



Security Audit Report

Stone – Smart Contracts

March 2025

Executive Summary	3
Scope	3
Findings	3
Critical Severity Issues	4
High Severity Issues	4
HI-01 Asset Loss in Contest Entry	4
HI-02 Slippage Risk in Swap Function	4
Medium Severity Issues	5
ME-01 Percentage Settings Can Lead to Transaction Failures	5
Low Severity Issues	6
LO-01 Percentage Distribution Inconsistency	6
LO-02 Lack of Validation for Complementary Percentages	6
Other Considerations	7
Centralization	7
Upgrades	7
Hardcoded Liquidity Pool ID	8
About CoinFabrik	8
Methodology	8
Severity Classification	9
Issue Status	10
Disclaimer	11
Changelog	11

Executive Summary

CoinFabrik was asked to audit **Stone's Smart Contracts**.

During this audit we found two high issues, one medium issue and two low severity issues.

All the issues were resolved.

Scope

The audited files are from the git repository located at <https://github.com/KM-StoneWallet/Smart-Contracts/>. The audit is based on the commit **b47888bea85020cc638caa0d0515419df158aacf**. Fixes were reviewed on commit **3212293b45b51a18d0cec5c03767b26d01167aa8**.

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

- **./contracts/stone-burst.clar**: Item purchase system with multiple revenue streams.

Fixes for the reported findings were implemented in **./contracts/StoneBurstFixes.clar**.

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	Asset Loss in Contest Entry	High	Resolved
HI-02	Slippage Risk in Swap Function	High	Resolved
ME-01	Percentage Settings Can Lead to Transaction Failures	Medium	Resolved

Id	Title	Severity	Status
LO-01	Percentage Distribution Inconsistency	Low	Resolved
LO-02	Lack of Validation for Complementary Percentages	Low	Resolved

Critical Severity Issues

No issues found.

High Severity Issues

HI-01 Asset Loss in Contest Entry

Location

- `./contracts/stone-burst.clar:304,305`

Description

The enter-contest function overwrites the user's existing gems and hearts inventories instead of adding to them. When a user enters a contest, any previously purchased gems and hearts are permanently lost as their inventory is reset to the fixed contest entry values (3 hearts and 10 gems). This creates a severe disincentive for users to participate in contests if they have already purchased items and represents a significant asset loss vulnerability.

Recommendation

Add the contest entry items to the user's existing inventory rather than overwriting it.

Status

Resolved. Fixed according to the recommendation.

HI-02 Slippage Risk in Swap Function

Location

- `./contracts/stone-burst.clar:72`

Description

The contract implements swap functionality with the minimum output amount parameter set to `u1`. This effectively disables slippage protection, as the swap would be considered successful even if it returns only 1 unit of the output token regardless of the expected amount. This leaves the contract vulnerable to sandwich attacks and front-running, where malicious actors could manipulate the market immediately before and after the transaction to extract value.

Recommendation

Implement meaningful slippage protection by calculating a minimum output amount based on the expected return.

Status

Resolved. Fixed according to the recommendation.

Medium Severity Issues

ME-01 Percentage Settings Can Lead to Transaction Failures

Location

- `./contracts/stone-burst.clar:60, 222, 267`

Classification

- CWE-20: Improper Input Validation¹

Description

The contract allows the owner to modify percentage variables via setter functions (e.g., `set-percentage-progressive`). These percentages determine how STX amounts are distributed in the `buy-gems` and `buy-hearts` functions. However, there are no constraints ensuring that these percentages result in non-zero shares, which can cause transactions to fail. These functions will revert if `share-to-burn` is zero due to the swap function's assertion.

Recommendation

Require `percentage-progressive < u100` to ensure `remaining-stx > 0` and require `percentage-burn > u0` to ensure `share-to-burn > 0`.

¹ <https://cwe.mitre.org/data/definitions/20.html>

Status

Resolved. Fixed according to the recommendation.

Low Severity Issues

LO-01 Percentage Distribution Inconsistency

Location

- `./contracts/stone-burst.clar:72`

Classification

- CWE-682: Incorrect Calculation²

Description

The contract uses inconsistent scaling for percentage calculations. The progressive percentage is set to `u10` and divided by 100, representing 10%. The other percentages (`multisig: 7950`, `burn: 50`, `kryptomind: 2000`) are divided by 10000, representing 79.5%, 0.5%, and 20% respectively. This inconsistent scaling leads to incorrect fund distribution and does not adhere to the total percentage principle.

Recommendation

Standardize percentage calculations by using the same scaling factor for all percentages.

Status

Resolved. Fixed according to the recommendation.

LO-02 Lack of Validation for Complementary Percentages

Location

- `./contracts/stone-burst.clar`

Classification

- CWE-20: Improper Input Validation

² <https://cwe.mitre.org/data/definitions/682.html>

Description

The contract distributes funds using a two-step percentage allocation process: first applying the progressive percentage, then distributing the remaining amount using three complementary percentages (multisig, burn, and kryptomind). However, the contract lacks validation to ensure these three secondary percentages sum to 100% (represented as 10000 in the contract) when they are individually modified.

Without validation, the owner could inadvertently set percentages that sum to less than or more than 100%. This could result in either funds being trapped in the contract (if less than 100%) or attempted distribution of more funds than are available (if more than 100%), potentially causing transaction failures. Over time, this could lead to significant accounting discrepancies and incorrect fund allocation across the various wallets.

Recommendation

Implement validation when updating any of the secondary percentages to verify that they collectively sum to exactly 100% of the remaining funds. This ensures mathematical consistency in the distribution model and prevents inadvertent fund loss or transaction failures.

Status

Resolved. Fixed according to the recommendation.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Centralization

The contract grants a single owner unrestricted control over all fund distribution wallets, revenue percentages, and ownership transfer.

Upgrades

The contract does not implement a mechanism for upgrading the contract.

Hardcoded Liquidity Pool ID

The contract uses a hardcoded liquidity pool ID (u79) for all swap operations. While this is a deliberate design choice favoring immutability, it introduces a resilience risk. If the pool with ID 79 becomes compromised, drained, or deprecated, the contract would need to be redeployed rather than simply updated.

In the revision commit, this was modified allowing the owner to change the pool id.

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. Our auditors spent three days 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 runtime 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

After delivering a report with our findings, the development team had the opportunity to comment on every finding and fix the issues they considered convenient. Once fixed and/or commented, our team ran a second review process to verify that the changes to the code effectively solve the issues found and do not unintentionally add new ones. This report includes the final status after the second review.

Severity Classification

Security risks are classified as follows³:

 Critical	<ul style="list-style-type: none">• 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
--	--

³ This classification is based on the smart contract Immunefi severity classification system version 2.3. <https://immunefi.com/immunefi-vulnerability-severity-classification-system-v2-3/>

High	<ul style="list-style-type: none">• 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
Medium	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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 smart contract 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 smart contracts, including those arising from undiscovered vulnerabilities or changes made to the codebase after the audit.

This report is intended exclusively for the **Stone** 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
2025-03-11	Initial report based on commit b47888bea85020cc638caa0d0515419df158aacf.
2025-03-14	Final report based on commit 3212293b45b51a18d0cec5c03767b26d01167aa8.