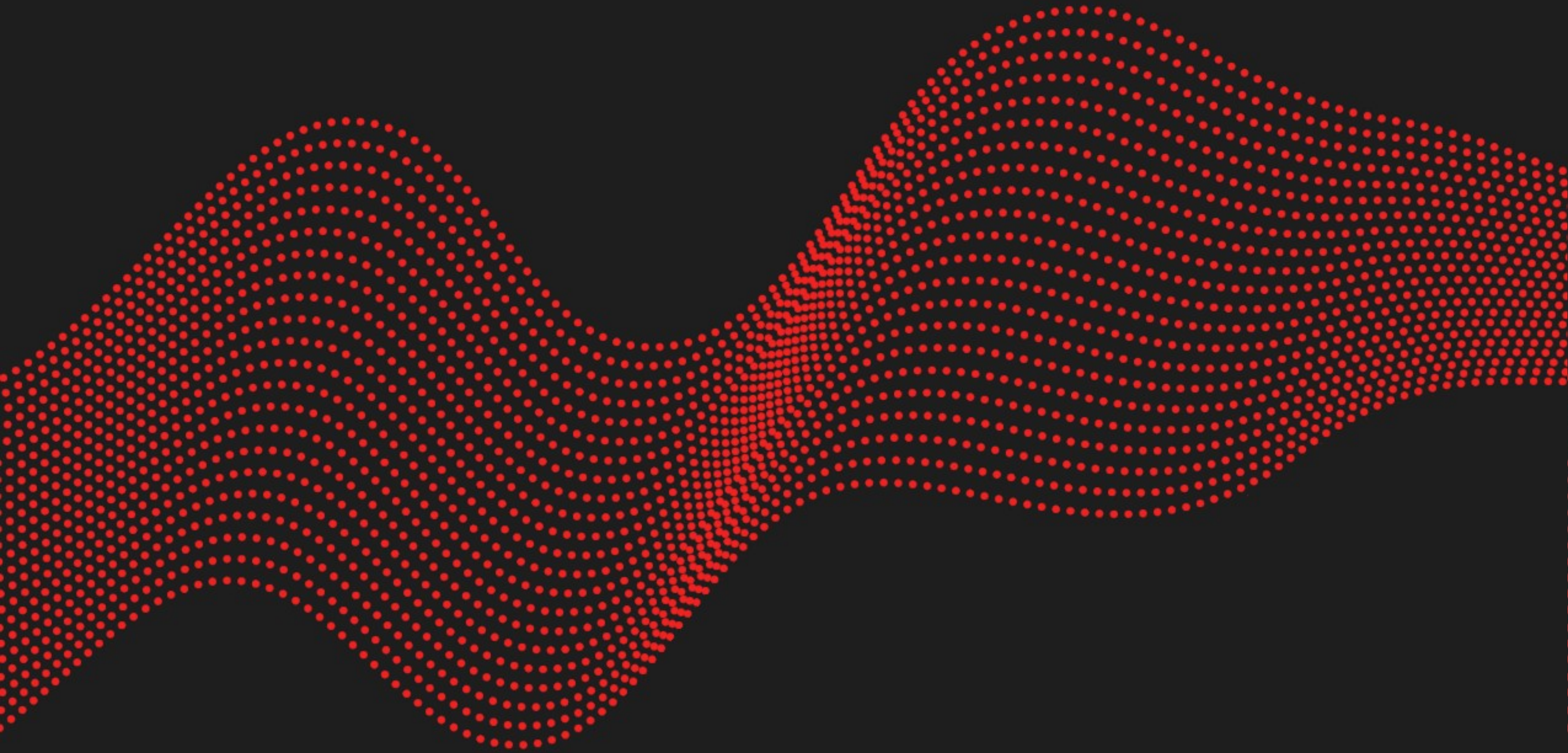
SYNACKTIV

LEDGER – SECURITY ASSESSMENT REPORT

2025/07//17



VERSION 1.0 – CONFIDENTIAL

Contents

Introduction

Context and objectives

Ledger has asked Synacktiv to perform a security assessment on the firmware running on Ledger's Security Element (SE) as part of their pre-release security review campaign. This firmware is fully developed by Ledger.

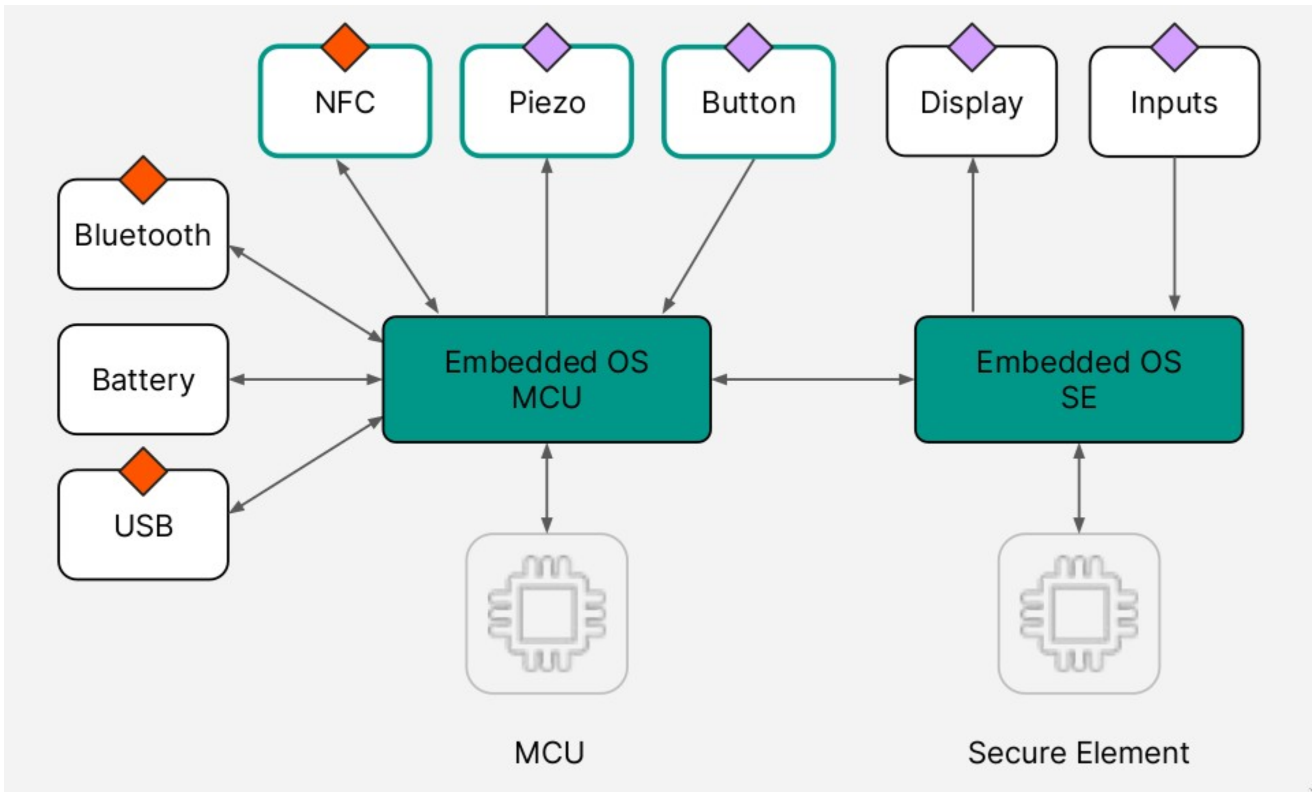


Illustration 1: Stax hardware architecture (schema provided by Ledger)

To facilitate a thorough examination, Ledger provided Synacktiv consultants with access to both the firmware's source code and build environment. This transparent, white-box methodology enables in-depth analysis of the firmware's architecture and implementation, helping to identify potential malicious elements.

The primary objective of this audit was to evaluate the firmware's security posture, specifically focusing on the reimplementation of the I/O stack:

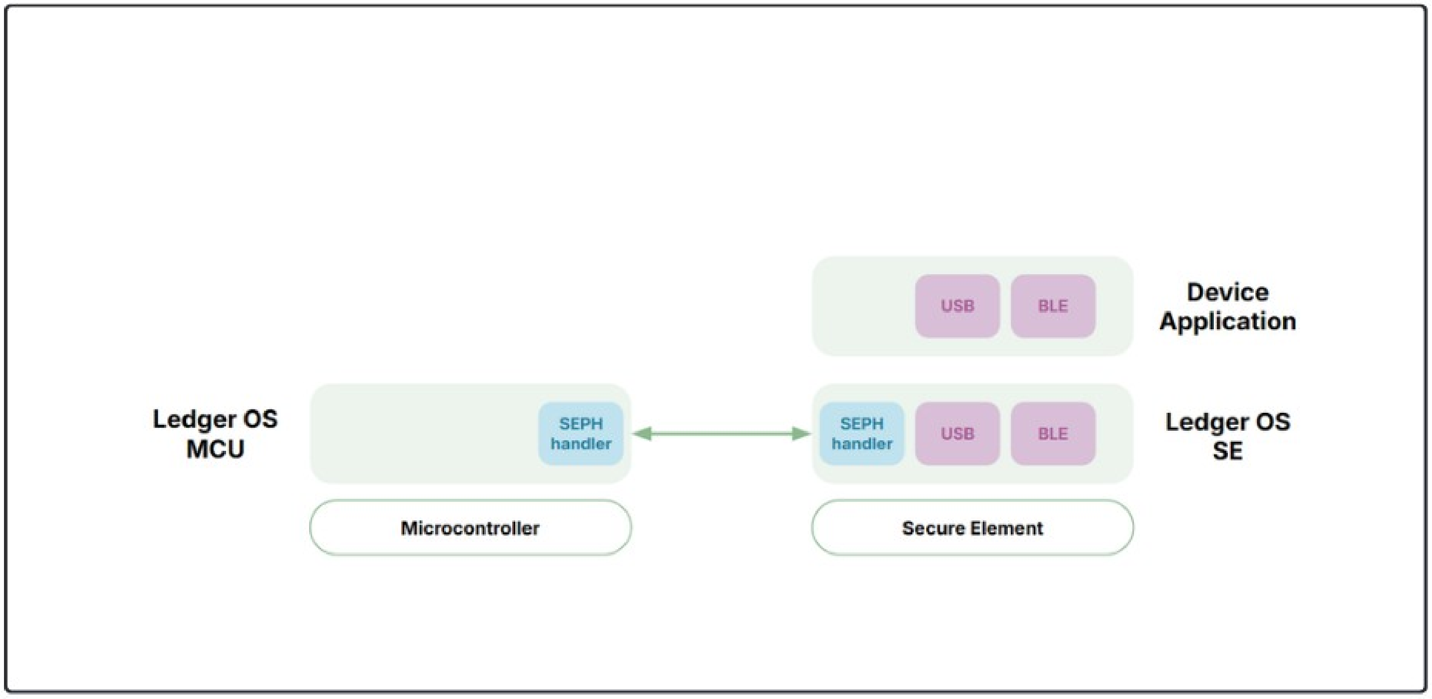


Illustration 2: Legacy I/O stack

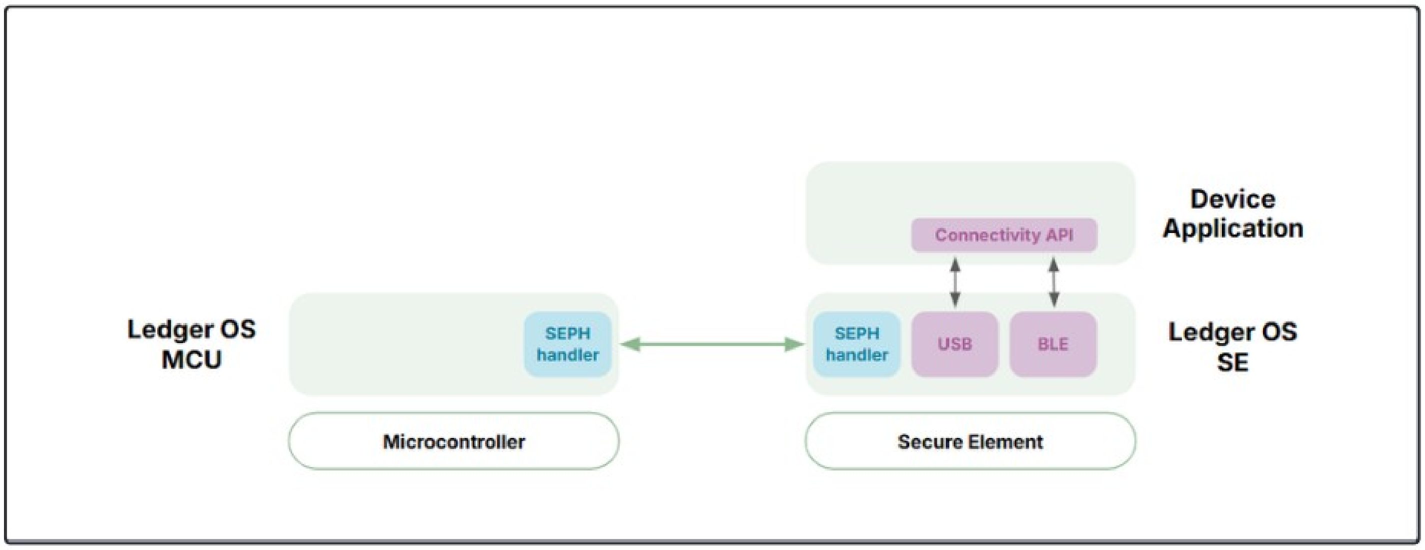


Illustration 3: Revamp I/O stack

The presence of features or vulnerabilities introduced by this reimplementaion would pose significant risks to the security and privacy of Ledger customers, potentially allowing unauthorized access or manipulation of sensitive information.

Scope and limits

The source code has been delivered within an archive containing all source code running on the SE chip, accompanied by a `Docker`-based build environment.

Source	File	SHA2-256
Nano S+ v1.4.0-rc1 Nano X v2.5.0-rc1 Flex v1.4.0-rc1 Stax v1.8.0-rc1	`Ledger_OS_Release_19_June_2025.tar.bz2`	ae1e9c16c533c9bc1f35d6e6b52177bd5865db33b20ac5e7a0fa21958ca1447a

Once built, this asset runs on the SE component of Ledger's products.

Product	File	SHA2-256
Nano S+ v1.4.0	`token.hex`	f9155b311f29dcb7c5d2bfdec288cf423974dab23663a5197d4cab116422b3cd
Nano X v2.5.0	`token.hex`	3673b14da98048f56730d99f951ecbb190d24649aa7b6fe6311f5a5b445e2175
Flex v1.4.0	`token.hex`	e5d7d6fa10060788450c373c841097ddc247224a2db426f5a254453c3cf19aaa
Stax v1.8.0	`token.hex`	ff192b961d66dc37feb9125bfc5a3829a47ffef355cc7b8d69ae248fd9d31b1e

Timeline

Date	Description
2025/06/19	Receipt of the source code
2025/06/19	Audit start
2025/07/04	End of audit

Assessment methodology

During a 24 man-day study, which involved two experts for 12 working days, Synacktiv experts reviewed archives provided by Ledger. These archives contained the full source code for the Secure Element for Nano S+ `v1.4.0`, Nano X `v2.5.0`, Flex `v1.4.0` and Stax `v1.8.0` devices, along with a Dockerfile to facilitate the projects builds. No untrusted binary blobs were observed during the review.

The external dependencies are consistent with those referenced in the previous report. The compilers used include `gcc_arm-7` and `gcc_arm-10`, fetched from the official ARM developer website (https://developer.arm.com/), and `clang-12.0.0` fetched from the official LLVM releases (https://github.com/llvm/llvm-project/releases). The rust toolchain `nightly-2024-03-19` is fetched via `rustup`, which comes from the official website (https://sh.rustup.rs). All the other external dependencies are fetched either using `apt` on `ubuntu:22.04` (the base image of the docker container) from the official repositories or via pip, which is itself installed from the official Ubuntu repositories.

It was verified that the MD5 hash of the final built code matched the hash provided by Ledger.

During the review, the source code was manually inspected for dangerous features. This included features aimed at exporting user secrets (such as seeds, private keys, or PINs), bypassing user consent (particularly for critical actions like upgrades or enabling recovery services) and exporting Ledger's secrets.

Additionally, all the methods used to interact with the Operating System—whether through syscalls from untrusted applications or via I/O channels such as APDU/NFC/BLE/USB—were reviewed to ensure no dangerous features were present or accessible through non-privileged code.

It is important to note that the internal implementations of the cryptographic algorithms within the code were not reviewed.

The entire study was conducted statically, as emulating the operating system was not feasible within the given time constraints. Synacktiv experts had access to Ledger devices to perform dynamic tests but didn't use them in the allotted time. The auditors analyzed the source code as well as the firmware resulting from the compilation.

Results

The source code of the Operating System, mainly written in C, appears to be well designed to ensure the security of the embedded system.

No dangerous features were found during the assessment. Moreover, the redesign of the communication stack and its centralization in the Ledger OS context did not introduce critical vulnerabilities. There are apparently no functions to export secrets from the device without user consent in this release of the firmware:

- No feature intended for exporting user secrets were identified:
  - Exporting the PIN appears to be impossible
  - The only identified method to export shards of the seed is through the **`recovery`** feature, which requires explicit user consent. No alternative methods for exporting the seed were found, meaning that if the **`recovery`** service is not used, the seed will remain securely within the Secure Element.
  - The firmware itself does not appear to export private keys under any circumstances. However, the security of an application's private key is closely tied to its derivation path. As outlined in Ledger's documentation, the cryptographic principles ensure that all private keys can be reconstructed if both the master seed and the associated derivation path are known. This means that if two applications share the same derivation path, they will also share the same private keys. Ledger OS ensures that an application can compute the private key only for the derivation path specified in its metadata. Thus, it is Ledger's responsibility to ensure that no pair of applications use the same derivation path. One way to ensure an untrusted application cannot leak private keys is to verify its derivation path within Ledger Live, or by examining the Makefile on Ledger's GitHub. Additionally, ensure that the application's signature matches the correct one on your device. You can check that by launching the app from Ledger Live, which requires user consent, and allow you to display more infos about the application. The important value here is called "Identifier".
- There does not appear to be any method to bypass user consent:
  - The consent routine appears to be well-designed, and no bypass mechanisms were identified.
  - Critical features require user consent, and no alternative methods for triggering these features were discovered:
    - Upgrading the firmware
    - Exporting/importing app's data

- Enabling the `recover` service (The Ledger Security Key was separately audited, and no explicit backdoor was found on its fermware, version 1.3.0-rc2.

Conclusions

In conclusion, Ledger's operating system demonstrates a robust and secure design, without any dangerous features identified during the review. Its architecture effectively addresses key security concerns and provides a solid foundation for safeguarding user data and device functionality.

However, it is important to note that the applications running on the device represent a critical point in the overall security model. These applications were not within the scope of this review and, as such, were not assessed for potential vulnerabilities or malicious behavior.

That said, the open-source nature of these applications provides an opportunity for users who are particularly security-conscious to conduct their own assessments. A suspicious user has the option to review the source code, build the applications independently, and verify the integrity of installed applications by comparing the hash of the compiled code with the hash of the application installed on the device, by checking the "Identifier" from a consent screen for launching the app. This process allows users to ensure that no unauthorized modifications have been made to the applications.

This transparency and the availability of source code contribute positively to the product's security posture, empowering users to take additional steps to verify and ensure the integrity of their applications if desired.

In this regard, it is worth noting that previous versions of the Nano S Plus firmware, as well as the Nano X and Nano S, were CSPN certified. CSPN, which stands for **`Certification de Sécurité Premier Niveau`** (First Level Security Certification), is awarded by ANSSI, the French National Agency for the Security of Information Systems. The CSPN certification signifies the robustness of these products against an **`enhanced basic attacker`** as defined by the Common Criteria standard. This certification process includes a conformity analysis and penetration tests conducted by an independent third-party evaluator. These certifications provide an initial level of confidence in the security of the Secure Element, further reinforcing the trustworthiness of the product's design.

SYNACKTIV

+33 1 45 79 74 75

contact@synacktiv.com

5 boulevard Montmartre

75002 – PARIS

www.synacktiv.com