Stacks Signer Binary Audit

April 4, 2024

By CoinFabrik

v202311

Executive Summary

CoinFabrik was asked to audit the contracts for the Stacks Signer Binary project.

During this audit we found one critical issue, one high issue, two medium issues and a minor issue. Also, an enhancement was proposed.

Scope

The audited files are from the git repository located at https://github.com/stacks-network/stacks-core/tree/next/stacks-signer. The audit is based on the commit `45c80a417907d757c4f17a3bd0597874587eb8b0`. During the audit, the code updated up to commit 0e9cf5dc1b6f66d1c184c2f4924fdf2c37706f2d that was the final audited commit.

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

- `stacks-signer/src/cli.rs`: Stacks-signer cli command parser
- `stacks-signer/src/config.rs`: configuration manager
- `stacks-signer/src/coordinator.rs: node coordinator module`
- `stacks-signer/src/main.rs: Top cli file`
- `stacks-signer/src/runloop.rs: Signer command loop`
- `stacks-signer/src/signer.rs: Signer main file`
- `stacks-signer/src/signerdb.rs: Signer block database manager`
- `stacks-signer/src/client/stackerdb.rs: Stackerdb client`
- `stacks-signer/src/client/stacks_client.rs: Stacks node client`
- `stacks-signer/src/client/mod.rs: Client module top file`

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.

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 four weeks auditing the source code provided, and then an additional week auditing updated code, 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
- Excessive gas usage
- 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

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.

ID	Title	Severity	Status
CR-01	Lack of StackDB chunk verification	Critical	Unresolved
HI-01	Denial Of Service Reading StackDB Chunks	High	Unresolved
ME-01	Improper Validation of BlockValidationResponse Event	Medium	Unresolved
ME-02	SignerDB Database Always Grows	Medium	Unresolved
MI-01	Manipulation Of Terminal Via Binary Data Chunk	Minor	Unresolved

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. Blocking bugs are also included in this category. They must be fixed **immediately**.
- **High:** These refer to a vulnerability that, if exploited, could have a substantial impact, but requires a more extensive setup or effort compared to critical issues. These pose a significant risk and **demand immediate attention**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but might be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

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

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

Critical Severity Issues

CR-01 Lack of StackDB chunk verification

Location:

- `stacks-signer/src/main.rs:168,175,182`

Classification:

- CWE-345: Insufficient Verification of Data Authenticity[1]

Chunks retrieved from StackerDB are not verified, so any malicious StackerDB instance (or any user in the network, as connection is not encrypted) could inject any data and impersonate any user of StackerDB.

Commands get-chunk, get-latest-chunk and list-chunks are affected, as well as the signer server and every function that uses Stackerdb data.

Recommendation

Verify every chunk signature. As signatures in StackerDB are recoverable, no additional data is needed apart from that stored on the StackerDB metadata.

Status

Unresolved

High Severity Issues

HI-01 Denial Of Service Reading StackDB Chunks

Location:

- `stacks-signer/src/main.rs:171,178,185`

Classification:

- CWE-400: Uncontrolled Resource Consumption[2]

While reading from StackerDB, the stacks-signer binary does not control for the size of data reading, allowing a malicious StackerDB instance to send unlimited data, that will cause stacks-signer to use all CPU and memory assigned, causing it to be killed by the operating system.

[1]https://cwe.mitre.org/data/definitions/345.html
[2]https://cwe.mitre.org/data/definitions/400.html

To trigger this issue, a simple HTTP server that sends infinite data is needed. An example server named evilserver-dos.py is provided. Instructions are:

1) Start evilserver-dos:

```
	$ python3 evilserver-dos.py
```

2) Retrieve a chunk from evilserver:

```
	$ ./stacks-signer get-chunk --host 127.0.0.1:8000 --contract
SP5N7QWC3BGM58AV63VK1HDRN217DH8JK42M292G.test --slot-id 1234 --slot-version 1
```

This will cause the signer-binary to consume all CPU and RAM, and crash in some seconds.

Recommendation

Limit the amount of data that is retrieved from StackerDB to the maximum chunk size.

Status

Unresolved

Medium Severity Issues

ME-01 Improper Validation of BlockValidationResponse Event

Location:

- `stacks-signer/src/signer.rs:1218`

Classification:

- CWE-400: Uncontrolled Resource Consumption[3]

The stacks-singer binary parses a BlockValidationResponse event for a reward cycle that does not belong to the signer. As a consequence it is possible to waste the signer-db storage inserting unused blocks.

ME-02 SignerDB Database Always Grows

Location:

- `stacks-signer/src/signerdb.rs:131`

[3]https://cwe.mitre.org/data/definitions/20.html

Classification:

- CWE-20: Improper Input Validation[4]

function `remove_block()` removes a block from the database, but this function is not called from anywhere in the code. As this is the only way to clean old blocks, it means the database will always grow, as there is no way to remove unused data, eventually consuming all storage.

Recommendation

Implement a method to remove old and unused data from the SignerDB database.

Status

Unresolved

Minor Severity Issues

MI-01 Manipulation Of Terminal Via Binary Data Chunk

Location:

- `stacks-signer/src/main.rs:171,178,185`

Classification:

- CWE-116: Improper Encoding or Escaping of Output[5]

The Stacks-signer commands get-chunk, get-latest-chunk and list-chunks retrieve chunks from StackerDB and print them to the system console. But the system console can be manipulated by binary control sequences, causing output data to be hidden or modified.

As an example, we provide a file called evilchunk.dat, with the following data:

```
EVIL

[2A[K2e1608dfa6e0b5569232559e3d385fea5a931122e1608dfa6e0b5569:232559e3d385fea5a93112
```

But when using stacks-signer to read this chunk, this is the output of the console:

```
$ ./stacks-signer get-chunk --host
2e1608dfa6e0b5569232559e3d385fea5a931122e1608dfa6e0b5569:232559e3d385fea5a93112
```

Note that the “`EVIL`” data is not printed, because it's hidden using the terminal control sequences “`[2A[K`”.

[4]https://cwe.mitre.org/data/definitions/400.html
[5]https://cwe.mitre.org/data/definitions/116.html

To trigger this issue, a simple HTTP server that sends infinite data is needed. An example server named evilserver-dos.py is provided. Instructions are:

1) Start evilserver:

```
        $ python3 evilserver.py
```

2) Retrieve a chunk from evilserver:

```
        $ ./stacks-signer get-chunk --host 127.0.0.1:8000 --contract
SP5N7QWC3BGM58AV63VK1HDRN217DH8JK42M292G.test --slot-id 1234 --slot-version 1
```

This will incorrectly print the chunk and trigger the issue.

Recommendation

Filter escape sequences, or preferentially, escape all binary data from the output to the terminal.

Status

Unresolved

Enhancements

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

ID	Title	Status
EN-01	Incorrect Debug Message At Event Parsing	Not implemented

EN-01 Incorrect Debug Message at Event Parsing

Location:

- `stacks-signer/src/signer.rs:1218`

The validation code is:

```
                if *signer_set != self.stackerdb.get_signer_set() {
                    debug!("{self}: Received a signer message for a reward cycle that does not belong
to this signer. Ignoring...");
                    return Ok(());
                }
```

But clearly this validation is for the `signer_set`, not the `reward cycle`.

Recommendation

Modify the debug message to reflect the correct validation operation.

Status

Not Implemented

Changelog

- 2024-05-04 – Initial report based on commit 0e9cf5dc1b6f66d1c184c2f4924fdf2c37706f2d.