FOUNDATION

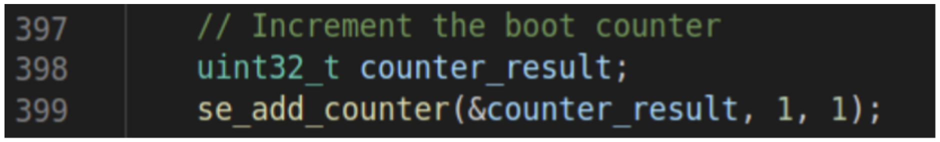
Public Response to Keylabs Audit of Passport – Q2 2021

Keylabs Audit – Unmodified Report	***Foundation Devices' Response***
Executive Summary	
Keylabs was tasked with performing a security audit of a pre-production version of the Foundation Devices Passport wallet. The main focus of the review was ensuring the security of the boot, firmware verification and login process, ensuring that an unauthenticated attacker can not access stored funds. The type of processor used in the device is known to be vulnerable to attacks that can allow i.e. a supply-chain attacker to modify the firmware. The threat-model of the device should be adjusted & communicated accordingly. During the review multiple vulnerabilities in the firmware handling code were found, incl. unauthenticated downgrade attacks. In addition, it was found that certain parts of the code are relatively susceptible to fault-injection attacks, and guidance on improving the defenses were provided. Foundation Devices quickly provided fixes for the identified issues, which were sanity-checked by Keylabs.	Foundation selected Keylabs to perform an audit of Passport's firmware and bootloader due to their extensive previous experience with other hardware wallets. You can read more about their work and qualifications at https://keylabs.io and https://wallet.fail. See the Foundation Devices security page for information on Passport's security model.

Audit limitations	
This audit was conducted on pre-production soft- and hardware, and certain functionality was not yet implemented or could not be tested on the current state of the software. (For example firmware upgrades were not possible on the test devices.) This document extends on the previous report on the configuration of the secure element and does not re-iterate the findings of it.	Although Keylabs was not able to fully exercise the firmware update feature, they did perform an in-depth review of the associated code. In addition, Foundation spent extensive additional time reviewing and testing firmware updates to ensure that they are reliable and safe. See section 9 of SECURITY.md for the current Secure Element configuration.
1.1. Boot-counter can be defeated	
On boot, the firmware increases a monotonic counter on the ATECC. The result of this counter increase is not checked, and as such an on-bus attacker can prevent the counter from being increased. This can allow an attacker to hide boot attempts from the user of the wallet.	The boot counter result is now checked and Passport will not boot if an attacker prevents it from incrementing. Note that the boot counter was added for passive information only. Users can optionally view the boot counter each time they use Passport and can compare its value against the previously noted value. This can help detect evil maid login attempts. It is not intended as a protection against physical attacks where the attacker disassembles Passport and connects it to sophisticated electronic equipment.

1.2. Firmware upgrades can be triggered without PIN approval

The normal UX flow only permits upgrading the firmware after entering the PIN. The reviewed firmware version however checks the external SPI flash to determine whether a firmware upgrade is available. As this SPI flash could be modified by an attacker, it can allow unauthorized firmware upgrades.



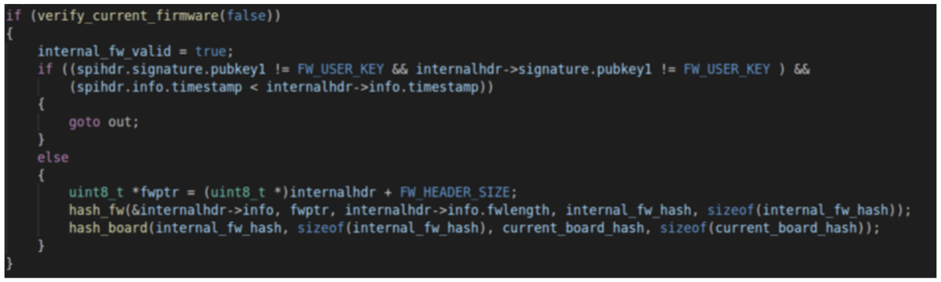
```
    // Increment the boot counter
    uint32_t counter_result;
    se_add_counter(&counter_result, 1, 1);
```

This issue would have allowed an attacker to install any Foundation-signed firmware, but Passport would still prevent any unsigned firmware from being installed.
To fix this issue, the external SPI flash must contain a hash of the firmware along with a hash that includes the MCU unique ID, the Secure Element ID, and the one-time pad (unique value per Passport). An external attacker would not be able to generate this hash.

Upon reboot, the bootloader recomputes this hash and if the SPI flash does not contain a matching hash, the SPI flash is cleared and the installation is canceled. Passport boots the old firmware without modification.

1.3. Downgrade protection bypass 1: Logic bug

The reviewed firmware contains a logic bug that disables the downgrade protection:



```
if (verify_current_firmware(false))
{
    internal_fw_valid = true;
    if ((spihdr.signature.pubkey1 != FW_USER_KEY && internalhdr->signature.pubkey1 != FW_USER_KEY ) &&
        (spihdr.info.timestamp < internalhdr->info.timestamp))
    {
        goto out;
    }
    else
    {
        uint8_t *fwptr = (uint8_t *)internalhdr + FW_HEADER_SIZE;
        hash_fw(&internalhdr->info, fwptr, internalhdr->info.fwlength, internal_fw_hash, sizeof(internal_fw_hash));
        hash_board(internal_fw_hash, sizeof(internal_fw_hash), current_board_hash, sizeof(current_board_hash));
    }
}
```

Most of the verify_ functions return 1 on success, however verify_current_firmware returns 0 on success, and as such the downgrade-check will not be executed if the current firmware is valid.

This was a bug inadvertently introduced just before the bootloader and firmware were sent out for the audit.

The code correctly checks the return code now.

1.4. Downgrade protection bypass 2: TOCTTOU

The firmware upgrade procedure is split into two parts: ● Check signature of firmware on external SPI flash ● Upgrade internal flash with contents of SPI flash For both steps the firmware is newly read from the external SPI flash. An active attacker can switch-out the SPI flash contents between both steps and flash a different firmware than the one that was checked.	When performing the signature and version check, Passport now computes a hash of the SPI flash header contents. It later recomputes this hash when copying the SPI flash contents into internal flash. If the hash does not match, then the firmware update does not proceed.

1.5. Downgrade protection bypass 3: Failed upgrade attack

An attacker can trigger the device into a failed-upgrade state, for example by removing power during a firmware upgrade. Once the internal firmware is in an invalid state, the downgrade-check will not be performed, allowing an attacker to downgrade the firmware of the device.	This was a misunderstanding by Keylabs. It's true that the downgrade check was skipped, however, there is a firmware hash stored in the Secure Element, and this would not be properly updated in the attack scenario, so even though the old firmware may have been installed, it would not boot since the firmware hash would not match. In addition, the fix Foundation implemented for finding 1.2 now makes it impossible for an attacker to inject a firmware update in the first place.

1.6. Fault injection resistance: Delays

To make fault-injection attacks on the firmware harder to perform the code contains randomized delays to make glitching specific instructions more difficult. However in some parts of the code the delay occurs immediately before controlling externally probable IOs, which provide a decent post-delay trigger. It is recommended to place the delay immediately in front of the security-relevant function calls/checks.	Additional random delays have been added to the bootloader startup sequence.

1.7. Fault injection resistance: Boolean & Switch Logic

The reviewed firmware contains multiple boolean-checks and switch-statements that are used for security-relevant checks. Branches on boolean logic and switch-statements without default branches are particularly vulnerable to fault-injection, as a single bit-flip can potentially invert the condition. It is recommended to compare against explicit bit-patterns.	Passport is now comparing against SEC_TRUE and SEC_FALSE values in several locations identified by Keylabs instead of using boolean true and false.

1.8. Potential EM side channels

The reviewed, pre-production device has significant coil-whine and other emissions when performing certain actions. It should be ensured that this noise can not be used as a side-channel.	This was an issue with the onboard power circuitry of the pre-production Passport board used for the review. Foundation had independently found the same issue, and it was addressed for Passport's production boards.

1.9. Attacks on ATECC608A

After the review, new attacks on the ATECC608A and its usage in the Coldcard firmware were published. Foundation Devices confirmed that these issues were fixed in the latest Passport firmware.	This defect was never in the Passport source code.

1.10. External flash AES-CTR encryption uses weak counter value

The external flash contents are AES-CTR encrypted. While the key seems to be secret enough, the counter-value is potentially weak and potentially re-used to encrypt different plaintexts. This can lead to situations where an attacker with an old known-plaintext and ciphertext is able to decrypt newer data, or where an attacker can potentially correctly encrypt new data in a way that they will be accepted by the firmware.	The external SPI flash is used to store hashed UTXO data to prevent a specific social-engineering attack that involves a compromised software wallet. This data is not particularly sensitive, as all the same information is publicly available on the blockchain too. Foundation will investigate the addition of a random counter for each stored entry.

1.11. External flash contents not signed

The external flash-contents are checksummed but not signed. It is recommended to sign the external flash contents.	This is related to the same UTXO data described in finding 1.10. This data is not sensitive and does not need to be signed.

1.12. Informational: Processor does not provide privilege separation

The Passport is based on the STM32H series processor, which is missing some of the advanced privilege separation features found on other STM32 processors (such as the “firewall” on the STM32L4). This feature allows creation of trusted areas that are separate from the main firmware). This can be used, for example, to separate the main firmware from accessing secrets such as the pairing secret directly. This means that a potential information leak vulnerability (such as an out-of-bounds read) can leak the pairing key. However the reduced attack surface provided by the QR-code based interface lowers the risk of such attacks.	In finding 1.13 we see that it is possible to glitch MCUs out of “read-out protection” mode given enough physical access, time, and money. This is true even of MCUs with the “firewall” feature, and as such, the “firewall” provides little additional security. Passport is air-gapped and has tightly controlled data paths via QR codes and microSD cards. Even if a defect in one part of the code exposed the pairing key, there is no way for that to “leak” out through the limited communication mechanisms that Passport provides.

1.13. Informational: Processor potentially vulnerable to fault-injection attacks

The processor used on the Foundation Passport is part of the STM32 family. Research shows that on a lot of STM32 variants it is possible to by-pass the read-out protection using attack-methods such as fault-injection. Given this the supply-chain security of the device is difficult to ensure, as the firmware might pass the authenticity check even if the device was for example backdoored by a supply-chain attacker.	This type of “fault-injection” vulnerability requires generating electrical spikes at specific times and frequencies into the pins of the MCU. Nearly all MCUs are susceptible to this type of attack. It does, however, require direct physical access to the device for extended periods of time, and requires damaging or destroying the device case to access the internal circuitry. In other words, you would notice if someone did this and returned the device to its storage place.

	The cost for the equipment for these “fault-injection” attacks varies up to about $2,000 depending on the MCU being attacked. Once the read-out protection is breached, however, the attacker has full access to the MCU, even if it has the “firewall” feature mentioned in finding 1.13. Even in this scenario though, an attacker will still not have access to your private seed, as the PIN is also required.
1.14. Informational: No active tamper detection measures	
The device does not have active tamper detection measures, at-rest physical attacks do not trigger a tamper response such as key-deletion	Passport’s Secure Element combined with the encryption used for the private seed provides a high level of protection against attacks, reducing the need for tamper detection. Still, Foundation will investigate the addition of this type of tamper detection in future products.