ERA Wallet Security Audit

Dec 30, 2024

Prepared by:

Keylabs

Nedos Consulting EMEA FZ-LLC
FDRK3801
Compass Building,
Al Shohada Road,
AL Hamra Industrial Zone-FZ,
Ras Al Khaimah
United Arab Emirates
contact@keylabs.io

Table of Contents

ERA Wallet Threat Model

Summary

The hardware wallet threat model outlined in this section was specifically adapted to account for the features of the ERA Wallet. Three main adversary types are considered: remote attackers who can compromise host devices, local attackers with physical access and hardware expertise, and insider threats who may abuse privileged access during manufacturing or distribution. The model details various threats including physical access attacks, supply chain compromises, remote malware attacks, communication interception, side-channel attacks, firmware vulnerabilities, and key extraction attempts. These threats can manifest through various attack vectors such as malicious software, evil maid scenarios, supply chain tampering, social engineering, authentication bypasses, firmware exploitation, insecure communications, and both physical and memory-based attacks. To counter these threats, the model recommends implementing strong physical security measures, device verification mechanisms, secure boot and firmware processes, multi-factor authentication requirements, robust encryption, and regular security audits. The model emphasizes the importance of comprehensive documentation and regular updates to address evolving security challenges. This model was used and referenced as a basis for conducting the remaining aspects of this comprehensive security audit.

Adversary Profiles

Remote Attackers: Actors capable of remotely comprising the host or mobile phone that is used to connect to the wallet and leveraging access to it.

Local Attackers: Actors with permanent or temporary access to the wallet. These actors generally have expertise in hardware hacking, knowledge of hardware vulnerabilities of the wallet and/or vulnerabilities in the current firmware version on the wallet.

Insider threats: Actors with authorized physical and/or remote access to the wallets who may misuse their privileges, for example during device manufacturing or provisioning or by shipping malicious software or firmware.

Threats

Physical Access/Evil Maid Attacks: Temporary or permanent physical access to a user's wallet. Sufficient for a malicious actor to interact with the device.

Supply Chain Attacks: Software, firmware or hardware tampered with during manufacturing, development, distribution, or delivery. Generally in conjunction with Internal threats.

Remote Attacks, Malware and Software Attacks: Malicious code on the host or mobile phone capable of interacting with the hardware wallet.

Man-in-the-Middle (MitM) or Replay Attacks: Intercepting or manipulating the communications between the wallet and the host software as it's being sent over the wire (USB) or via wireless protocols, such as NFC or Bluetooth.

Side-Channel Attacks: Extracting information from unintended channels like power consumption or electromagnetic emissions.

Firmware Vulnerabilities: Exploiting vulnerabilities in the firmware, including the upgrade routines and bootloaders to recover or compromise the wallet and/or seed.

Key Extraction: Techniques to recover the seed, private key or seed phrase at rest. This may include having to brute force the PIN space.

Downgrade attacks: Techniques to downgrade the software an/or firmware to a previous vulnerable version. For hardware wallets this is generally enforced by a bootloader.

Attack Vectors

Malware: Compromised hosts and mobile phones used to interact with the wallet stealing the user's pin and/or preventing transactions or directing them to different addresses.

Evil Maid Attacks: Temporary physical access to the device, sufficient to swap the device, reflash the device, but not necessarily to physically modify the device.

Loss, Theft or Destruction: physically losing access to the hardware wallet and the cryptographic seed by physically losing, having the device stolen or destroyed.

Supply Chain: Compromised hardware introduced during manufacturing or distribution and/or compromised firmware and/or software delivered via updates.

Social Engineering: Tricking the user into entering the pin incorrectly or sending funds to the wrong address.

Authentication Bypass: Insufficient or poorly implemented mitigation of brute forcing or bypassing the authentication scheme entirely.

Firmware Exploitation: Identifying and exploiting vulnerabilities in the firmware allowing for the seed to be recovered.

Insecure Communications: Intercepting data during communication between the wallet and the host as well as between different ICs on the wallet.

Fault Injection and Physical Attacks: Several forms of physical attacks against electronic components can cause them to operate abnormally, potentially bypassing authentication and enabling features that compromise device security, such as debugging.

Secrets in Non-Volatile Memory: Physical attacks against the device often result in the full extraction of the Non-Volatile Memory (NVM) contents which may include the seed.

Secrets in Volatile Memory: It is common to be able to recover the volatile memory contents of devices.

Mitigation Strategies

Strong Physical Security: Wallets should implement tamper detection and should have a case that provides sufficient tamper-evidence for the user to see a wallet that has been tampered with.

Device Verification: Implement mechanisms for users to verify the authenticity of their own wallet, such as nicknames and device pairing. Additionally users should be able to rely on firmware signatures to check the authenticity of their device.

Secure Boot and Firmware: Employ secure boot to verify the firmware prior to execution, regularly update firmware to address vulnerabilities. Since wallets may be stored, for example in a drawer, it's important to offer firmware updates whenever they're plugged in.

Multi-Factor Authentication: Require multiple forms of authentication to access the wallet. All interaction with the wallet should require the user PIN or Passphrase or at least a button press to confirm.

Encryption: Communications between the host or mobile phone and the wallet should be encrypted to prevent malware from MitM the communications. The seed should also be encrypted at rest on the device.

Regular Security Audits: Conduct regular security assessments of changes to the device firmware to ensure that the firmware mitigates all known attacks.

Documentation Recommendations

Create detailed documentation outlining security practices, threat mitigation strategies, and incident response procedures. The threat models should be specific to the particular context and product, and they should be reviewed and updated regularly as the threat landscape evolves.

ERA Wallet Architecture Assessment

Summary

This architecture security assessment evaluates the ERA Wallet's security measures across multiple threat vectors. The analysis was performed against pre-production devices with several changes being subsequently implemented. This section evaluates the pre-production device against attack vectors identified by the threat model. These include malware through QR code and wireless communication implementations, safeguards against Evil Maid attacks through device authentication and tamper detection, recovery mechanisms for loss or theft, supply chain attack prevention through device attestation and traceability, defense against social engineering through trusted display features, mitigation of authentication bypass attempts through attempt limiting, secure communication requirements for both wireless and PCB-level interactions, resistance to fault injection and physical attacks, and protection of secrets in both volatile and non-volatile memory. After evaluating all available STM32H7 options, the STM32H753 was recommended, a processor with enhanced security features. Additionally, it was recommended that a secure element be integrated, with the ERA Wallet now leveraging the Microchip ATECC608C. This secure element implements monotonic counters for improved PIN protection and glitch mitigation and also is responsible for firmware verification.

Malware

The primary benefit of a hardware wallet is its ability to shield users from malware that could infiltrate their computer or mobile device. The physical separation through a second device, utilizing a microcontroller with an independent execution environment, significantly reduces the susceptibility of the hardware wallet to compromise through malware on the host in particular. The ERA Wallet addresses this concern by utilizing QR code scanning and wireless communication. One point to consider is the implementation of a QR code or out-of-band update mechanism remains secure and necessitates user confirmation. Additionally, QR-code libraries are notoriously susceptible to exploitation through vectors such as buffer overflows. Such an attack could result in code execution on the wallet, compromising the integrity of the device. This can be mitigated with additional secure environments or multiple cores since such an exploit may not exploit the execution environment containing sensitive data. The integrity of the hardware wallet can also be undermined by malicious firmware updates. While the hardware wallet cannot independently validate the accuracy and absence of vulnerabilities in a firmware update, it can verify the authenticity of a hardware update through the manufacturer's signature. For verifying firmware authenticity a secure element can also be utilized as part of the signature verification. This ensures the security of the firmware supply chain security by implementing a secure delivery mechanism for updates.

Evil Maid Attacks

Hardware wallets must protect against instances of temporary access, commonly known as Evil Maid Attacks, in which a malicious actor gains unauthorized access to the wallet for a short period of time when it's left unattended. In such attacks, the malicious actor might replace or manipulate the device's hardware and/or firmware. The primary objective typically revolves around extracting the device's PIN. To counter this type of attack, the device must authenticate itself to the user and possess the ability to identify and alert about any tampering attempts. The ERA Wallet includes the necessary hardware to support this functionality. It is recommended wallets present a distinctive pattern, serial number, or equivalent identifier to users for wallet verification. Before the user enters the PIN, the device should consistently provide information about its tamper status. Both software and physical tampering need careful assessment to develop effective countermeasures against this threat.

Loss, Theft or Destruction

The safety of the user's keys and sensitive data against loss, theft, or damage is a critical feature of any hardware wallet. The wallet should incorporate a recovery solution, preferably involving an offline backup. Commonly, this involves documenting the seed phrase generated during wallet creation onto a paper backup. Consequently, the wallet must also facilitate the import of pre-existing seed phrases. Given the tangible nature of the wallet and its susceptibility to loss, theft, or damage, a resilient recovery mechanism becomes imperative.

Supply Chain

In the context of a supply chain attack, a malicious actor typically manages to gain physical access to the device and configure the wallet prior to it reaching the user. The prevailing form of attack involves pre-loading the wallet with a seed phrase that is known to the attacker. By knowing the seed phrase, the attacker has control over any funds stored on the addresses derived from that specific seed phrase. For an unsuspecting user, the device simply seems prepared for use and they remain oblivious to the attack.

Advanced supply chain attacks can involve the physical alteration of device components, where malicious components replace original components on the device. For instance, this might include substituting the microcontroller with one that is less secure or even counterfeit. A key objective in establishing a secure supply chain revolves around ensuring the traceability of manufactured devices. The process involves monitoring the device's progression throughout manufacturing and documenting status at every stage. This practice serves to hinder defective devices from reaching end users while also creating a comprehensive record of the total manufactured units. Moreover, it empowers manufacturers to precisely produce the intended quantity of units to be both manufactured and supported. Furthermore, this approach effectively thwarts attempts to create unauthorized or cloned devices, as such activity can be detected during the device authenticity verification process. Device attestation can be performed from the factory until the device reaches the user. For this the STM32 provides several immutable device keys, which can be combined along with a secure element in device attestation.

Social Engineering

In the absence of a trusted display, a hardware wallet becomes vulnerable to social engineering attacks aimed at coercing the user to perform unintended actions. Such attacks could lead the user to inadvertently authorize a malicious transaction. It's important to highlight that when dealing with smart contract transactions, there might be insufficient on-screen information for the user to comprehensively validate the transaction visually. In simpler scenarios, like transferring funds between two addresses, users typically only need to confirm the network and the recipient's address. The fidelity of the ERA Wallet display offers enough resolution to verify transactions. Nevertheless, the quality of the display's user experience is crucial in assisting users to effectively conduct transaction verification. Cumbersome display and interaction mechanisms can significantly diminish the overall security of the device. Even when employing a hardware wallet, the complex data involved in many smart contract calls often makes it challenging to visually encompass all the necessary data for evaluating the validity of a smart contract call.

Authentication Bypass

The challenge with hardware wallets, and embedded devices in a broader context, lies in their lower computational power compared to the attacker's CPUs and/or GPUs. Consequently, the standard security measures based on password-driven key derivation functions, commonly applied in cloud and desktop environments, do not translate well to such embedded systems. Notably, they can be subjected to brute force attacks on machines with more powerful computational resources.

As an alternative, these systems should hinge on the concept of limiting the number of unsuccessful attempts. Numerous techniques exist to hinder or prevent repetitive attempts, and these strategies will be assessed within the context of the ERA Wallet. Ensuring the wallet maintains a record of attempts is critical for the overall security of the hardware wallet.

Insecure Communications

Intercepting the communications between the host and the wallet, or between different integrated circuits (ICs) within the wallet, can lead to a compromise if the transmitted data is unprotected. If data is transmitted without encryption, it could potentially be exploited. Given that the ERA Wallet plans to include NFC and/or Bluetooth careful

consideration should be given both to the Over-the-Air (OTA) communications as well as the PCB level communications between the ERA Wallet MCU and any wireless ICs.

It's common to for example utilize Bluetooth ICs for unencrypted serial communication, even though the communication channel itself can be configured securely. This could be partly mitigated through an effective tamper response system. A significant point of interest pertains to the permanence of the tamper detection mechanism and whether it can be reset. This factor could potentially give rise to intriguing attack vectors.

In a manner akin to transmitting data in plaintext over buses, information can also be sent without encryption through wireless means, such as utilizing protocols like Bluetooth Low Energy. It's essential to exercise a comparable level of caution when dealing with any form of wireless transmissions.

Fault Injection and Physical Attacks

Fault injection involves deliberately introducing anomalies into the normal operation of a system, aiming to circumvent or temporarily disable security measures, through the introduction of temporary or transient faults. Notably, various instances of Electromagnetic Fault Injection (EMFI) and Voltage Glitching attacks have been demonstrated on hardware wallets. These attacks might target either the application code or the immutable embedded ROMs. Targeting the ROM or BootROM often implies that the compromised code cannot be rectified through field patches.

Secrets in Non-Volatile Memory

In a stateful design such as the ERA Wallet, the hardware wallet is expected to protect the seed as it is stored on the device in Non-Volatile Memory (NVM). If the wallet is lost or stolen, an attacker will attempt to recover the seed because if the seed is recovered, then funds can be transferred with a software wallet. Fault Injection and Physical Attacks may also be leveraged to yield the contents of the NVM. If the contents are sufficiently encrypted, extracting the NVM contents will not compromise the wallet seed.

Secrets in Volatile Memory

Just as with secrets stored in NVM, the lifetime of sensitive data in volatile memories should also be carefully considered. The seed is stored and loaded in the MCU. However if sensitive data such as the device private key are loaded into RAM, this may be susceptible to attack as was the case for many other STM32-based wallets, such as the Trezor. The simplest solution here is to reduce the lifetime of sensitive data in volatile memory and using memzero to clear memory after use.

Additional Hardware Recommendations: Main Processor

The STM32H7 is subdivided into cores with security features and cores without. The initial design of the ERA Wallet utilized the STM32H743AII6 (STM32H743, single M7 core, 169 BGA Package, without security features). Several potential upgrades to this MCU were under consideration including the STM32H757 and STM32H755 (Dual M7 + M4 core, with security features). Eventually the STM32H753 was selected, in part thanks to this preliminary analysis. The table below from the STM32H7 user manual lists the additional security features available for different STM32H7 MCUs.

Table 2. Availability of security features

Security feature	STM32H750xB and STM32H753xI	STM32H742xI/G and STM32H743xI/G
Embedded flash memory (FLASH): – Flash Secure-only area	Available	Not available
Security memory management: – Secure access mode – Root secure services (RSS)		
Cryptographic processor (CRYP)		
Hash processor (HASH)		

Security features of the STM32H75x and STM32H74x

In particular, the SECURITY flag allows for a simplified form of secure execution. Such an environment along with the additional flash features provides the primitives for secure pin verification and decryption of device secrets, but also the possibility of hosting application environments with limited access. The secure flash features may also provide additional security in light of JTAG and bootloader vulnerabilities that are common to several other families of STM32s, though this would need to be verified. The CRYP and HASH processors can be used for implementing many cryptographic

primitives such as PBKDF2. It was also determined that the STM32H7 allows disabling the system memory bootloaders completely by remapping the reset address for both bootmodes to flash. System memory bootloaders are a common attack path against the STM32 and this provides a significant security improvement.

The dual-core STM32H7 model was also considered, which consists of an ARM Cortex-M7 and ARM Cortex-M4. Though not directly security related, the Cortex-M4 core can significantly improve battery life and offload many tasks, such as updating the display. The low power core could in theory also handle much of the communication with other chips such as NFC and Bluetooth. Such communication protocols may be more susceptible to injection of malicious packets. Since the Cortex-M4 presents a separate execution environment exploitation of this core won't necessarily lead to sensitive data within the Cortex-M7 core from being exposed. As a rule of thumb, the Cortex-M7 can do everything the Cortex-M4 can do and more. It was determined that the memory of the Cortex-M4 core cannot be sufficiently segmented and hidden from the Cortex-M7 core. Hence, the STM32H753 was selected in the end.

Additional Hardware Recommendations: Secure Element

Initially a secure element was yet to be selected for the ERA Wallet. Several factors were considered including price, performance, commonality and how likely they are to be unaffected by any supply chain constraints (i.e. that they will in fact be available for a long time). For this reason, one of the most ubiquitous secure elements was selected. Most importantly, the secure element must implement a monotonic counter. For this reason, the ATECC608 family was selected for the ERA Wallet, though Infineon Optiga Trust M was also considered. The ATECC608 (and specifically the ATECC608C) feature monotonic counters, that greatly reduce the complexity of implementing features such as pin roll-back protection, glitching mitigation. They also provide primitives for firmware attestation and verification that were later implemented in the ERA Wallet firmware. Lastly other alternatives were proposed including arm-based secure elements. These have the advantage of being separate execution environments and separate code bases. However, they don't implement hardware monotonic counters, which is the most critical feature a secure element provides on a hardware wallet as this can be used to implement features such as a pin failed counter and exponential backoff.

ERA Wallet
Hardware Security Audit

Summary

This security evaluation of the ERA Wallet pre-production devices identified four informational findings and four low-impact findings. No critical findings were discovered during this assessment. The low-impact findings indicate minor vulnerabilities that can be addressed through planned mitigations in firmware and in changes to the production devices, while informational findings highlight potential areas of improvement that do not directly affect the overall security of the device. The evaluation confirms that the ERA Wallet Hardware is fundamentally secure, with specific recommendations for enhancing its resilience in the final production design.

Findings

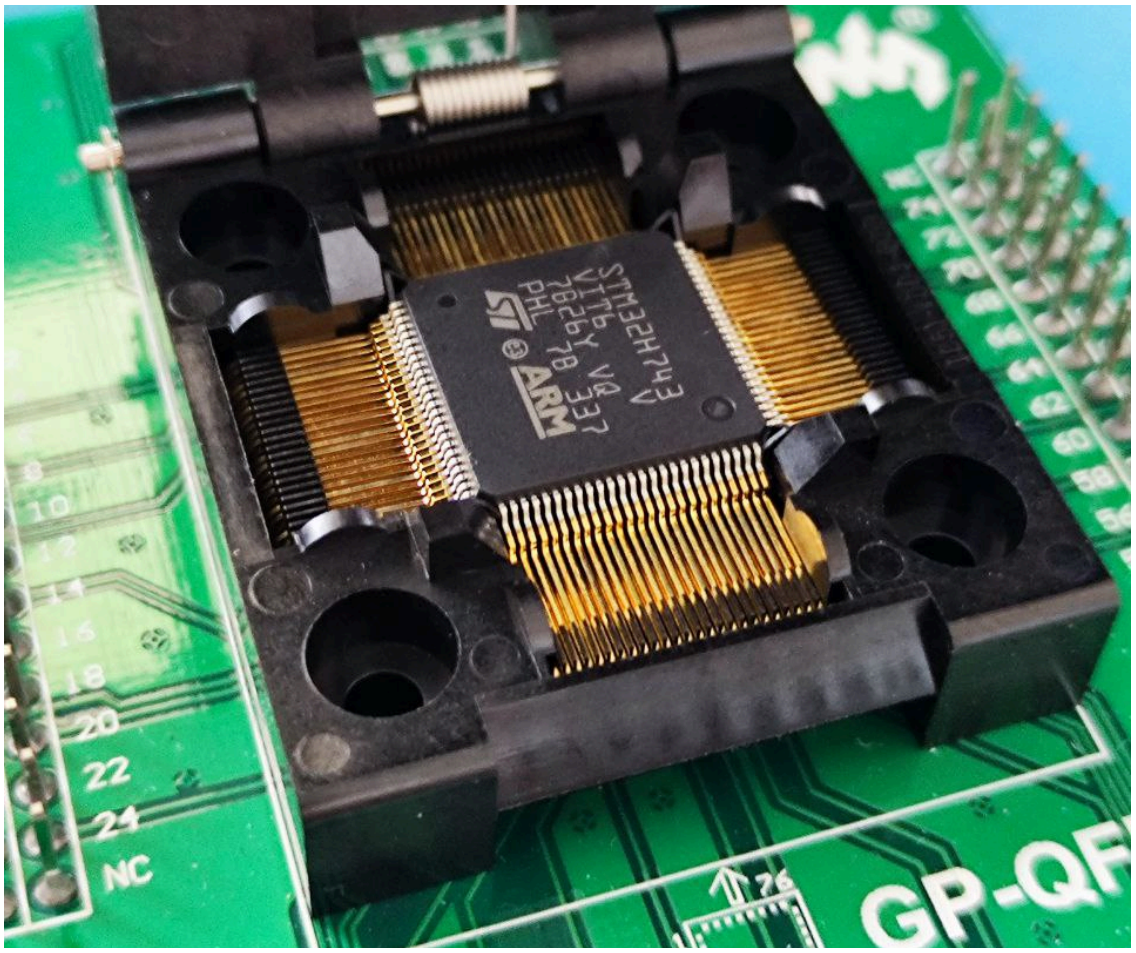
STM32H7 MCU Susceptible to Glitching Attacks (Impact: Low)

Description

Testing revealed that the STM32H7 microcontroller is susceptible to glitching attacks. During testing, an STM32F743VIT6 was used to test whether a security downgrade from RDP2 to RDP1 was possible on the STM32H7. Although no prior work has demonstrated this vulnerability on the STM32H7 specifically, it was in fact confirmed during testing. The use of a secure element mitigates many of these risks, as critical operations rely on the secure element and not solely on the STM32.

```
Open On-Chip Debugger 0.11.0
Licensed under GNU GPL v2
For bug reports, read
  http://openocd.org/doc/doxygen/bugs.html
DEPRECATED! use 'adapter speed' not 'adapter_khz'
Info : J-Link V10 compiled Jan 30 2023 11:28:07
Info : Hardware version: 10.10
Info : VTarget = 3.244 V
Info : clock speed 100 kHz
Warn : There are no enabled taps.  AUTO PROBING MIGHT NOT WORK!!
Info : JTAG tap: auto0.tap tap/device found: 0x6ba00477 (mfg: 0x23b
(ARM Ltd), part: 0xba00, ver: 0x6)
Info : JTAG tap: auto1.tap tap/device found: 0x06450041 (mfg: 0x020
(STMicroelectronics), part: 0x6450, ver: 0x0)
Warn : AUTO auto0.tap - use "jtag newtap auto0 tap -irlen 4
-expected-id 0x6ba00477"
Warn : AUTO auto1.tap - use "jtag newtap auto1 tap -irlen 5
-expected-id 0x06450041"
Warn : gdb services need one or more targets defined
```

OpenOCD output after a successful glitch. The STM32 was configured to RDP2.



The STM32H7VIT6 in a socket for testing.

Impact

Though the STM32H753AIJ6 used in the ERA Wallet was not evaluated, it is likely that the core is susceptible to the same attack as the STM32H743VIT6, as these are both variants of the same family. Moreover the impact of this vulnerability is relatively well known and likely limited to just SRAM readout during execution. However, the use of a secure element as well as disabling the STM32 System Memory Bootloader in firmware will greatly reduce the associated risks and prevent an attacker from being able to leverage this vulnerability in practice.

Recommendation

Using a secure element that verifies the user pin to decrypt sensitive data greatly reduces the impact of this vulnerability. This requires user input at run time and isolates key material by processing cryptographic data on the SE. Several additional mitigation strategies can be implemented in firmware, such as limiting the amount of time that sensitive data is retained in SRAM. There may be additional hardware

features, such as memory protections that can be leveraged to further reduce the impact of such glitching attacks.

SWD Header Accessible (Impact: Low)

Description

Pre-production ERA Wallet devices include accessible SWD connections, which are typically used for debugging and development. These connections, combined with access to STM32 VCAP pins, allow an attacker to perform glitching attacks or downgrade security levels. In production, these connections are unnecessary and increase the risk of in-situ attacks.

SWD header accessible at the edge of the pre-production PCB.

Impact

Accessible SWD connections facilitate potential in-situ attacks, but do not directly lead to exploitation due to other mitigations in place.

Recommendation

Remove SWD connections in production ERA Wallet devices. If debugging capabilities are required, consider using temporary connections or secure debugging protocols.

No Active Tamper Circuit (Impact: Low)

Description

The pre-production ERA Wallet lacks an active tamper detection circuit, which would alert the system if the device is physically opened or tampered with. While epoxy in the production design partially mitigates this issue, the absence of a tamper circuit leaves the device less resilient to physical attacks.

Impact

The absence of an active tamper circuit does not directly lead to exploitation, but reduces the ERA Wallet's ability to detect and respond to physical tampering.

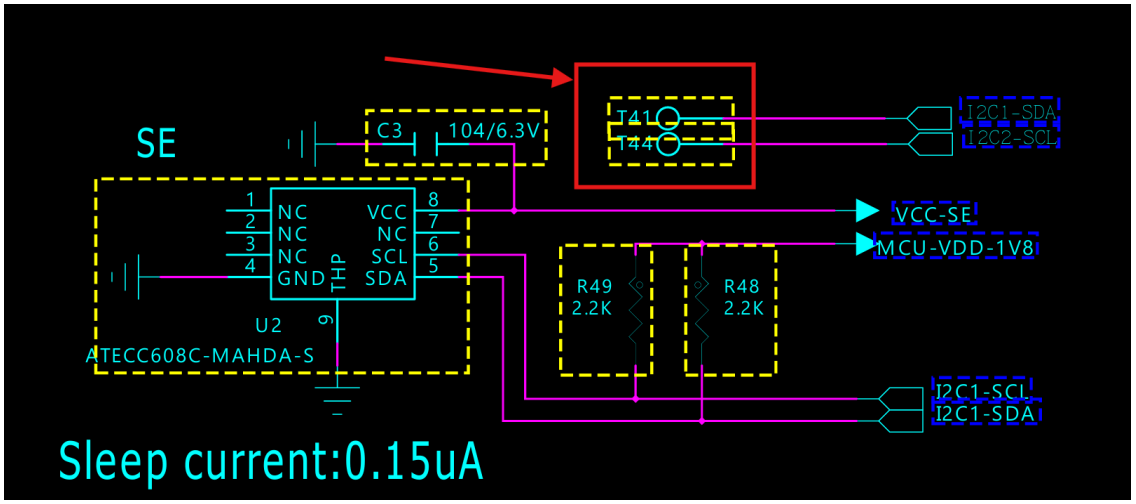
Recommendation

Consider integrating an active tamper circuit in future revisions of the ERA Wallet. This circuit could trigger a self-destruct mechanism or wipe sensitive data upon detecting physical tampering.

ATECC608C Test Points (Impact: Low)

Description

Test points for the ATECC608C secure element were identified in the proposed production schematic of the ERA Wallet. These test points could be exploited by attackers with physical access to replay, corrupt, or alter communications between the secure element and the MCU, for example, using a hardware implant or as part of an evil maid attack.



Test points T41 and T44 connected to the ATECC608C I2C bus.

Impact

Although encrypted communications reduce the risk of direct exploitation, test points increase the potential for supply chain attacks or brief physical compromises.

Recommendation

Remove test points connected to the secure element in the production design of the ERA Wallet. If test points are necessary for manufacturing, ensure they are disabled or inaccessible in the final product.

UART and BOOT0 Pins Accessible (Impact: Informational)

Description

The UART bootloader and BOOT0 pin are likely to be used in factory programming devices, i.e. loading the firmware and bootloader. These connections enable programming and debugging functionality. However, since they are not used after manufacturing, they are redundant and should be removed as they can be leveraged in reverse engineering.

UART programming port and BOOT0 pin.

Impact

Although the UART System Memory Bootloader can be disabled in firmware, an attacker may still be able to utilize these pins to exfiltrate data, for example, by first finding a vulnerability leading to code execution on the device.

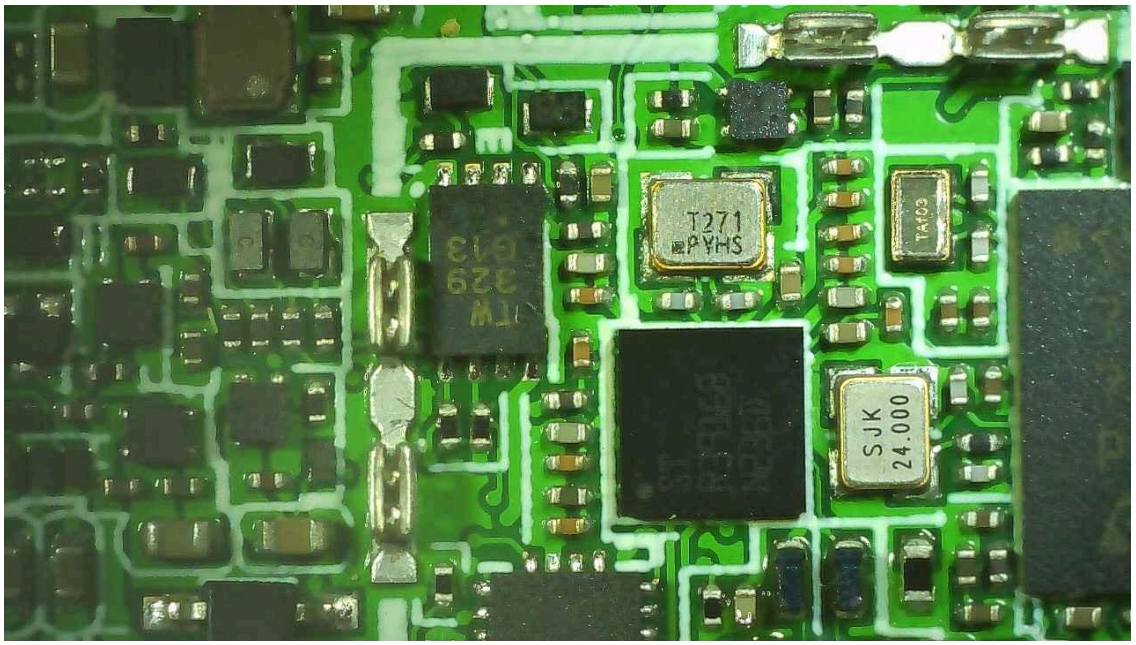
Recommendation

Remove the UART and BOOT0 connections in production ERA Wallet devices after programming is complete. Use PCB design techniques to sever these connections or conceal them completely. Fully disable the UART System Memory Bootloader in firmware to minimize exploitation risk.

RF Shields Not Soldered (Impact: Informational)

Description

Pre-production ERA Wallet devices are designed without permanent RF shielding or epoxy encapsulation, leaving critical components like the STM32H7 MCU and ATECC608C secure element exposed. This lack of physical protection makes it relatively easy for an attacker to probe or extract sensitive data from these components using advanced techniques. RF shields and epoxy are standard practices to add an additional layer of security to prevent physical access and tampering.



Visible PCB clips for the RF shield with the RF shield removed. This particular shield covers the Secure Element and MCU as well as several other components.

Impact

The pre-production devices are relatively easy to open leading to direct access to security relevant signals like the communications between SE and MCU, as well as the MCU VCAP pins.

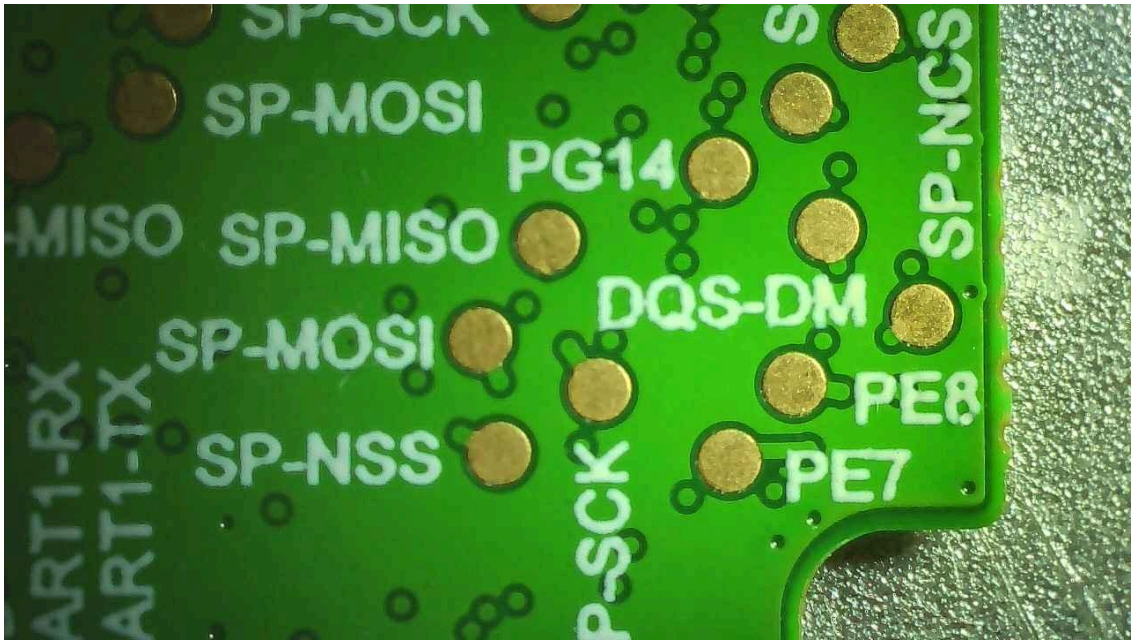
Recommendation

Solder RF shields in place and/or encapsulate the entire ERA Wallet device in epoxy to prevent physical access. Ensure that these measures are applied consistently in the final production process to enhance tamper resistance.

PCB Labels and Test Points (Impact: Informational)

Description

Pre-production ERA Wallet devices numerous testpoints and labels that identify critical signals and connections. While this practice simplifies development and debugging, it also provides attackers with valuable information that facilitates reverse engineering. Labels make it easier to locate security-relevant signals, which could be used in combination with other techniques to compromise the device.



Test points and labels on the pre-production PCB.

Impact

Labeled PCBs reduce the effort required for reverse engineering and increase the risk of exploitation through otherwise inaccessible interfaces.

Recommendation

Remove labels from production PCBs of the ERA Wallet, especially those related to security-critical components or signals. Remove testpoints that are not used after development. If the wallet design is open-sourced, the label information will be publicly available. Hence, this finding is informational in nature.

Exposed pads from unplaced components (Impact: Informational)

Description

Pre-production ERA Wallet devices may include exposed pads for components that are not placed in the final design. Similarly to test points, these pads may expose interfaces that would otherwise be unavailable to the attacker for exploitation.

Impact

These exposed pads do not present an immediate exploitation risk as they will likely not be initialized in the firmware. Nevertheless, they increase the attack surface and could be exploited if not properly secured. Their presence also makes reverse engineering easier.

Recommendation

Ensure that unused pads are removed or rendered inaccessible (for example with epoxy) in the production design of the ERA Wallet. Use PCB panelization or other techniques to sever connections physically. Conduct a thorough review of the PCB layout to minimize unnecessary exposure.

Conclusion

The findings confirm that no critical vulnerabilities were identified in the ERA Wallet. Addressing the identified four informational and four low-impact issues will ensure that the final production device is robust and secure against both physical and logical attacks.

ERA Wallet
Firmware Security Audit

Summary

This audit of the firmware for the ERA Wallet identified three Medium-impact issues, one Low-impact issue, and one Informational finding. The firmware consists of three main components: a bootstrapper that loads low-level code, an mcuboot-signed bootloader responsible for firmware updates, and a FreeRTOS-based firmware image containing the wallet applications. While the firmware is robust, critical attention is required for the bootstrapper vulnerabilities, as it cannot be updated post-manufacturing.

Findings

Vulnerable Format String in Bootstrapper (Impact: Medium)

Description

A format string vulnerability was identified in the bootstrapper's string manipulation code. This vulnerability arises from the use of unsafe string operations such as sprintf and strcat. Specifically, the bootstrapper attempts to construct a string using %s that includes firmware versions and serial numbers. Maliciously crafted input, a malicious firmware update or other low-level vulnerabilities could potentially exploit this vulnerability to read sensitive data or may even lead to arbitrary code execution.

```
/**
 * @brief Draw string with manufacture parameters
 * @return none
 */
static void drawManufacutureString(void)
{
    static char string[HWLT_MANUFACTURE_LEN];

    if (manufacture_string.text == NULL) {
        struct image_version *bl_version =
parser_getImageVersion(FLASH_GET_BASE(BLR_SLOT_0));
        hwlt_version_s       *ver        = hwltVersion_getCurrent();
        sprintf(string, "BL %d.%d.%d BS %s\n", bl_version->iv_major,
bl_version->iv_minor,
                bl_version->iv_revision, (ver->records)[0].ver);
        strcat(string, "SN ");
        strcat(string, (char *)bs_config.serial_number);
        manufacture_string.text = string;
    }
    paint_drawString_en(manufacture_string.x_start,
manufacture_string.y_start,
                        manufacture_string.text, &Font8, BLACK, WHITE);
}
```

era-bootstrapper/src/GUI/gui_screens.c

Impact

Exploitation could result in unauthorized access to sensitive data or arbitrary code execution. The inability to update the bootstrapper results in this issue having a higher impact than in code that can be updated after device manufacturing.

Recommendation

Replace unsafe string operations (sprintf, strcat) with safer alternatives such as snprintf and strncat. Ensure that string lengths are explicitly defined and static. Where possible, minimize or eliminate the use of runtime string operations.

Bootstrapper does not memzero SRAM (Impact: Medium)

Description

Upon boot SRAM memory contents are not zeroed. As a result residual data from previous sessions, such as sensitive PINs, PIN hashes, or screen contents, may not be overwritten and remain in SRAM. This flaw increases the risk of data leakage, as an attacker could exploit known vulnerabilities in the STM32 family to recover sensitive data retained in uninitialized memory regions.

Impact

Sensitive data retained in memory after a reboot could be exposed to attackers, undermining user security.

Recommendation

Modify the platform load script to zero all SRAM memory contents during boot, except for areas explicitly required for communicating between firmware components, such as the mailbox. Reserved areas can also be specifically flagged to determine whether or not they contain data and zeroed if they are unused.

SE TRNG not used for Entropy Generation (Impact: Medium)

Description

The firmware does not utilize the ATECC608C TRNG for entropy generation. Though the firmware does utilize several additional sources, for example pseudo-random inputs from the user, the only TRNG utilized in the process is the STM32H7 TRNG. This TRNG is not a certified high-security RNG and may generate insufficient entropy. Dependence on a single uncertified TRNG source increases the risk that weaknesses in the RNG could lead to low-quality cryptographic seeds or weak seed phrases, which are more susceptible to brute force attacks.

```
/**
 * @brief   Generates 32 bit random value by hardware.
 * @param   with_lock Lock random generator usage.
 * @return  random value
 */
static uint32_t rnd_generate(bool with_lock)
{
    char *name;
    uint32_t rnd = 0xFFFFFFFF;

    bool is_locked = false;
    if (with_lock) {
        is_locked = rnd_lock();
    }

    if (HAL_RNG_GenerateRandomNumber(&hrng, &rnd)) {
        name = pcTaskGetName(NULL);
        elog_e(ELOG_ID_RND, "RND- ERROR, tsk - %s", name);
    }

    if (is_locked) {
        rnd_unlock();
    }

    return rnd;
}
```

era-framework/boards/h7common/src/board_rnd.c

Impact

Potentially weak entropy directly compromises the security of cryptographic keys and seed phrases, which may lead to brute-forcing attacks on the seed that can result in the theft of user funds.

Recommendation

Several entropy sources are already used to strengthen randomness. However, the only TRNG used in the entropy generation process is the STM32H7 TRNG. ERA Wallet should utilize the ATECC608C's RNG alongside the STM32H7 TRNG, in combination with other existing pseudo-random sources from user input. Additionally, ERA Wallet should incorporate static device-specific identifiers to provide additional unique inputs, such as STM32 UID and ATECC608C unique ID. Additional static random data can be injected during manufacturing and source from a secure server or a trusted entropy source. This static entropy can be stored in the STM32 OTP area and supplement other dynamic sources of entropy and device unique identifiers. Though the pseudo-random generation implemented on the wallet may provide sufficient entropy in many cases, supplementing this with verifiably good entropy from verified TRNGs will result in verifiably better entropy.

Vulnerable Format String in Bootstrapper (Impact: Low)

Description

A format string vulnerability was identified in the GUI library used alongside the mcuboot bootloader. This vulnerability arises from the use of unsafe string operations such as sprintf and strcat. Specifically, the bootstrapper attempts to construct a string using %s that includes firmware versions and serial numbers. Maliciously crafted input, a malicious firmware update or other low-level vulnerabilities could potentially exploit this vulnerability to read sensitive data or may even lead to arbitrary code execution.

```
static void readVersions(void)
{
    static char ver_string[HWLT_VERSION_MAX_VER_LEN * 2 + 10] = {0};
    hwlt_version_s *ver = hwltVersion_getCurrent();
    if (ver->cnt > 0) {
        sprintf(ver_string, "BL %s", (ver->records)[0].ver);
    }

    strcat(ver_string, " BS ");
    char *bs_ver = (char *)ver_string + strlen(ver_string);
    if
(!mailEngine_getManufactureString(RAM_MBOX_FW_MANUFACTURE_VERSION_BS
, bs_ver,
                                      HWLT_VERSION_MAX_VER_LEN))
{
        return;
    }
    widgetLabel_versions.text = ver_string;
}
```

era-mcuboot/src/gui/gui_core.c

Impact

Exploitation could result in unauthorized access to sensitive data or arbitrary code execution. Because the mcuboot bootloader can be update through a firmware update after manufacturing, this issue is less critical than similar issues in the bootstrapper that cannot be updated

Recommendation

It's unclear whether this code is directly exploitable and the bootloader is updatable. It is recommended to eliminate any unsafe string operations and use safer operations, i.e., strncat and snprintf. The length of the strings should be well defined and the string length is not dynamic. Consider eliminating such string operations completely and using fewer calls to string operations.

Vulnerable Format Strings in Bootstrapper, Bootloader and Firmware (Impact: Informational)

Description

Numerous insecure format string operations exist across the bootstrapper, bootloader, and firmware. These operations use external data, such as reading the battery level, to construct strings for display or logging. While not necessarily directly exploitable, these operations could reveal sensitive data if an attacker manipulates external inputs, such as through man-in-the-middle attacks at the physical communication layer with physical access to the wallet.

```
void screen_warning(screen_warning_e warning_num, uint8_t batt_lvl)
{
    paint_newImage(screen_framebuffer, EPD_WIDTH, EPD_HEIGHT,
DISPLAY_ROTATION, WHITE);
    paint_clear(BLACK);

    ...

    char batt_lvl_string_text[7];
    sprintf(batt_lvl_string_text, "%d%%", batt_lvl);
    batt_lvl_string.text = batt_lvl_string_text;
    drawString(&batt_lvl_string, Font16);

    epaper_switchScreen((const uint8_t *)screen_framebuffer, false);
    epaper_updateScreen(false); /*only here partial when - true*/
    printf("Low battery screen drawn.\n");
}
```

era-bootstrapper-recursive/src/GUI/gui_screens.c

Impact

Non-critical format string vulnerabilities could still expose memory contents and potentially reveal sensitive data stored on the device.

Recommendation

Carefully review all string operations for potential vulnerabilities. Reduce reliance on runtime string manipulation wherever possible. Many string operations in the firmware are only used from debugging and can be eliminated completely from the production binaries through the use of compile-time macros (i.e., #ifdef). A library containing secure sanitized string operation wrappers can also be used to ensure that new vulnerabilities are not added to the firmware.

Conclusion

The firmware audit highlights three Medium, one Low, and one Informational findings in the ERA WLT firmware. While Medium-impact issues require immediate attention, the inability to update the bootstrapper makes it imperative to address the bootstrapper vulnerabilities before production. The identified RNG issue threatens the integrity of cryptographic operations and user data security, hence these should be fixed before the first production firmware is released to users. Other findings can be resolved through firmware updates post-release.

ERA WALLET Retest

No Active Tamper Circuit - Fixed

Epoxy added in production PCB.

ATECC608C Test Points - Fixed

Fixed in production PCB.

Vulnerable Format String in Bootstrapper - Fixed

Fixed in PR #94

Bootstrapper does not memzero SRAM - Fixed

Fixed in PR #93

SE TRNG not used for Entropy Generation - Fixed

Will be fixed in software update at launch

Vulnerable Format String in Bootstrapper - Fixed

Partially fixed additional fixes in future software updates

Vulnerable Format Strings in Bootstrapper, Bootloader and Firmware

Informational, doesn’t directly lead to exploitation. Will be fixed in the future

ERA Wallet Security Audit Conclusion

Based on this comprehensive security audit of the ERA Wallet, the overall architecture demonstrates excellent security design principles and robust implementation. The wallet effectively addresses key threats including malware protection, evil maid attacks, and supply chain vulnerabilities through its thoughtful hardware choices and security features. The use of a STM32H753 processor and making use of all the security features offered by it alongside a secure element with monotonic counter capabilities (ATECC608C) provides a strong foundation for secure operations. The architectural decisions, such as QR code scanning for air-gapped transactions and encrypted communications, show careful consideration of real-world security challenges.

Most notably, all significant security findings from the audit have already been addressed or are being mitigated. The medium-impact issues identified in the firmware, including vulnerable format strings in the bootstrapper and single entropy generation source, have been fixed prior to this report's release, with remaining improvements scheduled for the launch software update. The low-impact hardware findings, such as the exposed SWD headers and test points, are being addressed in the production design through proper RF shielding and epoxy encapsulation. This demonstrates the team's commitment to security and their ability to rapidly address potential vulnerabilities, resulting in a highly secure hardware wallet architecture that meets industry standards for protecting users' digital assets.