DEV NECTAR

CHUTE Contract Audit Report

Table of Contents

DEV NECTAR

Table of Contents

Overview

PROJECT SUMMARY

DESCRIPTION	CHUTE CONTRACT
PLATFORM	ETHEREUM
LANGUAGE	SOLIDITY
CODEBASE	CHUTE.SOL (FILE SHARED BY CHUTE TEAM)
TOKEN ADDRESS	0XBF20FCA6430985B700B87FDC4422BC2C1659E8C9

AUDIT SUMMARY

DELIVERY DATE	10/21/2023
AUDIT METHODOLOGY	STATIC ANALYSIS, MANUAL REVIEW

Overview

Finding Summary

VULNERABILITY LEVEL	TOTAL	STATUS
CRITICAL	0	NOT FOUND
MAJOR	0	NOT FOUND
MINOR	01	AKNOWLEDGE
MEDIUM	01	AKNOWLEDGE
INFORMATIONAL	01	AKNOWLEDGE

PURPOSE OF THE AUDIT

This report has been prepared for CHUTE Contract to discover issues and vulnerabilities in the source code of the CHUTE Contract and any contract dependencies not part of an officially recognized library.

A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

1. **Ensure that the smart contract functions as intended.**
2. **Identify potential security issues with the smart contract.**
3. **Ensuring contract logic meets the specifications and intentions of the client.**

The information in this report should be used to understand the risk exposure of the smart contract and as a guide to improving the security posture of the smart contract by remediating the identified issues.

Audit Scope

The scope of this audit is to perform an in-depth analysis of the CHUTE Program's smart contract. The primary goal is to identify and inform CHUTE of any potential security vulnerabilities, logic issues, or inefficiencies in the contract code and its dependencies. The scope includes but is not limited to:

1. Review of the codebase as provided by the CHUTE Team.
2. Analysis of the contract's use of external libraries and dependencies.
3. Review of the contract's adherence to accepted Ethereum smart contract best practices.
4. Identification of any code areas that could result in unexpected or dangerous behaviour.
5. Verification that the contract behaves as intended by its specifications.

Audit Methodology

The audit will be conducted using a two-pronged approach of static analysis and manual review:

- Static Analysis:

  This involves exa the codebase for known security issues using automated tools. It helps in identifying common vulnerabilities quickly.

- Manual Review:

  This is a line-by-line inspection of the code, conducted by experienced smart contract auditors. It helps in identifying complex security issues, logic problems, inefficiencies, and code quality issues.

Each potential issue will be assigned a severity level from critical to information based on the potential impact on the contract operation. This audit report will contain a detailed description of each issue, its potential impact, and proposed solutions or mitigation strategies.

Please note that even a comprehensive security audit cannot guarantee 100% security, but it can significantly reduce the risk associated with smart contracts.

Smart Contract Overview:

General Information

The provided code is a smart contract written in Solidity for the Ethereum blockchain. It appears to be for a decentralized application (dApp) that facilitates staking in a process. The contract provides functionality for staking a particular token (CHUTE in this case), claiming rewards, distributing prizes, and withdrawing staked tokens. The contract handles the allocation and distribution of CHUTE tokens to miners who participate in the program.

Here are the key parts of the contract:

1. Inheritance and Libraries:
- **The contract inherits from Context, IERC20, and Ownable.**
- **It utilizes the SafeMath library for safe arithmetic operations.**

2. State Variables:
- **developmentWallet: The address for a wallet to receive funds.**
- **_name, _symbol, and _decimals: Metadata for the ERC-20 token.**
- **_totalSupply: Total supply of the token.**
- **_balances: A mapping to track token balances.**
- **_allowances: A mapping to manage allowances for token transfers.**
- **isExcludedFromFee and isMarketPair: Mappings to exclude certain addresses from fees and identify market pairs.**
- **swapThreshold: A threshold for triggering automatic token swaps.**
- **swapEnabled and swapbylimit: Flags for enabling token swapping and limiting swaps.**
- **buyTax and sellTax: Tax percentages for buys and sells.**
- **feedenominator: A denominator for fee calculations.**
- **dexRouter and dexPair: Addresses related to DEX routing and liquidity pairing.**
- **tradingEnable: A flag to enable trading.**
- **inSwap: A flag to indicate if a swap is in progress.**

General Information

3. Events:
- **SwapTokensForETH: Event for token swaps.**

4. Constructor:
- **Initializes contract variables and sets up the DEX router and liquidity pair.**
- **Excludes certain addresses from fees and sets allowances.**
- **Initializes the contract owner with the total token supply.**

5. View and Getter Functions:
- **Functions to retrieve metadata such as name, symbol, decimals, total supply, and balance of an account.**

6. Transfer and Approval Functions:
- **transfer and transferFrom for transferring tokens.**
- **approve, increaseAllowance, and decreaseAllowance for managing allowances.**
- **_transfer function for internal token transfers.**
- **shouldNotTakeFee and takeFee functions for calculating and deducting fees from transfers.**

7. Token Swapping and Liquidity Management:
- **swapBack and swapTokensForEth functions for swapping tokens to ETH.**
- **rescueFunds and rescueTokens for fund rescue.**
- **Functions to set buy/sell fees, exclude addresses from fees, set development wallet, and configure swap settings.**
- **enableTrading for enabling trading when conditions are met.**

This contract primarily handles the management of an ERC-20 token, including transferring, fee calculations, liquidity provisioning, and fund management. It also provides various configuration options for the contract owner.

General Information

- **Owner Functions:**

  There are several owner-only functions for managing the contract. They include changing the staking token, setting deposit limits, setting time intervals, withdrawing stuck tokens, and more.

- **View Functions:**

  These functions allow anyone to view the contract's state or a user's information.

Note: All interactions with the contract require the contract to be in a launched state. The contract owner has to manually start the contract using the launch function.

Contract Functionality

The contract primarily revolves around two major functionalities:

ERC20 Tokens:

- **The contract defines the token's name, symbol, and decimals.**
- **It initializes the total supply of the token (27,000,000,000 with 18 decimal places).**
- **The token implements the ERC-20 standard.**

Owner Privileges:

The contract extends the "Ownable" contract, allowing the owner to perform certain administrative functions.

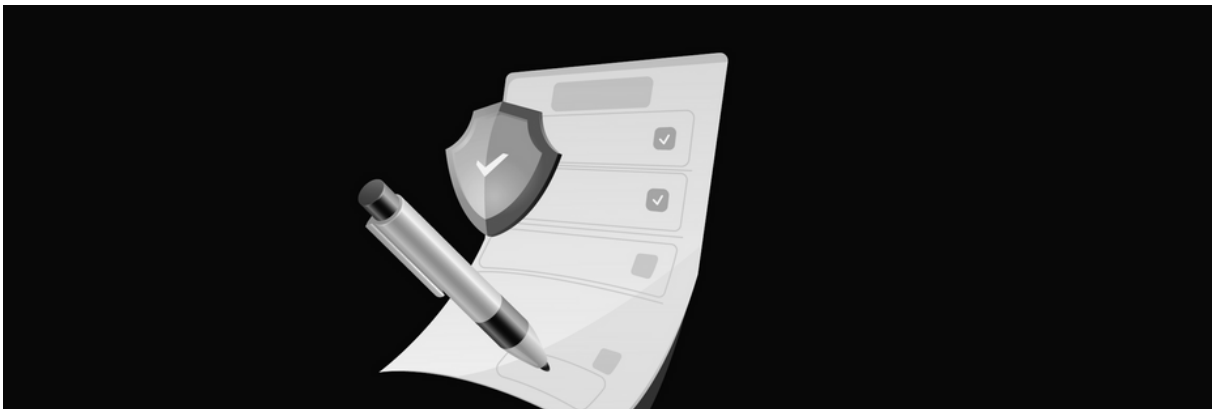
Transaction and Fee:

- **The contract applies transaction fees, where different fees are applied for buys and sells.**
- **Fees are calculated based on a percentage (buyTax for buys and sellTax for sells) of the transaction amount.**
- **A part of the fees is transferred to the contract itself.**

Swap Functionality:

- **The contract enables the automatic swapping of tokens for Ether (ETH) when certain conditions are met.**
- **Swap settings (swapEnabled and swapbylimit) can be controlled by the owner.**
- **There is a threshold (swapThreshold) that determines when swapping occurs.**

Architecture



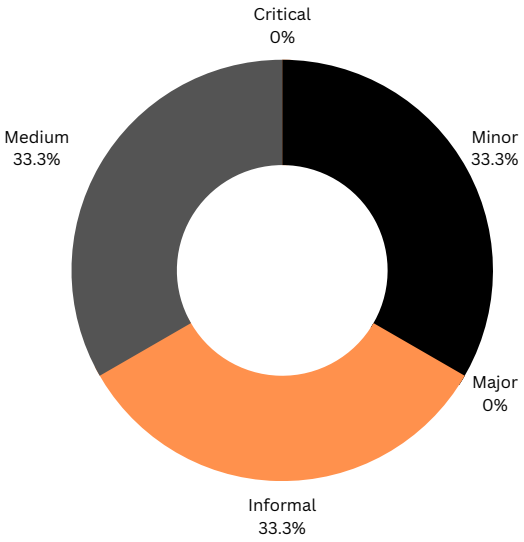
The architecture of the contract is fairly typical of Ethereum smart contracts. It employs a linear inheritance structure, with several base contracts providing fundamental functionality, and a final contract (the CHUTE Contract) that inherits from these base contracts.
The CHUTE Contract itself consists of several functions, events, and modifiers that handle everything from token minting to role management.

This contract primarily functions as an ERC-20 token with additional features for controlling fees, enabling/disabling swapping, and safeguarding emergency fund recovery. It is also designed to handle market pair conditions, which can influence fee calculations during transfers. The contract owner has significant control over various aspects, including fee settings, wallet addresses, and trading features.

AUDIT FINDINGS SUMMARY

This section details the vulnerabilities discovered during the audit, categorized by severity level.

- **Critical Findings - 0**
- **Major Finding - 0**
- **Minor Finding - 1**
- **Medium Finding - 1**
- **informational Finding - 1**



Critical Findings:

None

Major Findings:

None

AUDIT FINDINGS SUMMARY

Medium Findings

1. Logic Flaws in Token Distribution:

The CHUTE Contract has been found to grant full owner privileges. This is a major security risk as it could allow the owner to manipulate the contract, potentially leading to unauthorized actions or changes. This could disrupt the operations of the contract and jeopardize the assets contained within it.

Minor Findings

1. Use of latest solidity version:

The contract does not use the latest version of Solidity. While this is not a direct security issue, using an older version could expose the contract to vulnerabilities that have been fixed in later versions.

Informational Findings

1. Coding Style:

The coding style of the contract could be improved to enhance readability and maintainability. This is an informational finding and does not directly impact the security of the contract.

Risk Assessment

This section provides an evaluation of the identified issues in terms of their potential impact on the contract's operations, the likelihood of their occurrence, and the potential damage they might cause.

Critical Issue: Owner Privileges

The owner of the contract possesses extensive control over the contract's functionality, including the ability to halt the contract, add or remove miners, and make significant changes.

This centralized control point is vulnerable to exploitation if the owner's address is compromised, potentially leading to a halt in operations, the unlawful addition or removal of miners, and other forms of misconduct.

Recommendation

Implementing a multi-signature mechanism for critical decisions would reduce the risk associated with a single owner. This would require multiple trusted parties to agree before implementing crucial changes. In addition, consider limiting the owner's powers or building in a mechanism for transferring ownership under certain circumstances.

Medium Issue:
Logic Flaws in Token Distribution

The contract allocates tokens based on power. However, there are potential scenarios where this could lead to unfair distribution.

Recommendation

Implement checks and balances to ensure fair distribution of tokens. Monitor the token distribution regularly to detect and rectify any disparities early.

Conclusion

The audit of the CHUTE Contract has identified a variety of potential risks. These risks range from critical, such as the extensive control of the contract owner, to medium and minor issues related to contract dependency and logic flaws in token . It is imperative that these issues are addressed promptly to ensure the safe and fair operation of the contract.

While the presence of these risks indicates that there are areas for improvement in the contract's current design and operation, none of these issues are insurmountable. Implementing the recommended changes will greatly enhance the security and effectiveness of the contract, reducing the possibility of exploitation and misconduct.

Regular and thorough audits, such as this one, are crucial in the early detection and remediation of potential issues. They ensure the continued security and functionality of the contract, giving users and stakeholders confidence in the contract's operations.

It is also recommended that the contract design is revisited regularly to ensure alignment with the latest best practices in the rapidly evolving field of smart contract development.

DISCLAIMER

DEV NACTAR team conducts security assessments on the provided source code exclusively. In order to get a full view of our analysis, it is crucial for you to read the full report.

We have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions.

The audit documentation is for discussion purposes only.

Appendices

Appendix A: Severity Levels

The severity of the findings in this report is based on the risk posed by the issue of the functioning of the contract. Severity levels are categorized as follows:

Critical: The issue could result in a direct loss of funds or severely compromise the system's integrity.

Major: The issue could impact the main functionality of the system, leading to a disruption of service or a minor loss of funds.

Medium: The issue could lead to abnormal behavior under certain edge cases.

Minor: The issue does not pose a significant risk to the system but may lead to less-than-optimal contract behavior.

Informational: The issue does not pose a risk but is still something that could be improved or should be known.

Appendices

Appendix B: Methodology

This audit was performed using Static Analysis and Manual Review techniques. The audit aimed to ensure that the contract operates as intended, identify potential security issues within the contract and ensure that the contract logic meets the specifications and intentions of the client.

Appendix C: Disclaimer

This audit report is provided for informational purposes only. It is not an endorsement or disapproval of the CHUTE Contract. It should be used as a guide to understand the risk exposure of the smart contract and to improve the security posture by remediating the identified issues.

CHUTE
Audit

DEV NECTAR