Security Audit Report

Stacks - LibSigner

April 2025

V202411

Executive Summary

CoinFabrik was asked to audit the **LibSigner** component for the **Stacks** project.

During this audit we found two high severity issues and one low severity issue. Also, one enhancement was proposed.

All the issues were resolved or acknowledged.

Scope

The audited files are from the git repository located at https://github.com/stacks-network/stacks-core.git. The audit is based on the commit `c1a1f50fddcbc11054fae537103423e21221665a`.

The scope for this audit includes and is limited to the following files:

- **`src/errors.rs`**: Error definition
- **`src/events.rs`:** Handler for signer events
- **`src/http.rs`:** Decoding of http messages sent to signer server
- **`src/libsigner.rs`:** Top-level src file of libsigner
- **`src/runloop.rs`:** Top-level runloop, containing main signer logic
- **`src/session.rs`:** StackerDB session manager
- **`src/signer_set.rs`:** Definitions for stacks signer set.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

Each severity label is detailed in the Severity Classification section. Additionally, the statuses are explained in the Issues Status section.

Id	Title	Severity	Status
HI-01	Incorrect Event Public Key	High	**Resolved**
HI-02	Chunk size limit bypass	High	**Acknowledged**
LO-01	Trivial Service Shutdown	Low	**Acknowledged**

Critical Severity Issues

No issues found.

High Severity Issues

HI-01 Incorrect Event Public Key

Location

- `./src/event.rs:[430]`

Classification

- CWE-684: Incorrect Provision of Specified Functionality[1]

Description

The TryFrom implementation for StackerDBChunksEvent to SignerEvent overwrites the variable miner_pk with each valid chunk's PK.

We can see that in the for loop:

```
            let mut miner_pk = None;
            for chunk in event.modified_slots {
                let Ok(msg) = T::consensus_deserialize(&mut chunk.data.as_slice()) else {
                    Continue;
                };

                miner_pk = Some(chunk.recover_pk().map_err(|e| {
                    EventError::MalformedRequest(format!(
                        "Failed to recover PK from StackerDB chunk: {e}"
                    ))
                })?);
```

[1]https://cwe.mitre.org/data/definitions/684.html

```
                messages.push(msg);
            }
            SignerEvent::MinerMessages(messages,
miner_pk.ok_or(EventError::EmptyChunksEvent)?)
```

Here we see that the signer event will contain only the last valid key, leading to inconsistent PK usage if chunks have different keys. Messages may be incorrectly attributed to the last valid chunk's PK, risking invalid signature verification or DoS.

Recommendation

Store each PK with each message.

Status

Resolved. The signer no longer relies on public keys from events. Fix committed on `4d299ed3ad420ca8d8d0bcef5c1817c2d121032d`.

HI-02 Chunk size limit bypass

Location

- `./src/session.rs:[188.252]`

Classification

- CWE-770: Allocation of Resources Without Limits or Throttling[2]

Description

The `get_chunk()` and `put_chunk()` functions in the stackerDB session manager fail to enforce the chunk size limit. Only `get_latest_chunks()` function correctly enforces chunk size limits. This allows retrieval of oversized chunks, causing memory exhaustion during deserialization, leading to Denial of Service.

We can see this simple size limit check on `get_latest_chunks()`:

```
// Verify that the chunk is not too large
                    if body_bytes.len() > limit {
                        None
                    } else {
                        Some(body_bytes)
                    }
```

[2] https://cwe.mitre.org/data/definitions/770.html

Is missing on `get_chunk()` and `put_chunk()`, `allowing unlimited sized chunk retrieval from stackerDB.`

Recommendation

Enforce size limits on all StackerDB chunk manager functions.

Status

Acknowledged. This size is checked by the StackerDB implementation. Coinfabrik still recommends adding the size check in the libsigner project as validations must be done on the library-side too, following the defensive programming technique.

Medium Severity Issues

No issues found.

Low Severity Issues

LO-01 Trivial Service Shutdown

Location

- `./src/event.rs:[315]`

Classification

- CWE-400: Uncontrolled Resource Consumption [3]

Description

The `next_event` function lacks authentication for critical endpoints like `/shutdown`, enabling unauthenticated users to remotely terminate any utility that includes `libsigner`, leading to a Denial-of-Service (DoS) vulnerability.

However, by issuing this simple command:

```
~$ curl -x POST host:30000/shutdown
```

[3] https://cwe.mitre.org/data/definitions/400.html

This event will be caught by the libsigner event loop and will cause any utility that uses (including signer-binary) to shutdown.

None of the commands are authenticated and they use highly insecure plain HTTP for communication, but this might be accepted as the signer is expected to run in a trusted environment. However, trivially shutting down the service from anywhere in the local network can be considered a high-security issue.

Recommendation

Implement authentication for the shutdown command, or limit it to be used only from a trusted host like localhost or a whitelisted admin IP.

Status

Acknowledged. Because it's considered a trusted endpoint, only trusted callers should have any access to these network RPC endpoints.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

Id	Title	Status
EN-01	CRLF Injection in RPC Request	Not implemented

EN-01 CRLF Injection in RPC Request

Location

- `./src/http.rs:[239]`

Description

The `run_http_request()` function lacks validation for CRLF injection in user-provided verb, path, and host parameters, enabling HTTP header injection and potential request smuggling or response

splitting attacks.

We can see in this code snippet how the HTTP request is constructed:

```
format!("{} {} HTTP/1.1\r\nHost: {}\r\nConnection: close\r\n{}User-Agent:
libsigner/0.1\r\nAccept: */*\r\n\r\n",
verb, path, host, content_length_hdr
)
```

An attacker could set the path parameter to **`"/path HTTP/1.1\r\nInjected-Header: value\r\n\r\n"`**, injecting arbitrary headers into the request. Particularly, the `rpc_request()` function uses this vulnerable http request, and while no vulnerable use of this function was detected, it's highly recommended to sanitize all inputs to it.

Recommendation

Sanitize verb, path, and host inputs by stripping or escaping CRLF sequences (\r, \n, %0D, %0A).

Status

Not implemented.

About CoinFabrik

CoinFabrik is a research and development company specialized in Web3, with a strong background in cybersecurity. Founded in 2014, we have worked on over 500 decentralization projects, including EVM-based and other platforms like Solana, Algorand, and Polkadot. Beyond development, we offer security audits through a dedicated in-house team of senior cybersecurity professionals, working on code in languages such as Substrate, Solidity, Clarity, Rust, TEAL, and Stellar Soroban.

Our team has an academic background in computer science, software engineering, and mathematics, with accomplishments including academic publications, patents turned into products, and conference presentations. We actively research in collaboration with universities worldwide, such as Cornell, UCLA, and École Polytechnique in Paris, and maintain an ongoing collaboration on knowledge transfer and open-source projects with the University of Buenos Aires, Argentina. Our management and people experience team has extensive expertise in the field.

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior, and general documentation about the project. Our auditors spent two weeks

auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses.

- Arithmetic errors
- Race conditions
- Misuse of block timestamps
- Denial of service attacks
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Severity Classification

Security risks are classified as follows[4]:

Critical	• Manipulation of governance voting result deviating from voted outcome and resulting in a direct change from intended effect of original results • Direct theft of any user funds, whether at-rest or in-motion, other than unclaimed yield • Direct theft of any user NFTs, whether at-rest or in-motion, other than unclaimed royalties • Permanent freezing of funds • Permanent freezing of NFTs • Unauthorized minting of NFTs • Predictable or manipulable RNG that results in abuse of the principal or NFT • Unintended alteration of what the NFT represents (e.g. token URI, payload, artistic content) • Protocol insolvency
High	• Theft of unclaimed yield • Theft of unclaimed royalties • Permanent freezing of unclaimed yield • Permanent freezing of unclaimed royalties • Temporary freezing of funds • Temporary freezing NFTs

[4] This classification is based on Immunefi severity classification system version 2.3. https://immunefi.com/immunefi-vulnerability-severity-classification-system-v2-3/

Medium	• Smart contract unable to operate due to lack of token funds • Block stuffing • Griefing (e.g. no profit motive for an attacker, but damage to the users or the protocol) • Theft of gas • Unbounded gas consumption • Security best practices not followed
Low	• Contract fails to deliver promised returns, but doesn't lose value • Other security issues with minor impact

Issue Status

An issue detected by this audit has one of the following statuses:

- **Unresolved:** The issue has not been resolved.
- **Resolved:** Adjusted program implementation to eliminate the risk.
- **Partially Resolved:** Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Acknowledged:** The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **Mitigated:** Implemented actions to minimize the impact or likelihood of the risk.

Disclaimer

This audit report has been conducted on a **best-effort basis within a tight deadline defined by time and budget constraints**. We reviewed only the specific code provided by the client at the time of the audit, detailed in the Scope section. We do not review other components that are part of the solution: neither implementation, nor general design, nor business ideas that motivate them.

While we have employed the latest tools, techniques, and methodologies to identify potential vulnerabilities, **this report does not guarantee the absolute security of the contracts, as undiscovered vulnerabilities may still exist.** Our findings and recommendations are

suggestions to enhance security and functionality and are not obligations for the client to implement.

The results of this audit are valid solely for the code and configurations reviewed, and any modifications made after the audit are outside the scope of our responsibility. CoinFabrik disclaims all liability for any damages, losses, or legal consequences resulting from the use or misuse of the applications, including those arising from undiscovered vulnerabilities or changes made to the codebase after the audit.

This report is intended exclusively for the **Stacks** team and should not be relied upon by any third party without the explicit consent of CoinFabrik. Blockchain technology and smart contracts are inherently experimental and involve significant risk; users and investors should fully understand these risks before deploying or interacting with the audited contracts.

Changelog

Date	Description
4 April 2025	Initial report based on commit `c1a1f50fddcbc11054fae537103423e21221665a`.
8 April 2025	Comments and fixes from the Stacks Team
30 April 2025	Final report based on commit `4d299ed3ad420ca8d8d0bcef5c1817c2d121032d`