# SWELL STAKING SECURITY AUDIT REPORT

April 10, 2024

MixBytes()

# TABLE OF CONTENTS

# 1. INTRODUCTION

## 1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

## 1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

### 1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals

- Build an independent view of the project's architecture.
- Identifying logical flaws.

### 2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

## 3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

## 4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

## 5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

### 6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

## Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

| Severity | Description |
|---|---|
| Critical | Bugs leading to assets theft, fund access locking, or any other loss of funds. |
| High | Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement. |
| Medium | Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds. |
| Low | Bugs that do not have a significant immediate impact and could be easily fixed. |

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

| Status | Description |
|---|---|
| Fixed | Recommended fixes have been made to the project code and no longer affect its security. |
| Acknowledged | The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future. |

## 1.3 Project Overview

The staking part of the Swell protocol allows users to stake any whitelisted tokens to receive additional rewards in the SWELL token in the future. The protocol is going to calculate off-chain rewards based on the user's staked amount.

## 1.4 Project Dashboard

### Project Summary

| Title | Description |
|---|---|
| Client | Swell |
| Project name | Staking |
| Timeline | April 01 2024 - April 08 2024 |
| Number of Auditors | 3 |

### Project Log

| Date | Commit Hash | Note |
|---|---|---|
| 01.04.2024 | 5b421683380c3aa4077454ce6177f3d3ac24cb9b | Commit for the audit |
| 08.04.2024 | b30a92b068d5875e86ce5f0f34248b1506e1c343 | Commit for the reaudit |

### Project Scope

The audit covered the following files:

| File name | Link |
|---|---|
| src/SimpleStakingERC20.sol | SimpleStakingERC20.sol |
| src/Zap.sol | Zap.sol |

### Deployments

| File name | Contract deployed on mainnet | Comment |
|---|---|---|
| SimpleStakingERC20.sol | 0x38D43a...8531d774 | |
| Zap.sol | 0xBD9fc4...002aDe0d | |

## 1.5 Summary of findings

| Severity | # of Findings |
|---|---|
| Critical | 0 |
| High | 0 |
| Medium | 0 |
| Low | 2 |

| ID | Name | Severity | Status |
|---|---|---|---|
| L-1 | Solmate `SafeTransferLib` doesn't check for a token contract existence | Low | Fixed |
| L-2 | `ISimpleStakingERC20.Withdraw` is emitted with incorrect `staker` field | Low | Fixed |

## 1.6 Conclusion

During the audit process, 2 LOW findings have been spotted. All findings were fixed by the client. Additionally, we reviewed the code against an internal checklist that covers aspects such as business logic issues, common ERC-20 issues, attack vectors related to interactions with external contracts, calculations overflow, reentrancy attacks, and access control.

The current scope is very well written regarding the ease of understanding and reading the code. All functions can be easily comprehended. Overall, the audit identified a few potential vulnerabilities and areas of improvement, which were documented in the detailed findings section of this report. Following those recommendations will enhance contracts security and robustness.

# 2.FINDINGS REPORT

## 2.1 Critical

Not Found

## 2.2 High

Not Found

## 2.3 Medium

Not Found

## 2.4 Low

| L-1 | Solmate `SafeTransferLib` doesn't check for a token contract existence |
|---|---|
| **Severity** | Low |
| **Status** | Fixed in b30a92b0 |

**Description**

There is a known issue described here. It is possible to have a non-existent token contract at the specified address, which won't be checked by the `SafeTransferLib` functions as it is done in the OpenZeppelin `SafeERC20` library. `SafeTransferLib` functions are used at the lines SimpleStakingERC20.sol#53, SimpleStakingERC20.sol#67 and SimpleStakingERC20.sol#85.

**Recommendation**

We recommend taking into account that possible issue while whitelisting new tokens in the `SimpleStakingERC20` contract.

**Client's commentary**

> Acknowledged. Switched from Solmate implementation to OpenZeppelin.

|   |   |
|---|---|
| L-2 | `ISimpleStakingERC20.Withdraw` is emitted with incorrect `staker` field |
| **Severity** | Low |
| **Status** | Fixed in b30a92b0 |

### Description

Event `ISimpleStakingERC20.Withdraw` is emitted with `_receiver` as `staker` in `SimpleStakingERC20.withdraw()`:
SimpleStakingERC20.sol#L87
But the actual staker is `msg.sender`.

### Recommendation

We recommend fixing the event emition to:
`emit Withdraw(_token, msg.sender, _amount);`

### Client's commentary

> Acknowledged. Recommendation applied.

# 3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

## Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes