SOLIDIFIED

Audit Report for FreshCut - March 9, 2022

Summary

Audit Report prepared by Solidified covering the FreshCut FCD Polygon token and vesting smart contracts.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief on 7 March 2022.

Audited Files

The source code has been supplied in the form of two GitLab repositories:

https://github.com/freshcutgg/fcd-token-contracts

Commit number: `04a3269177f06c01b0f20b7ca6293ec89fb9c967`

The scope of the audit was limited to the following files:

```
contracts
├── FCDToken.sol
├── FCDToken_TestContractForUpgradability.sol
└── Vesting.sol
```

Intended Behavior

The smart contracts implement an upgradable ERC-20 token and a related vesting solution.

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than in a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	Medium	Whilst no additional documentation has been provided, most of the codebase builds on the excellently documented OpenZeppellin library.
Test Coverage	High	-

SOLIDIFIED

Audit Report for FreshCut - March 9, 2022

Issues Found

Solidified found that the FreshCut contracts contain no critical issues, 1 major issue, 1 minor issue, in addition to 1 informational note.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	Vesting Wallet inability to deal with ETH could lead to locked funds	Major	Resolved
2	Dependencies with well-known security vulnerabilities in the build system	Minor	Acknowledged
3	VestingWallet.soll: Missing parameter verification	Note	-

SOLIDIFIED

Audit Report for FreshCut - March 9, 2022

Critical Issues

No critical issues have been identified

Major Issues

1. Vesting Wallet inability to deal with ETH could lead to locked funds

The contract's documentation through code comments indicate that it should be able to handle ETH as a vesting asset. It also implements a `payable` function to receive ETH. However, the contract does not implement any further logic to handle ETH. Therefore, any ETH sent to the contract will be stuck forever.

Recommendation
Consider disallowing sending ETH to the contract by removing the fallback function or implementing full support for ETH.

Minor Issues

2. Dependencies with well-known security vulnerabilities in the build system

The build and test system relies on a number of outdated JavaScript libraries with well-known security vulnerabilities, some of which are critical. Since these are only used for deployment and testing they do not constitute a smart contract security risk. However, outdated dependencies in the build system make operational security incidents, such as key leakage more likely.

Recommendation
Consider Using `npm audit` to identify vulnerable dependencies and update them.

Informational Notes

3. VestingWallet.soll: Missing parameter verification

The `constructor` function of the contract does not place any bounds on the `startTimestamp` and `durationSeconds` parameters. This allows vesting schedules to be created in the past. Whilst having a start time in the past might be desired behavior, a guard to prevent that the end time has not passed already might help to prevent mistakes when creating vesting wallets.

Recommendation
Consider checking that the current timestamp exceeds the sum of `startTimestamp` and `durationSeconds`.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of FreshCut or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors from legal and financial liability.

Oak Security GmbH