SashimiSwap
Smart Contracts Audit

SMCAuditors

https://smcauditors.com / info@smcauditors.com

28th of December 2020

This document is the audit of SashimiSwap smart contracts performed by SMCAuditors.

1. Executive Summary

This report was written to provide a security audit for the SashimiSwap smart contracts. SMCAuditors conducted the audit focusing on whether SashimiSwap smart contracts are designed and implemented in accordance with publicly released information and whether it has any security vulnerabilities. The contracts as stated are listed below with their links.

- MasterChef.sol
  - https://github.com/SashimiProject/sashimiswap/blob/master/contracts/MasterChef.sol
- SashimiToken.sol
  - https://github.com/SashimiProject/sashimiswap/blob/master/contracts/SashimiToken.sol
- Timelock.sol

    - https://github.com/SashimiProject/sashimiswap/blob/master/contracts/Timelock.sol
- GovernorAlpha.sol
    - https://github.com/SashimiProject/sashimiswap/blob/master/contracts/GovernorAlpha.sol
- SashimiBar.sol
    - https://github.com/SashimiProject/sashimiswap/blob/master/contracts/SashimiBar.sol
- SashimiMaker.sol
    - https://github.com/SashimiProject/sashimiswap/blob/master/contracts/SashimiMaker.sol

We have run extensive static analysis of the codebase as well as standard security assessment utilising industry approved tools. There are no high level issues with the currently deployed contracts. Our medium and low level findings are available in the next section.

2. Audit Findings

MasterChef.sol, Timelock.sol, GovernorAlpha.sol

Medium Level Findings

1. migrate() is dependent on a currently unspecified migrator contract. The migrator can be set at any time by the owner. In the worst case, this could be set to a contract that transfers all underlying LP tokens to a new address. Since the owner is a timelock contract, consider setting the timelock of 2 days for users to mass exit the pool before the change could occur.
2. *emergencyWithdraw()* does not follow the checks-effects-interactions. The values for *user.amount* and *user.rewardDebt* are not being set to 0 until after the external call to *pool.lpToken.safeTransfer()*. If a compromised token is added, reentrancy may allow the theft of tokens.

Low Level Findings

1. If some LP token is added to the contract twice using function *add*, then the total amount of reward in function updatePool will be incorrect. We recommend validation placed here.

2. *massUpdatePools()* can run out-of-gas if too many tokens are added. Considering keeping the number low or validating the amount of gas.

3. Conclusion

In this audit, we thoroughly analyzed the SashimiSwap smart contracts. Overall, the smart contracts were well written and common security standards were used by the team. Our identified issues were informed and resolved by the team.

4. Disclaimer

This report is not advice on investment, nor does it guarantee adequacy of a business model and/or a bug-free code. This report should be used only to discuss known technical problems. It will be necessary to resolve addressed issues and conduct thorough tests to ensure the safety of the smart contract.