StackerDB Audit

26 feb 2024

By CoinFabrik

v202311

Executive Summary

CoinFabrik was asked to audit the contracts for the StackerDB module of the Stacks project.

StackerDB is a replicated database that stores commitment data from signers, but can be used as a generic storage for the Stacks blockchain..

Is has two components:

- Smart Contracts
- Nodes that store data.

Instead of on-chain storage, data is stored off chain using a relational database. This audit corresponds to the node off-chain storage.

During this audit we found one high, one medium and one minor issue.

Scope

The audited files are from the git repository located at https://github.com/stacks-network/stacks-core/tree/master/stackslib/src/net/stackerdb. The audit is based on the commit `4227debecea99001cba91096076b526f4a618d67`.

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

- `stacks-core/stackslib/src/net/stackerdb/config.rs`: Smart-Contract interface
- `stacks-core/stackslib/src/net/stackerdb/db.rs`: DB storage handler
- `stacks-core/stackslib/src/net/stackerdb/mod.rs`: Main module
- `stacks-core/stackslib/src/net/stackerdb/sync.rs`: Replication logic

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 three 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
- 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
HI-01	Arbitrary TCP Connections Via Hint-Replicas	High	Unresolved
ME-01	Denial Of Service Via Unlimited Storage Use	Medium	Unresolved
MI-01	Missing Input Validation Of Parameters	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

No issues found.

High Severity Issues

HI-01 Arbitrary TCP Connections Via Hint-Replicas

Location:

- `stacks-core/stackslib/src/net/stackerdb/config.rs:380`

Classification:

- CWE-20: Improper Input Validation[1]

One of the parameters defined by the StackerDB smart contract function `stackerdb-get-config` is `Hint-Replicas`:

```
hint-replicas: (list 128 { addr: (list 16 uint), port: uint, public-key-hash: (buff 20) })
```

The hint-replicas list containing IP:port:pubkey will be integrated into the replication peer node list. Subsequently, the node will attempt to establish a TCP connection with the specified IP, provided the port number exceeds 1024.

Since the `ip:port` pair lacks verification, the smart contract can indicate any IP address, potentially incorporating private local network addresses or random internet services. In this way, a malicious contract can initiate connections with arbitrary IPs and ports. Even if its capabilities remain restricted to an initial connection and handshake, this might still impact services if numerous connections are established simultaneously, abusing the replication mechanism.

Recommendation

Ideally the IPs in the hint-replicas list should be checked against an authorized white-list of valid nodes. Another simpler solution is to check that the list of hint-replicas should not include private IP networks like 192.168.0.0/24.

Status

Unresolved

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

Medium Severity Issues

ME-01 Denial Of Service Via Unlimited Storage Use

Location:

- `stacks-core/stackslib/src/net/stackerdb/db.rs:366`

Classification:

- CWE-400: Uncontrolled Resource Consumption[2]

According to the limits on config.rs, every StackerDB smart contract can have a maximum of `STACKERDB_INV_MAX` slots (4096), each one with a maximum size of `STACKERDB_MAX_CHUNK_SIZE` bytes (16 MB).

This results in a maximum storage of 64 GB for each smart contract. As there are no further limitations on the amount of StackerDB smart contracts, a few smart contracts can take all the storage space of the node. For example, as few as 15 smart contracts can claim up to 1 TB of storage, causing a denial of service if the node runs out of storage space.

Recommendation

Establish a global maximum slot amount, so the StackerDB system stops allocating space before the storage space runs out.

Status

Unresolved

Minor Severity Issues

MI-01 Missing Input Validation Of Parameters

Location:

- `stacks-core/stackslib/src/net/stackerdb/config.rs:259`

Classification:

- CWE-20: Improper Input Validation[3]

In the current implementation of StackerDB, certain configuration parameters assessed within the `eval_config()` function lack appropriate validations. Consequently, these insufficiently controlled settings might lead to incorrect functionality within the system.:

1. The parameter `chunk-size` has no minimal value, it can be zero thus preventing any data to be stored.

[2] https://cwe.mitre.org/data/definitions/400.html
[3] https://cwe.mitre.org/data/definitions/20.html

2. Parameters `write-freq` and `max_writes` also can be zero, preventing any writes.
3. Parameter `max_neighbors` can be zero, and also the maximum limit is too big, causing the node to create too many outgoing connections, consuming excessive system resources.
4. Finally, the length of the `hint_replicas` and `slot_list` lists are also unlimited, but they may be limited by the size of the transactions.

Recommendation

Validate that all parameters have reasonable upper and lower limits.

Status

Unresolved

Centralization

No significant centralization or administrative accounts were found in the provided code.

Changelog

- 2023-06-26 – Initial report based on commit `4227debecea99001cba91096076b526f4a618d67`.