Security Advisory: Flash Snapshot Restore Enables PIN and Passkey Rollback on Pico-FIDO (RP2350)

Token2 Sarl, Research and Development

December 9, 2025

Abstract

This security advisory presents a reproducible proof-of-concept demonstrating that, on a Pico-FIDO device based on the RP2350 microcontroller, a full flash snapshot taken via the BOOTSEL/picotool interface can be restored to the *same device*, thereby reverting PIN state, resident credentials, and wrong-PIN counters. This enables an adversary with brief physical access to (a) remove newly created passkeys, (b) unblock a PIN-blocked device, and (c) restore an earlier device state. Our experiments show that enabling Secure Lock or Secure Boot—which protect confidentiality and boot authenticity—does **not** prevent such rollback attacks. A mitigation roadmap emphasizing anti-rollback mechanisms (monotonic counters, signed and versioned updates, OTP-backed counters, and PIN complexity enforcement) is also presented. PIN complexity enforcement is implemented in Token2's fork of pico-fido [5].

1 Introduction

FIDO2 hardware authenticators provide strong phishing-resistant authentication and are increasingly implemented in open-hardware projects such as pico-fido. Embedded microcontrollers like the RP2350 offer features such as Secure Boot and one-time programmable (OTP) memory. However, these features primarily address confidentiality (via encrypted storage) and code authenticity (via signed boot) and do not automatically provide *anti-rollback* guarantees. This advisory documents a practical PoC showing that a full flash dump of an RP2350-based pico-fido device, taken via BOOTSEL, can be restored to the same device to roll back PINs, resident credentials, and counters.

2 Background

The RP2350 microcontroller includes OTP memory and Secure Boot features enabling Secure Lock (flash encryption tied to an OTP master key, MKEK) and signed bootloaders [2]. Secure Lock encrypts credential blobs so they cannot be decrypted on another device. Secure Boot allows only authenticated firmware to execute. However, neither prevents overwriting flash on the same device. The pico-fido project illustrates this gap in practice.

3 Threat Model

The adversary has temporary physical access to the device and can use BOOTSEL and `picotool`. OTP secrets are assumed non-extractable. The adversary seeks to unblock a locked device or remove new passkeys by restoring an earlier flash snapshot.

4 Methodology

1. Flash pico-fido UF2 image; set PIN=1234; create three resident credentials.
2. Enter BOOTSEL and save full flash: `picotool save --all all_flash_dump.bin`.
3. Exit BOOTSEL, create two more credentials (total 5), change PIN to 4321, and trigger PIN block.
4. Re-enter BOOTSEL and restore earlier dump with: `picotool load`.
5. Reboot and inspect state.

5 Results

The device reverted to the earlier state: PIN=1234, three credentials, and unblocked. Wrong-PIN counters reset, allowing new attempts. Secure Lock / Secure Boot did not prevent this rollback. Disclosure revealed PIN hashing used a weak construction; commit 6c85421 introduced OTP-seeded hashing on RP2350/ESP32, mitigating PIN extraction from dumps but not rollback.

6 Analysis

Confidentiality protections (Secure Lock) prevent exfiltration but not rollback. Freshness requires monotonic OTP counters and version checks. Without them, flash snapshots restore all state, defeating anti-brute-force mechanisms.

The standard practice on regular, secure element-based FIDO2 keys is that there is only a maximum of 8 attempts of wrong PIN to be entered, after which the key gets reset. This removes the risk of brute force attacks completely. However, with this reported vulnerability, we remove the limitation of 8 attempts, allowing an attacker to repeatedly restore flash and try unlimited PIN guesses.

The standard FIDO2 protocol enforces additional rate-limiting mechanisms: after **three** incorrect PIN attempts, the key must be physically reinserted (≈1 second delay if automated), and after **eight** incorrect attempts, the device requires a firmware reupload (≈20 seconds delay if automated). Token2's implementation further strengthens security by enforcing **PIN complexity requirements**—mandating a minimum **6-digit PIN**—on top of the standard FIDO2 behavior [5]. This increases the brute-force search space from **10,000** possible combinations (for a 4-digit PIN) to **1,000,000** combinations (for a 6-digit PIN).

6.1 Brute-Force Time Estimates: 4, 6, and 8 Digits

Assuming automated testing:

- Single PIN attempt: 0.1 s
- Reinsertion penalty after 3 failures: 1 s
- Flash restore penalty after 8 failures: 20 s

4-digit PIN (10,000 combinations)

- Reinsertion cycles: $\approx$ 3,333
- Flash-restore cycles: $\approx$ 1,250
- Total time: $\approx$ 29,333 s (about 8.15 hours)

6-digit PIN (1,000,000 combinations)

- Reinsertion cycles: $\approx$ 333,333
- Flash-restore cycles: $\approx$ 125,000
- Total time: $\approx$ 2,933,333 s (about 34 days)

8-digit PIN (100,000,000 combinations)

- Reinsertion cycles: $\approx$ 33,333,333
- Flash-restore cycles: $\approx$ 12,500,000
- Total time:

$$100{,}000{,}000 \times 0.1\,\mathrm{s} + 33{,}333{,}333 \times 1\,\mathrm{s} + 12{,}500{,}000 \times 20\,\mathrm{s}$$

- Total: $\approx$ 316,666,666 s (about 10.05 years)

Thus, an 8-digit PIN increases brute-force time by roughly two orders of magnitude compared to a 6-digit PIN and makes offline brute forcing economically and practically infeasible even with full automation.

7 Mitigations

1. Monotonic OTP counters for anti-rollback.
2. Signed, versioned firmware/data with enforced monotonicity.
3. OTP-seeded PIN hashing (as implemented in commit 6c85421).
4. Enforce PIN complexity rules (minimum 6-digit PIN) as implemented in Token2's fork of pico-fido [5].
5. Counters stored in OTP/secure element.
6. Restrict BOOTSEL; note disabling BOOTSEL makes devices non-upgradeable without an alternate authenticated sideload channel.
7. Test thoroughly before irreversible OTP programming.

8 Note on PIN Complexity Enforcement vs User Choice

The PIN complexity feature in Token2's fork of pico-fido enforces a minimum PIN length to prevent users from selecting trivially guessable PINs (e.g., four-digit numeric codes), reducing offline brute-force risks.

Users of the original pico-fido firmware can already choose long and complex PINs, including 6-digit, 8-digit, or alphanumeric PINs. However, the upstream firmware does not enforce a minimum complexity level, so weak PINs may be chosen inadvertently. Token2's firmware acts as a safeguard against weak PIN selection, while still allowing users to select stronger PINs if desired.

9 Responsible Disclosure

The issue was reported to the maintainer, who confirmed it and implemented OTP-seeded PIN hashing [4]. Token2's fork additionally implements PIN complexity to mitigate offline guessing attacks [5]. Plans for full anti-rollback measures are under consideration.

10 Conclusion

Flash snapshot/restore via BOOTSEL reverts PINs and credentials on pico-fido RP2350 devices. Secure Lock and Secure Boot do not address rollback. OTP-seeded PIN hashing and PIN complexity improve security, but full rollback resistance still requires monotonic counters and signed, versioned updates. Token2's fork implements the PIN complexity mitigation [5].

References

[1] P. Henarejos, “polhenarejos/pico-fido,” GitHub, 2023. [Online]. Available: https://github.com/polhenarejos/pico-fido

[2] Raspberry Pi Ltd., “RP2350 Datasheet and Security Features,” Raspberry Pi Documentation, 2023. [Online]. Available: https://datasheets.raspberrypi.com/rp2350

[3] Pico Keys, “Pico-FIDO Product Information,” 2024. [Online]. Available: https://www.picokeys.com/

[4] P. Henarejos, “Use OTP secret as seed for PIN hash, when available (RP2350 / ESP32),” *pico-fido commit 6c85421*, GitHub, Sep. 2025. [Online]. Available: https://github.com/polhenarejos/pico-fido/commit/6c85421eca094d83a655c8622255cceb88f38607

[5] Token2, “Token2 pico-fido fork,” GitHub, 2025. [Online]. Available: https://github.com/token2/pico-fido/tree/main