verichains

SECURITY AUDIT OF

CATO TOKEN SMART CONTRACT

Public Report

Mar 29, 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 Mar 29, 2024. We would like to thank the CATO 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 Cato Token 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 some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY 5

1.1. About Cato Token Smart Contract 5

1.2. Audit scope 5

1.3. Audit methodology 5

1.4. Disclaimer 6

1.5. Acceptance Minute 6

2. AUDIT RESULT 7

2.1. Overview 7

2.2. Findings 8

2.2.1. CRITICAL - Incorrect implementation of the charging fee mechanism 9

2.2.2. HIGH - Lack of maximum total supply check 10

2.3. Additional notes and recommendations 11

2.3.1. INFORMATIVE - Missing zero check 11

2.3.2. INFORMATIVE - Unnecessary use of both `Ownable` and `AccessControl` 11

2.3.3. INFORMATIVE - Typo in the variable name `taxReceipient` 12

2.3.4. INFORMATIVE - Should `setTaxRecipient` and `setExcludeFromTax` called in the same function 12

3. VERSION HISTORY 13

1. MANAGEMENT SUMMARY

1.1. About Cato Token Smart Contract

CATO is an innovative platform designed to seamlessly combine gaming experiences with decentralized farming opportunities. At the core of this ecosystem lies the Cato Token Smart Contract token, an ERC20 token specifically crafted for CATO players to utilize within the game environment.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Cato Token Smart Contract. It was conducted on git repository link `https://github.com/CATOLottery/cato-interchain-token`.

The latest version of the following files were made available in the course of the review:

SHA256 Sum	File
3cf237a0fcd2dfa992057442cc29b6e4aa63db5024c63f146a28570975292089	./contracts/CatoToken.sol

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

CATO 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. CATO understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, CATO 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 CATO will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the CATO, the final report will be considered fully accepted by the CATO without the signature.

2. AUDIT RESULT

2.1. Overview

The Cato Token Smart Contract was written in `Solidity` language, with the required version to be `0.8.9`.

The contract extends `Ownable`, `ERC20` and `ERC20Burnable` . It also introduces a taxation mechanism where a certain percentage of token transfers is taxed and sent to a designated tax recipient. The contract allows for minting and burning tokens, with roles assigned for these actions. Furthermore, it includes features to exclude specific accounts from being taxed.

PROPERTY	VALUE
Name	Cato Token
Symbol	CATO
Decimals	18
Maximum Supply	222,000,222,444x10^{18} (It represents over 222 billion)

Table 2. The Cato Token Smart Contract properties

For the ERC20 token, the security audit team has the following checklist of centralization standards:

Checklist	Status
Fee modifiable	Yes
Mintable	No
Pausable	No
Trading cooldown	No
Blacklist	No

Table 3. The decentralization checklist for ERC20

2.2. Findings

During the audit process, the audit team identified some vulnerabilities in the provided version of Cato Token Smart Contract.

Issue	Severity	Status
Incorrect implementation of the charging fee mechanism	CRITICAL	Fixed
Lack of maximum total supply check	HIGH	Fixed
Missing zero check	INFORMATIVE	Acknowledged
Unnecessary use of both `Ownable` and `AccessControl`	INFORMATIVE	Fixed
Typo in the variable name `taxReceipient`	INFORMATIVE	Fixed
Should `setTaxRecipient` and `setExcludeFromTax` called in the same function	INFORMATIVE	Fixed

Table 4. The issue list

2.2.1. CRITICAL - Incorrect implementation of the charging fee mechanism

Position:

- CatoToken.sol#L66-76

Description:

When users transfer tokens, the contract charges a fee based on the `taxRate` percentage. The fee should be calculated based on the amount and deducted from the transfer amount as well. However, the code mistakenly charges an additional fee to users (`from` address) outside the transferred amount. This is a security vulnerability that can be exploited when leveraging the skim function of Uniswap v2.

```
function _beforeTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual override {
        super._beforeTokenTransfer(from, to, amount);
        if (taxRate > 0 && !excludedFromTax[from] && !excludedFromTax[to]) {

            uint256 tax = (amount * taxRate) / 1e20;
            _transfer(from, taxRecipient, tax);
            // Must deduct the tax from the transfer amount here

        }
        super._transfer(from, to, amount); // INCORRECT - No subtraction of the fee
    }
```

RECOMMENDATION

The code should be updated to calculate the fee based on the transferred amount and deduct it from the transfer amount. The audit team recommends overriding the `_transfer` function to handle the fee deduction correctly.

UPDATES

- ***Mar 29, 2024***: The issue has fixed by CATO team.

2.2.2. HIGH - Lack of maximum total supply check

Position:

- CatoToken.sol#L78-80
- CatoToken.sol#L82-84

Description:

The smart contract does not include a check to ensure that the total token supply does not exceed the predefined MAX_SUPPLY.

```
uint256 public constant MAX_SUPPLY = 222_000_222_444 * 10 ** 18;

    function mint(address to, uint256 amount) public onlyRole(MINTER_ROLE) {
        _mint(to, amount); //INCORRECT - No check for MAX_SUPPLY
    }
```

RECOMMENDATION

The code should be updated to include a check to ensure that the total token supply does not exceed the predefined MAX_SUPPLY.

UPDATES

- ***Mar 29, 2024***: The issue has fixed by removing the mint function.

2.3. Additional notes and recommendations

2.3.1. INFORMATIVE - Missing zero check

Positions

- CatoToken.sol#L22
- CatoToken.sol#L44
- CatoTokenDeployer.sol#L20
- CatoTokenDeployer.sol#L21

Description

A wrong user input or wallets defaulting to the zero addresses for a missing input can lead to the contract needing to redeploy or wasted gas. The code should be added zero-address check to these functions.

UPDATES

- ***Mar 29, 2024***: The issue has acknowledged by CATO team.

2.3.2. INFORMATIVE - Unnecessary use of both `Ownable` and `AccessControl`

Position:

- CatoToken.sol#CatoToken

Description:

The contract uses both the `Ownable` and `AccessControl` contracts to manage roles and permissions. The `Ownable` contract provides a single owner role, while the `AccessControl` contract allows for multiple roles and granular control over permissions. Using both contracts can lead to confusion and unnecessary complexity in managing roles and permissions.

Recommendation:

The audit team recommends using only the `AccessControl` contract to manage roles and permissions. The `Ownable` contract can be removed to simplify the contract structure and avoid confusion. Replace owner role with `DEFAULT_ADMIN_ROLE` in `AccessControl` contract.

UPDATES

- ***Mar 29, 2024***: The issue has fixed by CATO team.

2.3.3. INFORMATIVE - Typo in the variable name `taxReceipient`

Description

The variable name `taxReceipient` is misspelled and should be corrected to `taxRecipient`.

UPDATES

- ***Mar 29, 2024***: The issue has fixed by CATO team.

2.3.4. INFORMATIVE - Should `setTaxRecipient` and `setExcludeFromTax` called in the same function

Position:

- `CatoToken.sol`#setTaxRecipient()

Description:

The `setTaxRecipient` and `setExcludeFromTax` functions should be called in the same transaction to avoid potential issues with the tax recipient and excluded addresses.

UPDATES

- ***Mar 29, 2024***: The issue has fixed by CATO team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 27, 2024*	Public Report	Verichains Lab
1.1	*Mar 29, 2024*	Public Report	Verichains Lab

Table 5. Report versions history