verichains

SECURITY AUDIT OF

PEPE TOKEN

Public Report

Jun 14, 2024

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 was prepared by Verichains Lab on Jun 14, 2024. We would like to thank the CorePepe 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 PEPE Token. 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 issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About PEPE Token

PEPE Token, a meme coin established on the Coredao blockchain, features an orange Pepe the Frog.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the PEPE Token that was deployed on Coredao Blockchain.

The latest version was made available in the course of the review:

FIELD	VALUE
Address Deploy	0xC0E5f8E2a9a2f1FBf34ab4d5FbF417ffd02Fdb79
Tx Deploy	0x347924ca4fdbade8795c9a8825149307d31a9454b1100655fa48fd3a6e68fb63
Deployer	0x47E6415Ed363dc1D26f339da4633D6eAff1e67cc
Block Number	13694965

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 were 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

CorePepe acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. CorePepe understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, CorePepe agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the CorePepe will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the CorePepe, the final report will be considered fully accepted by the CorePepe without the signature.

2. AUDIT RESULT

2.1. Overview

The PEPE Token was written in `Solidity` language, with the required version to be `0.8.4`.

The contract extends `Context` and `Ownable` from the `OpenZeppelin` library. Below table describes some properties of the audited PEPE Token (as of the report writing time).

PROPERTY	VALUE
Name	Pepe
Symbol	PEPE
Decimals	18
Total Supply	$420{,}000{,}000{,}000{,}000_{18}$ (it represents 420 trillion tokens)

Table 2. The PEPE Token properties

All the supply was minted to the owner (`corepepedev*core`) with the address `0x47E6415Ed363dc1D26f339da4633D6eAff1e67cc`. And then, the owner has burned 63% of the total supply to the address `0x...dead`.

2.2. Findings

During the audit process, the audit team found no vulnerability in the given version of PEPE Token, only some recommendations.

2.3. Additional notes and recommendations

2.3.1. INFORMATIVE - Unuse SafeMath in contracts

In the head of all files that in the audit scope, the contracts imported `SafeMath` library but it doesn't use inside the contract. In addtion, the SafeMath checking overflow is unnessary because solidity `0.8.0+` already do that by default.

RECOMMENDATION

We suggest removing this library for readability.

2.3.2. INFORMATIVE - Redundant `receive` function

This contract enables native token (CORE) receiving:

```
receive() payable external  {}
```

But there's no method to recover/rescue these funds.

RECOMMENDATION

We suggest removing the `receive` function.

APPENDIX

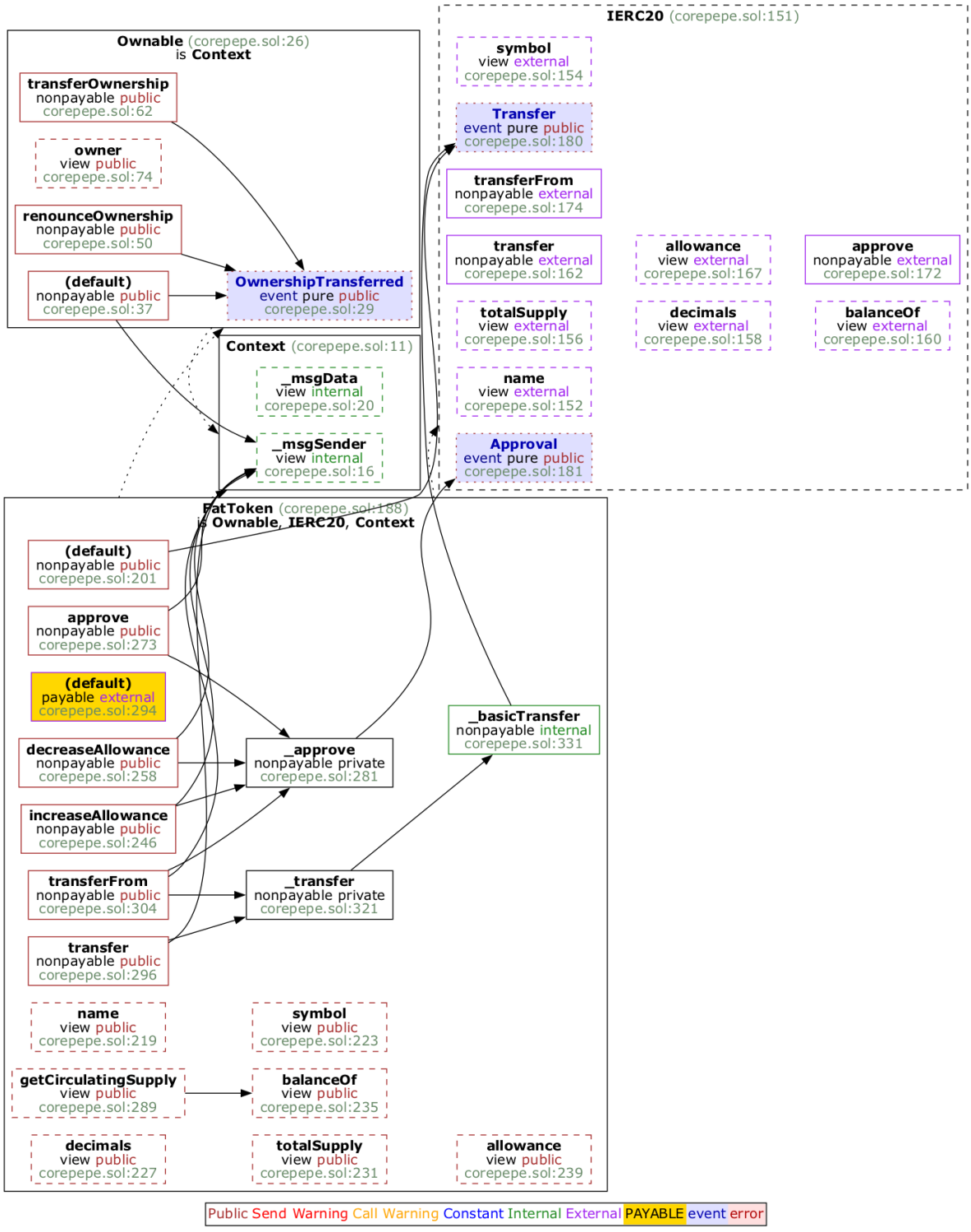

Image 1. ABI token smart contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jun 14, 2024*	Public Report	Verichains Lab

Table 3. Report versions history