SECURITY AUDIT OF

FARMME SMART CONTRACTS

Public Report

Jan 07, 2022

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 Jan 07, 2022. We would like to thank the FarmMe 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 FarmMe Smart Contracts. 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 FarmMe Smart Contracts

FARM Me is a fantasy metaverse exploring all aspects of "Play, Connect, Experience, Earn" for multiplayer to experience wholeset of the first immersive farm game with a battle-to-survive element.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of FarmMe Smart Contracts. It was conducted on commit 405cb395ef62ab99bc70eee147110045c9ef9ab2 from git repository *https://github.com/FarmGameVillas/farmme-smartscontracts/tree/master/contracts*.

There are 2 files in our audit scope. They are FarmMe.sol and FarmMeVestingFactory.sol files.

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 30, 2021 and a total effort of 5 working days was dedicated to identifying and documenting security issues in the code base of the FarmMe Smart Contracts.

The following files were made available in the course of the review: FarmMe.sol and FarmMeVestingFactory.sol files.

2.2. Contract code

The FarmMe Smart Contracts was written in Solidity language, with the required version to be ^0.8.2 in the FarmMe token contract and ^0.8.0 in the FarmMeVestingFactory contract. The source code was written based on OpenZeppelin's library.

2.2.1. FarmMe token contract

FarmMe contract is an ERC20 token contract. Besides the default ERC20 functions, the contract implements additional functions which allow the owner of the contract to block the suspected address. The contract also inherits the Pausable contract so the owner can pause or unpause the activities of the contract.

This table lists some properties of the FarmMe token contract (as of the report writing time).

PROPERTY	VALUE
Name	FarmMe
Symbol	FARMME
Decimals	18
Max Supply	1,100,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 1,100,000,000.

Table 2. The FarmMe token contract properties

2.2.2. FarmMeVestingFactory contract

FarmMeVestingFactory is a contract that inherits the VestingWallet contract. In the contract, the owner can create new VestingWallets that release the tokens following the tokenomics.

2.3. Findings

During the audit process, the audit team found no vulnerability in the given version of FarmMe Smart Contracts.

APPENDIX

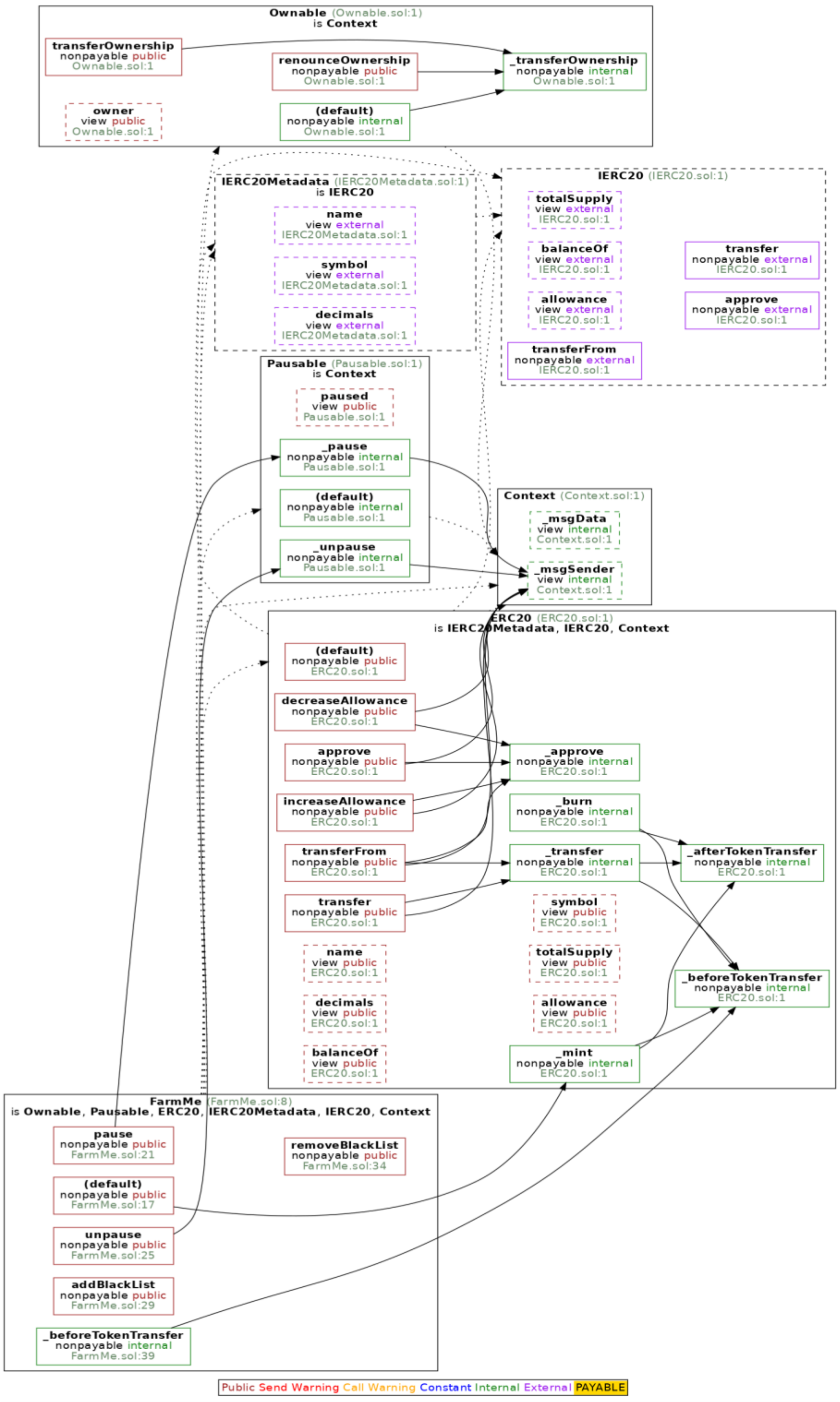
Ownable (Ownable.sol:1) is Context
transferOwnership nonpayable public Ownable.sol:1
renounceOwnership nonpayable public Ownable.sol:1
_transferOwnership nonpayable internal Ownable.sol:1
owner view public Ownable.sol:1
(default) nonpayable internal Ownable.sol:1
IERC20Metadata (IERC20Metadata.sol:1) is IERC20
name view external IERC20Metadata.sol:1
symbol view external IERC20Metadata.sol:1
decimals view external IERC20Metadata.sol:1
IERC20 (IERC20.sol:1)
totalSupply view external IERC20.sol:1
balanceOf view external IERC20.sol:1
transfer nonpayable external IERC20.sol:1
allowance view external IERC20.sol:1
approve nonpayable external IERC20.sol:1
transferFrom nonpayable external IERC20.sol:1
Pausable (Pausable.sol:1) is Context
paused view public Pausable.sol:1
_pause nonpayable internal Pausable.sol:1
(default) nonpayable internal Pausable.sol:1
_unpause nonpayable internal Pausable.sol:1
Context (Context.sol:1)
_msgData view internal Context.sol:1
_msgSender view internal Context.sol:1
ERC20 (ERC20.sol:1) is IERC20Metadata, IERC20, Context
(default) nonpayable public ERC20.sol:1
decreaseAllowance nonpayable public ERC20.sol:1
approve nonpayable public ERC20.sol:1
_approve nonpayable internal ERC20.sol:1
increaseAllowance nonpayable public ERC20.sol:1
_burn nonpayable internal ERC20.sol:1
transferFrom nonpayable public ERC20.sol:1
_transfer nonpayable internal ERC20.sol:1
_afterTokenTransfer nonpayable internal ERC20.sol:1
transfer nonpayable public ERC20.sol:1
symbol view public ERC20.sol:1
name view public ERC20.sol:1
totalSupply view public ERC20.sol:1
_beforeTokenTransfer nonpayable internal ERC20.sol:1
decimals view public ERC20.sol:1
allowance view public ERC20.sol:1
balanceOf view public ERC20.sol:1
_mint nonpayable internal ERC20.sol:1
FarmMe (FarmMe.sol:8) is Ownable, Pausable, ERC20, IERC20Metadata, IERC20, Context
pause nonpayable public FarmMe.sol:21
removeBlackList nonpayable public FarmMe.sol:34
(default) nonpayable public FarmMe.sol:17
unpause nonpayable public FarmMe.sol:25
addBlackList nonpayable public FarmMe.sol:29
_beforeTokenTransfer nonpayable internal FarmMe.sol:39
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. FarmMe token Smart contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jan 07,2022*	Public Report	Verichains Lab

Table 3. Report versions history