Keylabs Inc

1712 Carey Ave Ste 100
Cheyenne, WY 82001, USA
Phone: +1 (720) 310-0633

Cypherock X1 Audit

Executive Summary

The Cypherock X1 is an innovative wallet that uses many hardware and software security best practices and even features several security firsts that we have not yet seen in other wallets. These include hardware attestation through manufacturer BIP39 derived signatures, the use of several JavaCard based NFC as a kind of multifactor authentication at the time of signing and Shamir Secret Sharing for storing the bip39 seed phrase of the user. Unlike most other wallets, the Cypherock wallet leverages the security of several independent integrated circuits on multiple devices. This includes the X1 Cards (“the cards”), as well as the STM32L4 that is used in conjunction with the ATECC608A (secure element) on the X1 wallet (“the wallet”). The security of the overall Cypherock wallet is based on the security of several devices working together. As a result the compromise of a single device or card is insufficient to compromise the seed and/or funds stored on the wallet. Additionally, the JavaCards are EAL5+ certified and the ATECC608A, as well as the STM32L4 do not currently have any known and publicly documented hardware vulnerabilities, which means they can also be considered reasonably secure.

Cypherock quickly provided fixes for all the findings. These fixes were subsequently verified by Keylabs.

Table of Contents

1. Overview

An audit of the X1 wallet hardware and firmware was conducted to identify potential security issues in this device. As the X1 wallet uses a consumer grade microcontroller it is reasonable to assume that this will be the weakest link in the Cypherock. However, because it is secured by the 608A and provisioned at manufacturing time, the attack surface is in fact quite limited. For example, trivial attacks such as replacing the STM32L4 with a malicious STM32L4 is not possible because of how the STM32 is paired with the 608A. Moreover there are currently no known publicly documented attacks against the STM32L4 or the ATECC608A.

1) Threat Model

Architectural vulnerabilities are vulnerabilities affecting the overall architecture of the hardware wallet.

- In practice, architectural vulnerabilities encompass many forms of supply chain attacks, for example, replacing any component on the device.
- Because Cyperock provisions the X1 Cards during production, it is likely that these will not be exploitable in practice.
- Because Cypherock programs both the STM32L4 and the ATEC608A during manufacturing and both of these in turn generate a pairing key for their communication, it is impossible to simply replace either of these components, for example, with an STM32L4 running malicious firmware.
- The signing keys, derived from a Cypherock specific BIP32 seed, also ensure software security and secrets that are then loaded into all of the components.

Firmware vulnerabilities are vulnerabilities affecting the software that runs on the hardware wallet.

- Firmware vulnerabilities can affect the overall security of the wallet, in particular vulnerabilities in the bootloader.
- Since the Cypherock utilizes a Shamir Secret Sharing protocol for splitting the cryptographic seed, exploiting the physical wallet requires also exploiting at least one of the X1 Cards.

- Because the X1 Cards are EAL5+ certified and run a vendor certified OS, whilst the Cypherock specific code is running in an JavaCard applet, it's reasonable to assume, that an attacker will not be able to exploit the device in practice as long as industry best practices such as PINs and counters are used.

Hardware vulnerabilities are vulnerabilities affecting the underlying hardware components of the hardware wallet.

- The STM32 family is known to be exploitable. Such an attack would allow an attacker to for example downgrade the X1 Wallet to RDP1 and read RAM
- However, these attacks do not apply in practice to the STM32L4 since it lacks an external Vcore voltage that can be exploited with hardware. Moreover there is reason to believe that PCROP capable STM32 microcontrollers are less exploitable in practice, which the STM32L4 is.

Physical vulnerabilities are vulnerabilities affecting the hardware design of the hardware wallet.

- Physical vulnerabilities include ease of access to the hardware.
- The device tested as part of this assessment lacked any sort of potting to protect the surface of the PCB. However, the final version will include conformal coating, which will make direct physical access to the circuit more difficult.

Software vulnerabilities are vulnerabilities affecting the host software that runs on the PC or smartphone and communicates with the hardware wallet.

- Software vulnerabilities were not considered as part of this audit.
- Software running on the host can be easily updated and fixed. Hence, software vulnerabilities were not analyzed as part of this audit.

2) High-level Overview

The X1 Wallet (the Device) is made up of one MCU, the STM32L4 and two peripheral ICs: the ATECC608A which provides device authentication. Additionally an NXP PN5321 provides NFC communication to the smart cards.

The sole purpose of a cryptocurrency hardware wallet is to prevent leakage of the seed, or derived key, during operation of a cryptocurrency signing operation. Cypherock has chosen to create the seed and then split it among a quorum of NFC cards (X1 Cards). The X1 Cards are running a custom Java Card applet and have not been investigated further as part of the audit.

There are several attack vectors to recover the seed, but the one to focus on is attacking the STM32L4 from either hardware or software because ultimately, the seed is exposed on this MCU. This MCU is not a MCU rated for security. According to ST, only the Secure Boot and Secure Firmware Update (SBSFU) has received the SESIP Level 3 certification, whereas the chip itself has only received the lowest self-assessment from ARM PSA level 1.[1] In general this means that sensitive operations on the ST32L4 should be kept to a minimum.

3) Hardware Review

While the rating of the smart cards may have an EAL5+ rating, the entire security of this system is as strong as its weakest link which is the STM32L4. Thus, the two most important defensive mechanisms are the hardware protection set against debug readout and the software code quality. The STM32F2 has a well-known RDP bypass. In 2017, ST, presumably in response to some of the hardware attacks, added a feature called Proprietary code read out protection (PCROP)[2]. PCROP ensures that flash sectors are *execute only* and is essential to ensure proper glitch protection. PCROP ensures that memory continents within protected sectors cannot be

---

1 https://www.st.com/content/ccc/resource/sales_and_marketing/promotional_material/flyer/group0/75/95/53/70/8d/2e/4d/79/flyerstm32trust/files/flstm32trust.pdf/jcr:content/translations/en.flstm32trust.pdf

2 https://www.st.com/resource/en/application_note/an4968-proprietary-code-read-out-protection-pcrop-on-stm32f72xxx-and-stm32f73xxx-microcontrollers-stmicroelectronics.pdf

accessed via the BootROM Bootloader (STM32 Bootloader). This is especially the case when code execution can be gained within the application code, for example, by only rewriting part of the flash, then executing it. Additionally, glitches on chips running at RDP2 of the STM32 family have been known to induce RDP1 behavior, re-enabling the embedded BootROM Bootloader and yielding a device that behaves identically to a chip at RDP1. This means that an attacker can relatively easily gain access and utilize the BootROM bootloader for reading sensitive data from the device. The attack could read the code out for example to discover an exploit. The firmware should utilize the STM32 PCROP feature, which is available on the STM32L4. This feature is not set by the Cold Card[3] nor the Trezor T. The Trezor T sets Write Protection on the sectors utilized by the Bootloader, however. In both cases, the bootloaders of the Cold Card and the Trezor T are open source and not proprietary. PCROP, would ensure that a glitching attack would not succeed in reading any data from these memory regions.

4) Firmware Review

Hardware Security is only as good as the underlying firmware. For example, a more appropriately secure MCU, the LPC55S69 with the latest ARM Cortex for TrustZone-M, PUFs, hardware accelerators, and more, has a DFU bootROM bug that bypass secure boot. As a result, any applications running on this device, offering significantly more security features than the STM32, are now vulnerable. Similarly, in a hardware wallet, that is likely to not be updated in the field, the firmware quality, especially all drivers that interface with the external world and the bootloader, are especially critical.

Several vulnerabilities were identified in the firmware. For example, the random number generation function, which is a bedrock primitive for cryptography, ignores result codes from both of the hardware random number functions[4]. It ignores the random value from the

---

3 https://github.com/Coldcard/firmware/blob/d7c41ce88ee06864583574d87b25e3edf4573041/stm32/bootloader/storage.c#L503

4 https://github.com/Cypherock/x1wallet_firmware/blob/761a8ce86ed7a797ffd285ecd11c9db8dbcb96da/common/libraries/util/utils.c#L275

ATECC608, for which there is no authentication from the random command and can easily be spoofed, and it ignores the STM32 RNG which can timeout and fail.[5]

Additionally, there is no clear Software Bill of Materials (SBOM) therefore it's not readily apparent what third parties' libraries are used, what versions, and if they have been modified or not. For example the Microchip CryptoAuthLib being used is from 2017. Additionally, this library is so dated, it does not even have a version number. There have been security relevant patches to this library since 2017. The current supported release is 3.3.3 released in November, 2021.

Similarly, the bootloader also does not have a SBOM but uses third party libraries. There are additional examples of security critical code that does not adequately check return values such as in the RandomDelay function.[6] We recommend creating an internal document with security relevant SBOMs for both the bootloader and firmware and evaluating them as part of developing an overall threat model.

5) Key Management

The Cypherock uses a key derivation chain for supply chain security, except using NIST P256 curves as that is what is supported in the ATECC and the NFC cards. Cypherock shared their internal key management and provisioning architecture with Keylabs. Additionally, Keylabs has reviewed the device provisioning authorization documentation which is publicly available[7]. Cypherock is adequately protecting the provisioning root keys.

---

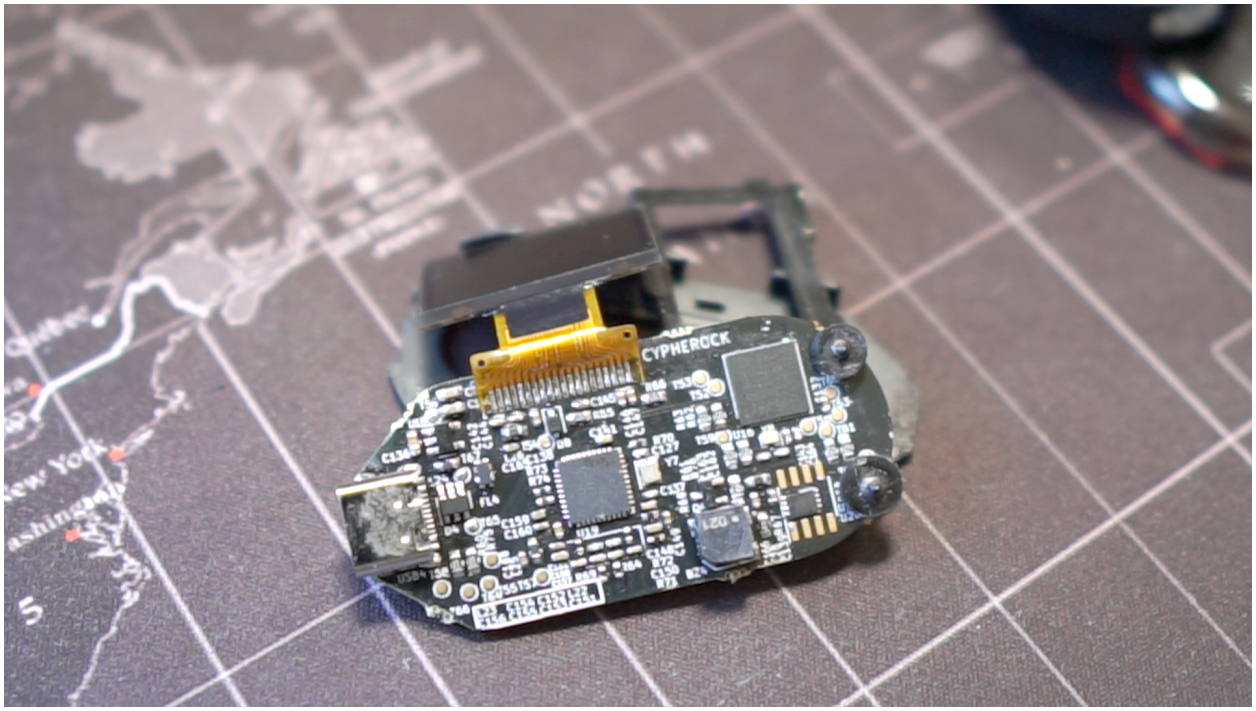
5 https://github.com/Cypherock/x1wallet_hal_stm32/blob/024f8a631d9b62cd0223e413ba605fc70bcb3a22/Drivers/STM32L4xx_HAL_Driver/Src/stm32l4xx_hal_rng.c#L657

6 https://github.com/Cypherock/x1wallet_bootloader/blob/579c4c507d335848acb75fb277a1c1103c882f2a/Application/Bootloader/random_gen/crypto_random.c#L47

7 https://github.com/Cypherock/x1_wallet_firmware/blob/main/docs/device_provision_auth.md

2. Findings

1) Test points easily accessible on the wallet (Severity: Low)



There are numerous test points on the devices that are visible leads to all the relevant communication between the main microcontroller, the NFC interface and the ATECC. Though not directly exploitable, such test points do make it significantly easier to sniff and interface the device. More importantly such test points can be utilized to build a programming jig that can quickly and easily interface with the device as part of a physical evil maid attack and/or as part of a supply chain attack. Interfacing to an already manufactured device is so straightforward that it may be difficult to detect such malicious access as part of a forensics analysis by the manufacturer or the user.

2) ATECC608A placed with two footprints (Severity: Low)

The ATECC608A has two footprints on the device. Essentially, the larger footprint is even more accessible and easy to access, effectively providing test points for the ATECC608A and further reducing the amount of effort required to interface to the device. Since the larger package is not used on the device, the larger footprint pinout should be removed.

3) PCB Marking readily available (Severity: Low)

Though they don't directly have a security implication, for a device that is not going to be serviced in the field, the PCB marking should be removed in the final manufactured version. This can be as simple as omitting the PCB solder mask markings on the side of the device that contains the microcontroller and security relevant components, whilst, for example, leaving a device board revision, manufacturer and device names and copyright notices on the reverse side. In particular such markings make any test points that are left on the device particularly easy to identify and group.

4) Device does not use newer ATECC608B (Severity: Low)

Microchip released the ATECC608B, which they describe as a “security-enhanced version of the ATECC608A”[8]. It’s not clear what the security enhancements are, but Microchip states they are “implemented in the device [and] are largely behind the scenes and are not directly observable during normal operation.” Unfortunately, there is no open source analysis of what these changes are but Microchip recommends the ATECC608B for new designs.

5) No Potting on Security-Relevant Circuitry (Severity: Low)

There is no potting material inside the enclosure. The device is not meant to be user-serviceable therefore, there is no reason it should be opened. While epoxy potting material can be removed, it is incredibly tedious and risks destroying the device while removing the material, which also helps protect the key split. Additionally, it provides environmental protection to the components on the PCB as well. The final release version of the hardware will utilize conformal coating. Though this is not the same as epoxy coating, it is a good tradeoff in practice.

[8]http://ww1.microchip.com/downloads/en/Appnotes/Migrating-from-the-ATECC608A-to-the-ATECC608B-DS40002237A.pdf

6) Device Lacks Tamper-Evidence, Tamper-Resistance and Tamper Circuitry (Severity: Low)

Both the case and the device PCB lack tamper evidence. For example, the device keys are not erased through opening the physical device case. The device continues to operate nominally even though the physical integrity of the surrounding case has been compromised. This helps to protect against hardware implant attacks as well as aiding in detecting evil maid attacks. Production devices will be ultrasonically welded, which will in practice make opening the device more difficult.

7) Functions should use strnlen, not strlen (Severity: Low)

Some functions[9] within the X1 wallet firmware use strlen instead of strnlen. In the majority of cases this can easily be fixed with find and replace.

8) Device PIN recovery (Severity: Informational)

Cypherock implements a Proof of Work algorithm in the case of a forgotten PIN.[10] This is of course a security critical function, however upon review of this algorithm, we are not sure it is needed. Presumably, the point of this process is that it prevents a denial of service from a malicious user who guesses random PINs on the device. Unfortunately, this approach seems needlessly complex. After about 12 incorrect entries, the delay to enter the next PIN is incredibly long, which effectively permanently locks the device. Additionally, the approach constantly writes to the same flash sector. This is because the status of the proof of work is updated in flash. The flash page only has a write endurance of 10k writes. Therefore, even if the authorized user can recover the PIN, after waiting quite a while, the device may later catastrophically fail when a flash write occurs. Therefore, it's not clear if this is better than just hard-locking the wallet after 5 or so guesses.

[9]https://github.com/Cypherock/x1wallet_firmware/blob/28f729745892a5278e65b0047937b151a0fdbe12/src/level_four/core/controller/verify_wallet_controller.c#L75

[10]https://github.com/Cypherock/x1wallet_firmware/blob/v0.3.12/docs/cylock__proof_of_work.md