tx data with ether.js gives us:

As expected, this is the data for the first transaction shown above.

After signing, the Vault generates:

Which decodes to:

Fairly straightforward, and seems to be doing what it says on the tin.

Security & Teardown

All of this is very interesting, but probably not why you're reading this article. How secure is it, and what does it look like inside?

The Vault has attestation keys loaded onboard — meaning that the secure controller has a keypair that was loaded at the factory, enabling it to sign messages asserting that it's a "real" Vault. The secure controller is designed to make it difficult to extract the attestation keys from the device, even destructively. If opening the device without detection is difficult, and replacing it with a lookalike is prevented by the attestation keys, you can be fairly sure that it you receive a device, and it passes the authentication check, it's secure.

Cobo make a big deal of their physical security against supply-chain and evil maid attacks; they havearticles and videos about its self-destruct mechanism, designed to wipe the secure microcontroller if anyone ever attempts to open the device. If effective, this means that you can be fairly certain that your device is intact when you receive it. How good is this security?

First, let's take a look at the published materials. Their article says there's a button battery powering the self-destruct, and that it uses a continuity sensor with the display to detect if it's been opened. Fortunately, they are kind enough to publish partial schematics and BoM (Bill of Materials). Let's see what we can learn from those.

Here's the relevant part of their schematic — the detection circuit:

The pin on the left here labelled 'DET0' leads to the pad that connects across the back of the screen. In normal operation, that connection grounds DET0, but if it goes open-circuit, because the screen has been removed, R1101 is a pullup connected to the button-cell, which pulls that pin high, turning on Q1101, which in turn enables Q1102, allowing power to flow from the battery, through D1102 to the same power regulator normally powered by the rechargeable battery.

In other words, interfering with the tamper sensor will wake the device up. The "secure element" is also shown on the schematic, albeit without its true name and with a number of connections missing:

VCC5V, here, is the power supply generated from the backup battery (or the rechargeable one, in normal operation). An odd choice, since the rest of the schematic shows that the secure element uses 3.3v logic levels — why does it need a 5V supply? This 5V supply isn't used anywhere else, though it may be used to power the 3.3v regulator; that part isn't shown in the schematic provided.

A connection to DET0 isn't shown, but we can safely assume (and thepublished firmware confirms) it's connected to one of its general-purpose IO (GPIO) pins. The firmware also shows three other pins used as "passive" tamper sensors; these ones can't wake the device up from sleep, but can detect tampering of some kind while the device is active.

This is the point at which a 'real' adversary would have bought several Vaults, and would take one apart as a 'sacrificial' device so they can plan how to attack the others. I don't have that luxury, so instead I took my Vault to my local Veterinarian, and asked them to XRay it for me. They were happy to oblige, and gave me this:

Interesting! This definitely isn't the device in the video and the photos from the article; in those photos, the PCB takes up the whole device, but in these it's only in the top part. Presumably this is a newer version?

Here’s a closeup, where I’ve labelled the components we can decipher:

“PMIC” is the 5v power regulator. We can also see a number of “cans” — shielding enclosures used to cut down on RF interference or to make it more difficult to tamper with components. The 5v regulator appears to be under one, and there’s a couple of overlapping ones in the middle of the board — on opposite sides of the PCB, no doubt.

Since the X-Ray sees through everything, we’ve no way of knowing which side the battery is on — we can only hope it’s the outside, not the side facing the screen. We can clearly see the wires from it come down to the bottom left corner where they connect to a pair of pads, though, which gives us all we need.

A little work with a Dremel and we have a hole in the back, through which we can see the wires, just where they should be:

(Sorry for the poor image quality; I haven’t managed to locate the camera adaptor for my microscope since I moved house.)

A check with the multimeter confirms this is the backup battery wiring. After snipping the red wire, we can take the screen off and be fairly sure that it can’t trigger any kind of tamper detection. A little work with a heatgun, some prying tools, and some IPA, and presto:

There’s some sort of cover protecting the electronics. Removing it reveals how the tamper sensor operates on this version of the device:

Two pads at the edge of the PCB are each divided in half, and two pieces of conductive fabric on the cover contact them, bridging them. Removing the cover will break the circuit.

After putting a blob of solder on each to defeat the tamper sensor, and reconnecting the battery, we can boot it up again and see if we’ve been as clever as we think we have. Will it notice that its guts are spewed out in front of it, or will it think everything is still okay?

Success! It hasn’t detected our tampering, and boots up without warnings or errors, ready to sign transactions.

Now, if I were an Evil Maid, or someone conducting a supply chain attack, this is what I’d do next:

1. Find the boot mode selection pins for the main processor and change them to boot from an external device. If I can’t do that, just replace the EMMC chip with one of my own.
2. Load an alternate Android firmware with a modified copy of the Vault app. This app can behave as normal, except when signing certain transactions of my choosing — in which case it will show the user an innocent description that matches what they expect, but actually sign my malicious transaction.

2b. Alternately, for a supply-chain attack, modify it to always generate one of a few keypairs I control — meaning that I already know what account my target will ‘generate’ before they receive their device.

1. Take a second Vault and remove all the components from it — no need to worry about the tamper sensor here. Put my modified Vault inside the sacrificial Vault’s pristine case, and send it on to my victim.

So, what could Cobo do better? This is difficult; realistically, a sufficiently well resourced and determined attacker will always win, if they have their hands on your hardware.

One option would be to put the battery on the side of the PCB facing the screen. This would require a substantial redesign, but make it a lot harder to clip the wires.

A much more comprehensive solution would be to connect the battery directly to the secure MCU. Many MCUs support backup batteries for this purpose, and some deliberately store keys in volatile memory that is wiped if power is cut. This would be a “hard core” option that puts a strict limit on the lifetime of the device, but would make an attacker’s life a lot

harder. With the MCU powered 24/7, other tamper protections can be used — for example, a thermometer can detect attempts to desolder components. Any attacker would be forced to try and compromise the device while it’s powered, and without using heat.

If a rechargeable button cell was used, this could be charged from the main battery when it’s available — so rather than dying after 3 years or so, you’d merely have to use it about that often to keep it alive.

Teardown

I’m not

engaging in a supply chain attack, but I am interested in what’s inside, so let’s have a closer look at the PCB. This is the point where we give up on booting this device again, and I start taking things apart.

As expected from the X-Ray, the bulk of the PCB is protected by a can. A smaller one protects the 5v regulator — likely to reduce the RF interference it emits. Let’s unclip the main can and take a look underneth:

Mostly this is as expected from the published schematics. There’s the Mediatek MT6850 System-on-Chip, and a Kingston part that combines eMMC and SDRAM, along with various support components. In the top left is the secure microcontroller — but its identifying marks have been lasered off! They really are determined for us not to know what this is.

This seems a bit odd, to be honest — secure MCU manufacturers typically require NDAs to access their datasheets and tooling, so it’s no surprise they can’t tell us a lot about it, but the manufacturer generally won’t forbid you from revealing what part you’re using. If they did, all they would have to do is not label them in the first place. It also contrasts with Cobo’s generally open-source approach to the rest of their platform.

Let’s see what we can figure out what it might be. We can figure out a fair bit about it:

- It’s a 5x5mm QFN40.
- It’s some kind of “secure microcontroller”.
- It uses a 12MHz main crystal.
- It uses 3.3v logic for its IOs, but accepts (or at least tolerates) 5V on what appears to be a supply pin.
- We know from the build files that it’s an Arm Cortex M0.
- We know from the firmware that it has hardware-level ECDSA, RSA and RNG support.

Searching common supplier databases, there’s not many hits for this criteria. Our best match is theMAX36010-BNS-T, a “security supervisor” from Maxim. It meets most of our criteria, but it’s unclear if it even includes a programmable MCU (much less an Arm Cortex M0), or whether it has circuitry to support a 5v supply. On the other hand, it’s got built-in tamper detection and support for all the algorithms we know the firmware uses, and it’s the right form-factor, which makes it our best candidate.

The back of the board is fairly straightforward. It has the expected battery and pager motor, the SD card slot, as well as yet another can:

Under the can is nothing particularly exciting: the power circuitry for the main CPU.

Next Steps

The next steps in a serious security analysis of this device would be to start taking apart relevant parts of the circuit, measuring them and tracing nets on the PCB, to construct a schematic we can compare to the published one to identify the unknown parts of it — particularly the tamper detection circuits that are unpublished. I can speculate about what those are, but I don’t think it would be terribly difficult for a determined adversary with similar resources to me to identify and bypass them — especially since the device is inactive in this state, and the pins of the secure MCU can be exposed and traces cut or pins strapped to VCC or GND as required.

Conclusion

Despite the ease with which I was able to bypass its tamper-detection circuitry, the Cobo Vault Pro seems well designed in both hardware and software. The fact it has protections against supply-chain and evil-maid attacks puts it a step above most of its competition, and raises the bar for such attacks — but there is certainly room for improvement in that area to reach the levels of security it aspires to. By more clearly targeting frequent users who want the highest level of protection — and are prepared to deal with the inconvenience that brings — they could produce a device that stands head-and-shoulders above the current competition.

I’m a little sceptical of the security value of using QR codes for transmission of data. Still, the UX is good — no messing with getting USB devices working — and if they succeed in getting their changes to MetaMask merged, or otherwise add support for web3 apps via a solution such as WalletConnect, this device will be a viable alternative to the Ledger and Trezor, with a much nicer UI that seems likely to be further improved.

Which is a shame, because the one I have appears to be non-functional and in pieces.