verichains

SECURITY AUDIT OF

FTMLAUNCH SMART CONTRACT

Public Report

Dec 21, 2021

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report prepared by Verichains Lab on Dec 21, 2021. We would like to thank the FTMlaunch for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the FTMlaunch Smart Contract. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified no vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About FTMlaunch Smart Contract

FTMlaunch is the First Decentralized and Permissionless Crowdfunding platform launching the next generation of disruptive applications on the Fantom chain with an ambient goal of building an inclusive Defi infrastructure in the Fantom Ecosystem.

FTMlaunch will design a completely Decentralized and Permissionless platform for connecting early investors to project devs that wish to launch on Fantom.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the FTMlaunch Smart Contract. It was conducted on the source code provided by the FTMlaunch team.

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The initial review was conducted on Dec 19, 2021 and a total effort of 3 working days was dedicated to identifying and documenting security issues in the code base of the FTMlaunch Smart Contract.

The following file was made available in the course of the review:

FILE	SHA256 SUM
Campaign.sol	8908265db8fd55d07a3291dc4bd6f6ccbdb5621f555e8634d25fc3714df9076e

2.2. Contract codes

The FTMlaunch Smart Contract was written in Solidity language, with the required version to be 0.6.12.

The contract is used to support to release tokens in the Initial DEX Offering.

There are four phases during the contract works. The contract will handle 4 logical ido phases according to time as below:

- RegisterPhase: initial phase, users can register tiers. To register a tier, the user has to stake amount of specific tokens which was chosen by the Factory.
- TierSalePhase: sale phase, the users who registered successfully in the previous phase can buy tokens with amount of the tier.
- FCFSPhase: After TierSalePhase, if the collectedFTM is still less than hardcap, the registered users can purchase more tokens that were missed from some other registered users. After this phase, if the contract has any problems, the Factory may set cancelled state variable to allow the user to receive the native token which they used to buy tokens.
- FinishSalePhase: anyone can call finishUp function, the native token will be delivered to feeAdrress and campaignOwner. All unsoldTokens will be sent to the burnAdrress or campaignOwner. After that, the Factory or campaignOwner can call setTokenClaimable function to allow the users to get the token corresponding to the native token they bought.

2.3. Findings

During the audit process, the audit team found no vulnerability in the given version of the FTMlaunch Smart Contract.

2.4. Additional notes and recommendations

2.4.1. Outdated version of Solidity INFORMATIVE

The required version of Solidity in the FTMlaunch Smart Contract source code is 0.6.12, which is outdated.

Updating the Solidity version from 0.6.12 to 0.8.10 will introduce many features and fix some bugs like:

- Automatic integer overflow/underflow checking for arithmetic operations.
- Allow overrides to have stricter state mutability: view can override nonpayable and pure can override view.
- Disallow public state variables overwriting pure functions.

2.4.2. StakerContract may cause problems in the future INFORMATIVE

Since we do not control the logic of the StakerContract, there is no guarantee that StakerContract will not contain any security related issues. With the current context, in case the StakerContract is compromised, there is not yet a way to exploit the FTMlaunch Smart Contract, but we still note that here as a warning for avoiding any related issue in the future.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 21, 2021*	Public Report	Verichains Lab

Table 2. Report versions history