

Penetration Test Report

Open Technology Fund

V 1.2
Amsterdam, November 10th, 2026
Confidential

Document Properties

Client	Open Technology Fund
Title	Penetration Test Report
Targets	• https://github.com/ooni/userauth/ • https://github.com/sigma-rs/sigma-proofs/ • https://github.com/sigma-rs/sigma-compiler/ • https://github.com/sigma-rs/cmz
Version	1.2
Pentester	Morgan Hill
Authors	Morgan Hill, Marcus Bointon
Reviewed by	Marcus Bointon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	July 28th, 2026	Morgan Hill	Initial draft
0.2	July 31st, 2026	Marcus Bointon	Review
1.0	July 31st, 2026	Marcus Bointon	1.0
1.1	November 5th, 2026	Morgan Hill	Retest
1.2	November 10th, 2026	Morgan Hill	Retest of regressions

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Radically Open Security B.V. is registered at the trade register of the Dutch chamber of commerce under number 60628081.

Table of Contents

1	**Executive Summary**	**4**
1.1	Introduction	4
1.2	Scope of Work	4
1.3	Project Objectives	5
1.4	Timeline	5
1.5	Results In A Nutshell	5
1.6	Summary of Findings	5
1.6.1	Findings by Threat Level	6
1.6.2	Findings by Type	7
1.7	Summary of Recommendations	7
2	**Methodology**	**9**
2.1	Planning	9
2.2	Risk Classification	9
3	**Findings**	**11**
3.1	OTF-017 — Replayable submit authorization proofs	11
3.2	OTF-002 — Probe context encoding is ambiguous	13
3.3	OTF-013 — Update handle crash	15
3.4	OTF-016 — Open registration crash	20
3.5	OTF-003 — Python bindings do not expose an upper measurement-count bound	24
4	**Non-Findings**	**26**
4.1	NF-007 — Cargo audit CMZ	26
4.2	NF-008 — Cargo audit UserAuth	26
4.3	NF-009 — Cargo geiger UserAuth	27
4.4	NF-011 — Cargo geiger sigma-proofs	27
5	**Future Work**	**28**
6	**Conclusion**	**29**
Appendix 1	**Testing Team**	**30**

1 Executive Summary

1.1 Introduction

Between May 4, 2026 and July 4, 2026, Radically Open Security B.V. carried out a penetration test for Open Technology Fund.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of Work

The scope of the penetration test was limited to the following targets:

- https://github.com/ooni/userauth/
- https://github.com/sigma-rs/sigma-proofs/
- https://github.com/sigma-rs/sigma-compiler/
- https://github.com/sigma-rs/cmz

The scoped services are broken down as follows:

- Scoping and preparation: 1 days
- Basic SAST scanning and dependency check: 0.5 days
- AI driven code review with human validation: 0.5 days
- Fuzzing: 1 days
- Constant-time audit: 2 days
- Validation of implementation against design: 4 days
- Manual analysis to evaluate anonymity: 1.5 days
- Communication: 1 days
- Reporting: 2 days
- PM & Review: 1-1.5 days
- Retest and fix verification (incl. update of the report): 0-2 days
- **Total effort: 14.5 - 17 days**

1.3 Project Objectives

ROS will perform an audit of OONI UserAuth with OTF in order to assess the security of UserAuth and its supporting libraries. To do so ROS will access the UserAuth source code and guide OTF in attempting to find vulnerabilities, exploiting any such found to try and gain further access and elevated privileges.

1.4 Timeline

The security audit took place between May 4, 2026 and July 4, 2026.

1.5 Results In A Nutshell

During this crystal-box penetration test we found 1 High, 1 Moderate, 2 Low and 1 N/A-severity issues.

The main finding of this report is that the authorization proofs are replayable OTF-017 (page 11). An attacker with one valid token can submit unlimited measurements with arbitrary payloads. We also found that the probe encoding is ambiguous OTF-002 (page 13).

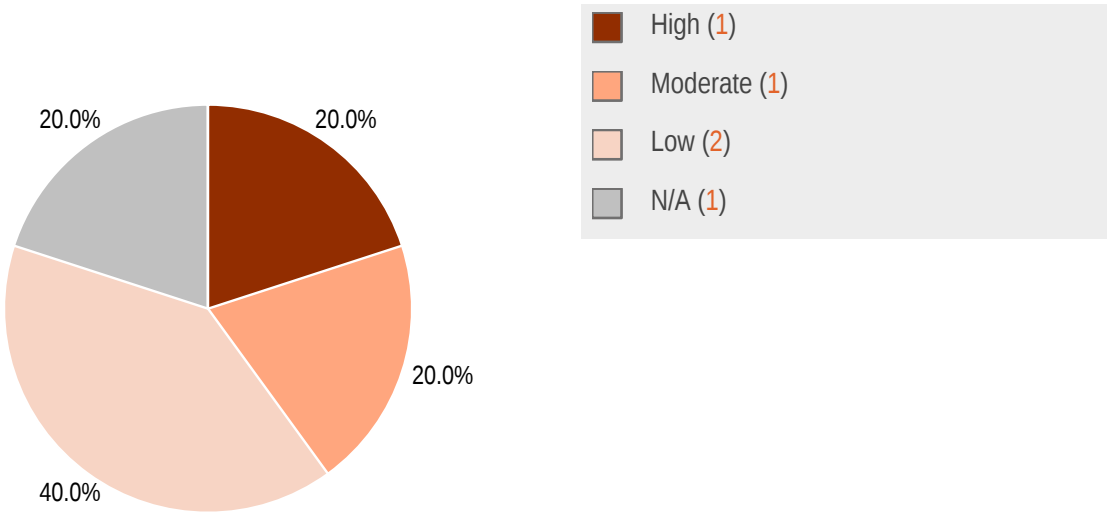
Our fuzzing efforts found some crashes stemming from insufficient result error handling in OTF-013 (page 15) and OTF-016 (page 20).

1.6 Summary of Findings

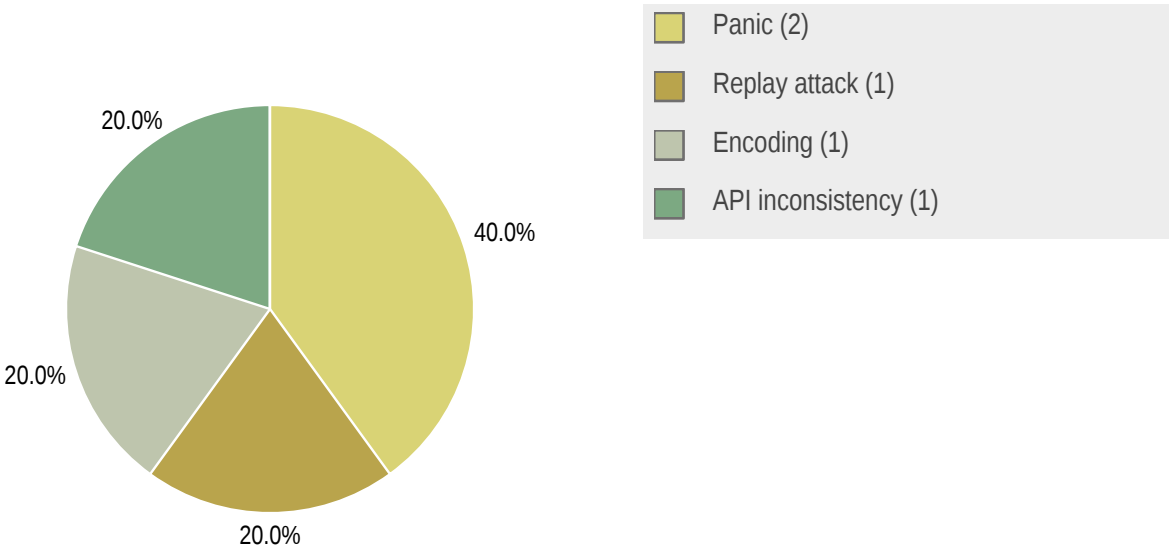
Info	Description
OTF-017 **High** **Type:** Replay attack **Status:** Resolved	A single valid SubmitRequest can be replayed indefinitely, allowing one submission authorization to be re-used an unlimited number of times within a probe context.
OTF-002 **Moderate** **Type:** Encoding **Status:** Resolved	probe_cc and probe_asn encodings can collide.
OTF-013 **Low** **Type:** Panic **Status:** Resolved	The update handler can be crashed with input to its public interface.
OTF-016 **Low** **Type:** Panic **Status:** Resolved	The open registration function can be crashed with input to its public interface.

OTF-003 **N/A** **Type:** API inconsistency **Status:** Wont Fix	Consumers of the python binding can't enforce a maximum measurement count.

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

Info	Recommendation
OTF-017 **High** **Type:** Replay attack **Status:** Resolved	• Implement measures to prevent authorization re-use.
OTF-002 **Moderate** **Type:** Encoding **Status:** Resolved	• Hash a canonical structured encoding of `(probe_cc, probe_asn)`, such as length-prefixed fields or a deterministic binary/JSON tuple. • Validate `probe_cc` and `probe_asn` at API boundaries. For example, require ISO country-code shape for `probe_cc` and a strict ASN grammar that excludes `/`. • Normalize case and representation before hashing.
OTF-013 **Low** **Type:** Panic **Status:** Resolved	• Allow the `Result` error to flow up the stack and be handled properly without a panic.
OTF-016 **Low** **Type:** Panic **Status:** Resolved	• Allow the `Result` error to flow up the stack and be handled properly without a panic.

OTF-003 **N/A** **Type:** API inconsistency **Status:** Wont Fix	• Add a `max_measurement_count` parameter to `make_submit_request` and `handle_submit_request`. • Pass `min_measurement_count..max_measurement_count` into core. • Add Python tests proving requests above the configured max are rejected.

2 Methodology

2.1 Planning

Our general approach during audit is as follows:

1. **Manual code review**
   We investigate code paths line by line and try to identify exploitable behaviours. As needed we build test harnesses to observe the code running under interesting conditions. We pay special attention to sensitive areas such as cryptographic implementations. We also check the implementation against the spec any discrepancies.
2. **Fuzzing**
   We employ fuzzers such as libfuzzer and AFL to explore code paths automatically searching for crashes. We typically fuzz parts of the public interface that are likely to be exposed. When a fuzzer finds a crash we then investigate it to find out the root cause and assess its exploitability.
3. **SAST**
   We use static analysis tools to gather a list of any common mistakes and look for application specific patterns. Anything we find with SAST we subject to further testing in order to identify if it qualifies as a weakness in the context of the application.

2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.
- **High**
  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**
  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**
  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.

- **Low**

  Low risk of security controls being compromised with measurable negative impacts as a result.

3 Findings

We have identified the following issues:

3.1 OTF-017 — Replayable submit authorization proofs

Vulnerability ID: OTF-017	**Status:** Resolved
Vulnerability type: Replay attack	
Threat level: High	

Description:

The OONI UserAuth submit protocol authorizes a measurement submission with a zero-knowledge proof that is not bound to any server-issued nonce or to the measurement content, and the server verifies it without keeping state. As a result, a single valid `SubmitRequest` can be replayed indefinitely, and because the proof is not tied to the measurement body, the same authorization can be attached to arbitrary attacker-chosen payloads. This lets one authorized submission be turned into an unlimited number of authorized submissions within a probe context.

Technical description:

The submit relation proves possession of a valid credential, carries `nym_id/age` forward, increments `measurement_count`, computes the domain pseudonym `NYM = nym_id · DOMAIN`, and enforces range bounds on age and measurement_count. The Fiat–Shamir transcript for this proof is seeded only with a hard-coded constant session identifier:

ooniauth-core/src/submit.rs:12 — `const SESSION_ID: &[u8] = b"submit";`

ooniauth-core/src/submit.rs:163 — client proves with the constant session id.

ooniauth-core/src/submit.rs:248-251 — server verifies with the same constant session id.

The proof therefore contains no server-generated challenge, request identifier, fine-grained timestamp, or digest of the measurement payload. Verification is also stateless: ServerState::handle_submit takes an immutable `&self` receiver, never records used proofs or challenges, never consumes the old credential, and never persists the incremented measurement_count. The counter advances only inside the client's own credential, and a replay re-sends the old request carrying the old count, so the server observes no monotonic progress. The only intrinsic bounds are (1) domain confinement — the proof is tied to `DOMAIN = H("ooni.org/{cc}/{asn}")`, so a captured token replays only within the same (cc, asn) — and (2) a coarse validity window from the server's age range check (e.g., ~30 days), which is trivially refreshed via re-registration or the update protocol. The PoC

`poc_submit_request_replay_is_accepted_twice` calls `handle_submit` twice with the same cloned request and the same `probe_id`; both calls return Ok.

Update :

Notes from OONI: Acknowledged. `SESSION_ID = b"submit"` was replaced in `44b490f` (on 11 May). The zk proof and measurement digest are now bound together, so a proof can no longer be re-attached to another payload. We consider this to be fixed in main.

Impact:

An attacker who obtains one valid submit token can submit an unlimited number of authorized measurements within the captured (cc, asn) context, using arbitrary payloads. This defeats the anti-abuse purpose of the authentication layer: measurement_count-based rate limiting is unenforceable, and sybil/flood resistance is bypassed. The headline risk is dataset poisoning: an adversary such as a censor can inject fabricated measurements attributed to a targeted country/network to mask real blocking or discredit genuine censorship signals, directly undermining the integrity of OONI's measurement data. Severity is high where the submit proof is treated as the authorization gate for accepting a measurement, reducing toward moderate only if an independent external layer (per-upload report IDs, upload-session nonces, server-side dedup, or per-nym rate limiting) provides replay protection — a defense that must not be assumed, given the crate's stateless, nonce-free design.

Recommendation:

- Bind each submit proof to a fresh, server-issued challenge/session identifier instead of the constant `b"submit"`, so proofs cannot be reused.
- Bind the measurement payload digest (and a canonical probe-context encoding) into the proof transcript, so an authorization cannot be reattached to different content.
- Make verification stateful: store and reject already-seen challenges or proof identifiers, and/or enforce `measurement_count` as consumed single-use state; also expose and enforce the `measurement_count` upper bound.
- Treat the credential measurement_count as advisory unless the server enforces single-use state, and document that this crate provides no replay protection on its own so integrators do not rely on it.

Update 2026-11-04:

fixed

Description:

ROS has reviewed the changed for this issue.

Technical description:

The session ID is now bound to the measurement hash therefore the proofs can no longer be reused: https://github.com/ooni/userauth/blame/00a66d7acb00f549d36eee3e66261965559111df/ooniauth-core/src/submit.rs#L207 https://github.com/ooni/userauth/blame/00a66d7acb00f549d36eee3e66261965559111df/ooniauth-core/src/submit.rs#L295

3.2 OTF-002 — Probe context encoding is ambiguous

Vulnerability ID: OTF-002 **Status:** Resolved

Vulnerability type: Encoding

Threat level: Moderate

Description:

`probe_cc` and `probe_asn` encodings can collide.

Technical description:

The cryptographic domain is a raw slash-delimited string. Distinct `(probe_cc, probe_asn)` tuples can produce the same domain string, for example:

- `("US", "AS1234/extra")`
- `("US/AS1234", "extra")`

Both encode to `ooni.org/US/AS1234/extra`.

Affected code:

- `ooniauth-core/src/submit.rs:89` : client computes `format!("ooni.org/{}/{}", probe_cc, probe_asn)`.
- `ooniauth-core/src/submit.rs:225` : server recomputes the same string.

```
#[test]
fn poc_submit_domain_string_collision_cross_context() {
    let rng = &mut rand::thread_rng();
    let server_state = ServerState::new(rng);
```

```
    let mut user_state = UserState::new(server_state.public_parameters());

    let (reg_request, reg_client_state) = user_state.request(rng).unwrap();
    let reg_response = server_state.open_registration(reg_request).unwrap();
    user_state
        .handle_response(reg_client_state, reg_response)
        .unwrap();

    let today = ServerState::today();
    let age_range = (today - 30)..(today + 1);
    let measurement_count_range = 0..100;

    let client_cc = "US".to_string();
    let client_asn = "AS1234/extra".to_string();
    let server_cc = "US/AS1234";
    let server_asn = "extra";
    assert_eq!(
        format!("ooni.org/{}/{}", client_cc, client_asn),
        format!("ooni.org/{}/{}", server_cc, server_asn)
    );

    let ((request, _client_state), nym) = user_state
        .submit_request(
            rng,
            client_cc,
            client_asn,
            age_range.clone(),
            measurement_count_range.clone(),
        )
        .unwrap();

    assert!(server_state
        .handle_submit(
            rng,
            request,
            &nym,
            server_cc,
            server_asn,
            age_range,
            measurement_count_range,
        )
        .is_ok());
}
```

Command:

```
cargo +stable test -p ooniauth-core poc_submit_domain -- --nocapture
```

Observed result:

```
running 2 tests
test submit::tests::poc_submit_domain_string_collision_cross_context ... ok
```

Update :

Notes from OONI: We acknowledge the issue. We've made sure that the caller can't invoke this function with ambiguous probe_cc/probe_asn, and that they are validated beforehand. This is now done in

`check_measurement_meta`, `ooni/backend` `47e63e1b` (on 19 May). Therefore, we consider this to be fixed in main.

Impact:

Domain-specific pseudonyms (`NYM`) and `probe_id` values can collide across semantically different probe contexts if inputs are not already strictly normalized and delimiter-free.

Recommendation:

- Hash a canonical structured encoding of `(probe_cc, probe_asn)`, such as length-prefixed fields or a deterministic binary/JSON tuple.
- Validate `probe_cc` and `probe_asn` at API boundaries. For example, require ISO country-code shape for `probe_cc` and a strict ASN grammar that excludes `/`.
- Normalize case and representation before hashing.

Update 2026-11-05:

partially-fixed

Description:

ROS has verfied the changes made relating to this finding.

Technical description:

There are now debug assertions to sanitize probe_cc and probe_asn: https://github.com/ooni/userauth/blob/00a66d7acb00f549d36eee3e66261965559111df/ooniauth-core/src/submit.rs#L81-L93 The collision is still possible in production but should be caught in development.

3.3 OTF-013 — Update handle crash

Vulnerability ID: OTF-013	**Status:** Resolved
Vulnerability type: Panic	
Threat level: Low	

Description:

The update handler can be crashed with input to its public interface.

Technical description:

The fuzz harness has discovered a crash in a macro here https://github.com/ooni/userauth/blob/795fa8647cee23d8901f612dd82018b97d70eea7/ooniauth-core/src/update.rs#L11 .

The panic is actually triggered by an explicit guard in sigma proofs when canonicalizing linear relations which returns a `Result` with an error: https://github.com/sigma-rs/sigma-proofs/blob/91753c339997408a5dce7672e28ffb522aaedd76/src/linear_relation/canonical.rs#L477

The `Result` is then unwrapped further up the stack.

The fuzzers can be found here: https://github.com/ooni/userauth/pull/49

Full fuzzer output:

```
thread '<unnamed>' (5850) panicked at /home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/src/update.rs:11:1:
called `Result::unwrap()` on an `Err` value: InvalidInstance { message: "Trivial kernel in this
 relation" }
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
==5850== ERROR: libFuzzer: deadly signal
    #0 0xc91889e07ff0  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x267ff0) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #1 0xc9188a200778  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x660778) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    <a href="#f2-probe-context-encoding-is-ambiguous"/> 0xc9188a1f79fc  (/home/test/ooni-2026/fuzz/
userauth/work/userauth/ooniauth-core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle
+0x6579fc) (BuildId: c95240c48974cd2004bd072bcd465624c9813616)
    <a href="#f3-python-bindings-do-not-expose-an-upper-measurement-count-bound"/> 0xf9a708ad68f4
 (linux-vdso.so.1+0x8f4) (BuildId: 3a7243f9730a676016844bb488b25156f3f7db98)
    #4 0xf9a708947604  (/lib/aarch64-linux-gnu/libc.so.6+0x87604) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
    #5 0xf9a7088fcb38  (/lib/aarch64-linux-gnu/libc.so.6+0x3cb38) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
    #6 0xf9a7088e7dfc  (/lib/aarch64-linux-gnu/libc.so.6+0x27dfc) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
    #7 0xc91889d6e264  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1ce264) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #8 0xc91889d6e098  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1ce098) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #9 0xc91889d69e40  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1c9e40) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #10 0xc9188a253a00  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x6b3a00) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #11 0xc9188a2471c0  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x6a71c0) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
```

```
    #12 0xc9188a23ea60  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x69ea60) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    <a href="#f13-update-handle-crash"/> 0xc9188a2478e0  (/home/test/ooni-2026/fuzz/userauth/
work/userauth/ooniauth-core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x6a78e0)
 (BuildId: c95240c48974cd2004bd072bcd465624c9813616)
    #14 0xc91889d6e90c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1ce90c) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #15 0xc91889d6e6d8  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1ce6d8) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    <a href="#f16-open-registration-crash"/> 0xc91889d6214c  (/home/test/ooni-2026/fuzz/userauth/
work/userauth/ooniauth-core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1c214c)
 (BuildId: c95240c48974cd2004bd072bcd465624c9813616)
    <a href="#f17-replayable-submit-authorization-proofs"/> 0xc91889d61078  (/home/test/
ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/fuzz/target/aarch64-unknown-linux-gnu/release/
update_handle+0x1c1078) (BuildId: c95240c48974cd2004bd072bcd465624c9813616)
    #18 0xc91889d5ff04  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1bff04) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #19 0xc91889d569f0  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1b69f0) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #20 0xc91889e67f58  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x2c7f58) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #21 0xc91889e7ed74  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x2ded74) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #22 0xc9188a1f111c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x65111c) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #23 0xc9188a1f1e68  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x651e68) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #24 0xc9188a1f0970  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x650970) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #25 0xc9188a1f7f2c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x657f2c) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #26 0xc9188a1fdd28  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x65dd28) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #27 0xc9188a1fed14  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x65ed14) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #28 0xc9188a1ffb7c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x65fb7c) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #29 0xc9188a218034  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x678034) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #30 0xc91889d6ebd0  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1cebd0) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)
    #31 0xf9a7088e84c0  (/lib/aarch64-linux-gnu/libc.so.6+0x284c0) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
    #32 0xf9a7088e8594  (/lib/aarch64-linux-gnu/libc.so.6+0x28594) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
```

```
    #33 0xc91889d6ed2c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/target/aarch64-unknown-linux-gnu/release/update_handle+0x1ced2c) (BuildId:
 c95240c48974cd2004bd072bcd465624c9813616)

NOTE: libFuzzer has rudimentary signal handlers.
      Combine libFuzzer with AddressSanitizer or similar for better crash reports.
SUMMARY: libFuzzer: deadly signal
MS: 3 CrossOver-InsertByte-InsertRepeatedBytes-; base unit: c4275fffe7867dc63085bfa4eebf35ef9cf79fb5
0x7e,0xa,0x2a,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,
0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0x0,0x0,
0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x8,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,
0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0
```

Impact:

When the service is exposed to the Internet an adversary may send a crafted request that results in a panic, leading to a denial of service.

Recommendation:

- Allow the `Result` error to flow up the stack and be handled properly without a panic.

Update `2026-11-05`:

open

Description:

ROS has retested this finding

Technical description:

The panic was reproduced on the main branch of userauth at `00a66d7acb00f549d36eee3e66261965559111df`.

Update `2026-11-10`:

fixed

Description:

ROS has retested this finding

Technical description:

The panic was no longer reproduced on the main branch of userauth at `ff179a361ca90d2a25f65ad36099c0044d2f1110`.

3.4 OTF-016 — Open registration crash

Vulnerability ID: OTF-016 **Status:** Resolved

Vulnerability type: Panic

Threat level: Low

Description:

The open registration function can be crashed with input to its public interface.

Technical description:

While fuzzing the open registration function we uncovered a crash in https://github.com/ooni/userauth/blob/795fa8647cee23d8901f612dd82018b97d70eea7/ooniauth-core/src/registration.rs#L85.

The panic happens in this unwrap, and is the same error as the sigma protocol error we reported in OTF-013 (page 15). The `Request::open_registration::try_from` is implemented by the `muCMZProtocol` macro: https://github.com/ooni/userauth/blob/795fa8647cee23d8901f612dd82018b97d70eea7/ooniauth-core/src/registration.rs#L38

The fuzzers are available here: https://github.com/ooni/userauth/pull/49

Fuzzer output:

```
#5056   NEW    cov: 1046 ft: 1454 corp: 12/269b lim: 43 exec/s: 0 rss: 53Mb L: 43/43 MS: 1
 InsertByte-

thread '<unnamed>' (10681) panicked at /home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/src/registration.rs:38:1:
called `Result::unwrap()` on an `Err` value: InvalidInstance { message: "Trivial kernel in this
 relation" }
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
==10681== ERROR: libFuzzer: deadly signal
    #0 0xb3291d987878  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x247878) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #1 0xb3291dd34668  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x5f4668) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    <a href="#f2-probe-context-encoding-is-ambiguous"/> 0xb3291dd2a8ec  (/home/test/ooni-2026/fuzz/
userauth/work/userauth/ooniauth-core/fuzz/target/aarch64-unknown-linux-gnu/release/registration_open
+0x5ea8ec) (BuildId: bb7fe8da9360a09ccb870eca825409034e9dd892)
    <a href="#f3-python-bindings-do-not-expose-an-upper-measurement-count-bound"/> 0xf52ac43318f4
 (linux-vdso.so.1+0x8f4) (BuildId: 3a7243f9730a676016844bb488b25156f3f7db98)
    #4 0xf52ac41a7604  (/lib/aarch64-linux-gnu/libc.so.6+0x87604) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
    #5 0xf52ac415cb38  (/lib/aarch64-linux-gnu/libc.so.6+0x3cb38) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
    #6 0xf52ac4147dfc  (/lib/aarch64-linux-gnu/libc.so.6+0x27dfc) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
```

```
    #7 0xb3291d8ebae4  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x1abae4) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #8 0xb3291d8eb918  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x1ab918) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #9 0xb3291d8e76c0  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x1a76c0) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #10 0xb3291dd888f0  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x6488f0) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #11 0xb3291dd7c0b0  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x63c0b0) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #12 0xb3291dd73950  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x633950) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    <a href="#f13-update-handle-crash"/> 0xb3291dd7c7d0  (/home/test/ooni-2026/fuzz/userauth/work/
userauth/ooniauth-core/fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x63c7d0)
 (BuildId: bb7fe8da9360a09ccb870eca825409034e9dd892)
    #14 0xb3291d8ec18c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x1ac18c) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #15 0xb3291d8ebf58  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x1abf58) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    <a href="#f16-open-registration-crash"/> 0xb3291da12a68  (/home/test/ooni-2026/fuzz/userauth/
work/userauth/ooniauth-core/fuzz/target/aarch64-unknown-linux-gnu/release/registration_open
+0x2d2a68) (BuildId: bb7fe8da9360a09ccb870eca825409034e9dd892)
    <a href="#f17-replayable-submit-authorization-proofs"/> 0xb3291da12308  (/home/test/
ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/fuzz/target/aarch64-unknown-linux-gnu/release/
registration_open+0x2d2308) (BuildId: bb7fe8da9360a09ccb870eca825409034e9dd892)
    #18 0xb3291d8d9078  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x199078) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #19 0xb3291d9fd4a4  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x2bd4a4) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #20 0xb3291d9b8b80  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x278b80) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #21 0xb3291d9bad60  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x27ad60) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #22 0xb3291dd24010  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x5e4010) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #23 0xb3291dd24d5c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x5e4d5c) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #24 0xb3291dd23864  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x5e3864) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #25 0xb3291dd2ae1c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x5eae1c) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #26 0xb3291dd31c18  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x5f1c18) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
```

```
    #27 0xb3291dd32c04  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x5f2c04) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #28 0xb3291dd33a6c  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x5f3a6c) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #29 0xb3291dd4bf24  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x60bf24) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #30 0xb3291d8ec450  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x1ac450) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)
    #31 0xf52ac41484c0  (/lib/aarch64-linux-gnu/libc.so.6+0x284c0) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
    #32 0xf52ac4148594  (/lib/aarch64-linux-gnu/libc.so.6+0x28594) (BuildId:
 d5ef86dde36cbd3289566cf5098226035d76f2e1)
    #33 0xb3291d8ec5ac  (/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/
fuzz/target/aarch64-unknown-linux-gnu/release/registration_open+0x1ac5ac) (BuildId:
 bb7fe8da9360a09ccb870eca825409034e9dd892)

NOTE: libFuzzer has rudimentary signal handlers.
      Combine libFuzzer with AddressSanitizer or similar for better crash reports.
SUMMARY: libFuzzer: deadly signal
MS: 3 CrossOver-InsertByte-CrossOver-; base unit: adc83b19e793491b1c6ea0fd8b46cd9f32e592fc
0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,
0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0xbd,0xa,
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000
\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\275\012
artifact_prefix='/home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-core/fuzz/artifacts/
registration_open/'; Test unit written to /home/test/ooni-2026/fuzz/userauth/work/userauth/ooniauth-
core/fuzz/artifacts/registration_open/crash-65c49588a953807ec7d0ceab7bfc661069fb7fbd
Base64: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC9Cg==

_____________________________________________________________________________________

Failing input:

    fuzz/artifacts/registration_open/crash-65c49588a953807ec7d0ceab7bfc661069fb7fbd

Output of `std::fmt::Debug`:

    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 189, 10]
```

Update :

Notes from OONI: Acknowledged, with one correction: the `unwrap()` is in sigma-compiler's generated `protocol()`, not the CMZ macro. Now fixed on `mu/draft-v3` in `998e45d` (10 Aug). The same input now returns `CliProofFailed`. It will hit main soon.

Impact:

When the service is exposed to the Internet an adversary may send a crafted request that results in a panic, leading to a denial of service.

Recommendation:

- Allow the `Result` error to flow up the stack and be handled properly without a panic.

Update `2026-11-05`:

open

Description:

ROS has retested this finding

Technical description:

The panic was reproduced on the main branch of userauth at `00a66d7acb00f549d36eee3e66261965559111df`.

Update `2026-11-10`:

fixed

Description:

ROS has retested this finding

Technical description:

The panic was no longer reproduced on the main branch of userauth at `ff179a361ca90d2a25f65ad36099c0044d2f1110`.

3.5 OTF-003 — Python bindings do not expose an upper measurement-count bound

Vulnerability ID: OTF-003	**Status:** Wont Fix
Vulnerability type: API inconsistency	
Threat level: N/A	

Description:

Consumers of the python binding can't enforce a maximum measurement count.

Technical description:

The Rust core supports `measurement_count_range: Range<u32>` and enforces both lower and upper bounds. The Python API accepts only `min_measurement_count` and passes `min_measurement_count..u32::MAX` on both the client and server paths.

Affected code:

- `ooniauth-py/src/protocol.rs:122-130`
- `ooniauth-py/src/protocol.rs:240-246`
- `ooniauth-py/ooniauth_py.pyi:45-53`
- `ooniauth-py/ooniauth_py.pyi:77-83`

Update :

Notes from OONI: We understand the issue, but we don't think this is a problem or represent a security issue. In fact, quite the opposite, having fixed one of the argument has been intentional to prevent misuse of the credential.

Impact:

The Rust API exposes the ability to configure the max measurement count, but this ability is not exposed via the Python API. Therefore, Python clients do not have the ability to limit the max measurement count via the API and must implement the limit themselves if desired.

Recommendation:

- Add a `max_measurement_count` parameter to `make_submit_request` and `handle_submit_request`.

- Pass `min_measurement_count..max_measurement_count` into core.
- Add Python tests proving requests above the configured max are rejected.

Update `2026-11-04`:

risk-accepted

Description:

ROS re-verified the code related to this finding.

Technical description:

The Python API remains the same at the current HEAD of main `00a66d7acb00f549d36eee3e66261965559111df`.

4 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

4.1 NF-007 — Cargo audit CMZ

`1a4bfb1766f3fa5a321e8b4396f8c7f9ff42dc19`

```
Crate:     bincode
Version:   1.3.3
Warning:   unmaintained
Title:     Bincode is unmaintained
Date:      2025-12-16
ID:        RUSTSEC-2025-0141
URL:       https://rustsec.org/advisories/RUSTSEC-2025-0141

Crate:     paste
Version:   1.0.15
Warning:   unmaintained
Title:     paste - no longer maintained
Date:      2024-10-07
ID:        RUSTSEC-2024-0436
URL:       https://rustsec.org/advisories/RUSTSEC-2024-0436

Crate:     rand
Version:   0.8.5
Warning:   unsound
Title:     Rand is unsound with a custom logger using `rand::rng()`
Date:      2026-04-09
ID:        RUSTSEC-2026-0097
URL:       https://rustsec.org/advisories/RUSTSEC-2026-0097

Crate:     rand
Version:   0.10.0
Warning:   unsound
Title:     Rand is unsound with a custom logger using `rand::rng()`
Date:      2026-04-09
ID:        RUSTSEC-2026-0097
URL:       https://rustsec.org/advisories/RUSTSEC-2026-0097

warning: 4 allowed warnings found
```

4.2 NF-008 — Cargo audit UserAuth

`187594915bf2450b244a3bb1e7907b34453e01b6`

```
Crate:     bincode
Version:   1.3.3
Warning:   unmaintained
Title:     Bincode is unmaintained
Date:      2025-12-16
ID:        RUSTSEC-2025-0141
URL:       https://rustsec.org/advisories/RUSTSEC-2025-0141
```

```
Crate:     paste
Version:   1.0.15
Warning:   unmaintained
Title:     paste - no longer maintained
Date:      2024-10-07
ID:        RUSTSEC-2024-0436
URL:       https://rustsec.org/advisories/RUSTSEC-2024-0436

Crate:     rand
Version:   0.8.5
Warning:   unsound
Title:     Rand is unsound with a custom logger using `rand::rng()`
Date:      2026-04-09
ID:        RUSTSEC-2026-0097
URL:       https://rustsec.org/advisories/RUSTSEC-2026-0097

Crate:     rand
Version:   0.10.0
Warning:   unsound
Title:     Rand is unsound with a custom logger using `rand::rng()`
Date:      2026-04-09
ID:        RUSTSEC-2026-0097
URL:       https://rustsec.org/advisories/RUSTSEC-2026-0097

warning: 4 allowed warnings found
```

4.3 NF-009 — Cargo geiger UserAuth

ooniauth-core does not directly use unsafe, however it does not explicitly forbid unsafe code. If there is no intention to use unsafe in the code base then it would be prudent to forbid unsafe. Several dependencies use unsafe which is typical and nothing to be overly concerned about. The full output can be found in the uploads folder of the pentest repository.

4.4 NF-011 — Cargo geiger sigma-proofs

sigama-proofs does not directly use unsafe, however it does not explicitly forbid unsafe code. If there is no intention to use unsafe in the code base then it would be prudent to forbid unsafe. Several dependencies use unsafe which is typical and nothing to be overly concerned about. The full output can be found in the uploads folder of the pentest repository.

5 Future Work

- **Retest of findings**
  When mitigations for the vulnerabilities described in this report have been deployed, perform a repeat test to ensure that they are effective and have not introduced other security problems.
- **Regular security assessments**
  Security is a process that must be continuously evaluated and improved; this penetration test is just a single snapshot. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security.

6 Conclusion

We discovered 1 High, 1 Moderate, 2 Low and 1 N/A-severity issues during this penetration test.

This code audit focused on OONIs UserAuth library and the supporting sigma-compiler, sigma-proofs, and CMZ libraries. We looked at the Rust implementation, the bindings to other languages (python), and the cryptographic implementation.

OONI UserAuth provides a framework for anonymous authorization, based on sigma proofs, to authorize the submission of measurements from OONI probes. It is important that OONI's database of Internet censorship data is trustworthy. UserAuth provides a critical step in the trustworthiness of the submitted measurements.

We found that the authorization proofs are replayable and that the probe encoding is ambiguous. Overall, we consider the proofs and cryptographic implementation sound. We expect that these issues are relatively minor in the production application.

We also uncovered some crashes via fuzzing. We believe fuzzing is a valuable approach for finding bugs in unexpected places. We have made our fuzzing harnesses available for the developers to use and build on in future.

This has been an extremely interesting project to work on. The combination of technologies and Internet freedom have been extremely motivating. We look forward to future opportunities to work with OONI.

We recommend fixing all of the issues found and then performing a retest in order to ensure that mitigations are effective and that no new vulnerabilities have been introduced.

Finally, we want to emphasize that security is a process – this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Appendix 1 Testing Team

Morgan Hill *(pentester)*	Morgan is a seasoned security consultant with a background in IoT and DevOps. He currently specialises in Rust and AVoIP.
Melanie Rieback *(approver)*	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.

Front page image by dougwoods (https://www.flickr.com/photos/deerwooduk/682390157/), "Cat on laptop", Image styling by Patricia Piolon, https://creativecommons.org/licenses/by-sa/2.0/legalcode.