PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

April 10, 2025

ABSTRACT

In this report, we consider the security of smart contracts of Stabiliti project. Our task is to find and describe security issues in the smart contracts of the platform.

DISCLAIMER

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

SUMMARY

In this report, we considered the security of Stabiliti smart contracts. We described the audit process in the section below.

The initial audit revealed two low severity issues. They do not endanger project security.

After the audit, the developers provided us with an updated version of the code. In that update, they fixed all the issues.

The overall code quality is good.

GENERAL RECOMMENDATIONS

We do not have any recommendations.

PROJECT OVERVIEW

Project description

For the audit, we were provided with Stabiliti project on a private GitHub repository, commit 56ab55690bb82cf4a41f582e9bef8918b256cd03.

The scope of the audit included all smart contracts in the **contracts** folder.

The documentation for the project included a pdf file, shasum **d0ef42d0696f9d5dd20caae214612d78a19b3e4c.**

All 29 tests pass successfully. The code coverage is 100%.

The total LOC of audited sources is 211.

Codebase update

After the audit, the developers provided us with a commit 990663a1bfdcc475adba1dfc77bdd7412e5cc7b7. In that commit, they fixed all the issues and implemented additional tests.

All 30 tests pass successfully. The code coverage is 100%.

AUDIT PROCESS

We started and finished the audit on February 25, 2025.

We inspected the materials provided for the audit. Based on that, we identified the following key points for the audit:

- Token compliance with the ERC-20 standard;
- The correctness of role assignments;
- The possibility of blacklisted addresses bypassing restrictions.

We manually analyzed all the contracts within the scope of the audit and checked their logic. We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

On February 26, 2025, the developers provided us with an updated version of the code. In that update, they fixed all the issues, added tests, and updated the documentation.

We reviewed the updated codebase and did not find any new issues.

MANUAL ANALYSIS

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

The audit showed no issues of medium severity.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Misleading NatSpec comments (fixed)

NatSpec comments of some functions mention that they can only be called by the default admin. However, that is not the case in the following functions:

- `rescueTokens` function and `RESCUE_TOKENS_ROLE` role;
- `destroyBlockedTokens` function and `DESTROY_BLOCKED_TOKENS_ROLE` role;
- `enableMint` function and `MINT_BURN_ENABLER_ROLE` role;
- `enableBurn` function and `MINT_BURN_ENABLER_ROLE` role;
- `disableMint` function and `MINT_BURN_DISABLER_ROLE` role;
- `disableBurn` function and `MINT_BURN_DISABLER_ROLE` role;
- `addBlacklist` function and `ADD_BLACKLIST_ROLE` role;
- `removeBlacklist` function and `REMOVE_BLACKLIST_ROLE` role;
- `addLostAddress` function and `ADD_LOST_ADDRESS_ROLE` role;
- `removeLostAddress` function and `REMOVE_LOST_ADDRESS_ROLE` role.

In addition to the above inconsistency, `transfer` and `transferFrom` functions mention the transfer fee. However, the fee is not present in the code.

There is also a possible typo in the NatSpec comment for the `burn` function. According to line 125, it can only be called by the minter smart contract. However, the role name suggests that it should be called by the burner smart contract.

> **The issues have been fixed and are not present in the latest version of the code.**

L02. Max supply can be lower than the totalSupply (fixed)

User with the `DEFAULT_ADMIN_ROLE` is able to change the maximum supply of the token on the chain. However, in the current implementation it is possible to make it lower than the `totalSupply` of the token.

> **The issue has been fixed and is not present in the latest version of the code.**

Notes

N01. Missing Approval event (fixed)

`Approval` event is not emitted when the allowance is decreased with the `decreaseAllowance` function. While this does not contradict the ERC-20 standard, consider adding it in analogy with `increaseAllowance` and `approve` functions.

> The issue has been fixed and is not present in the latest version of the code.

N02. Privileged roles (commented)

The project contains several privileged roles with crucial responsibilities:

- `DEFAULT_ADMIN_ROLE` is responsible for granting and revoking other roles, enabling and disabling token transfers, changing the maximum token supply, and upgrading the contract;
- `MINTER_ROLE` is responsible for minting tokens to any address;
- `BURNER_ROLE` is responsible for burning tokens from any address;
- `MINT_BURN_ENABLER_ROLE` and `MINT_BURN_DISABLER_ROLE` are responsible for enabling and disabling minting or burning;
- `ADD_BLACKLIST_ROLE` and `REMOVE_BLACKLIST_ROLE` are responsible for adding and removing users from the blacklist;
- `DESTROY_BLOCKED_TOKENS_ROLE` can burn all tokens from a blacklisted address;
- `ADD_LOST_ADDRESS_ROLE` and `REMOVE_LOST_ADDRESS_ROLE` are responsible for adding and removing addresses from the list of lost addresses;
- `RESCUE_TOKENS_ROLE` can burn tokens from lost addresses.

> The developers commented that this behavior meets the code requirements.

This analysis was performed by Pessimistic:

Daria Korepanova, Senior Security Engineer
Pavel Kondratenkov, Senior Security Engineer
Irina Vikhareva, Project Manager
Alexander Seleznev, CEO

April 10, 2025