EQUINOX BUSINESS

Smart Contract Audit Report

April, 2022

Secureverse

Table of Content

Summary

Project Name	Equinox Business Token Contract
Overview	The EQX Coin is a Fin-tech Infrastructure for Teams, DAOs, NGOs, Guilds and Communities for borderless governance and financial operations.
Timeline	April 21, 2022 - April 27, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse Equinox smart contract's code-base for quality, security, and correctness.
Source Code	https://github.com/equinoxbusiness/tokencontract/blob/main/EquinoxToken.sol

	High	Medium	Low	Informational
Open Issues	0	0	1	1
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Transaction-Ordering Dependence
- ✓ Use of tx.origin
- ✓ Exception disorder
- ✓ DoS with Block Gas Limit
- ✓ Gas Limit and Loops
- ✓ Timestamp Dependence
- ✓ Unchecked math
- ✓ Malicious libraries
- ✓ Unsafe type inference
- ✓ Redundant fallback function
- ✓ Unchecked external call
- ✓ Style guide violation
- ✓ Compiler version not fixed
- ✓ Send instead of transfer
- ✓ ERC20 API violation
- ✓ Implicit visibility leave

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of BEP-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the white paper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - EquinoxToken.sol

High Severity Issues

No issues were found

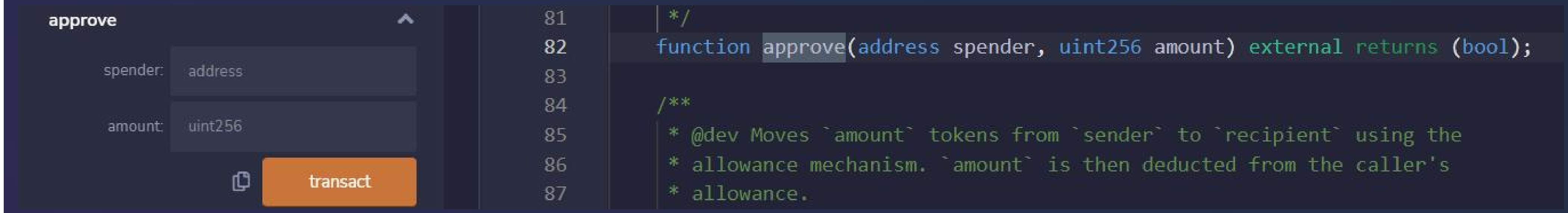
Medium Severity Issues

No issues were found

Low Severity Issues

- Type: Front Running/Race Condition
- Description: The ERC20 token standard includes a function called 'approve' which allows an address to approve another address to spend tokens on their behalf. The miner who solves the block also chooses which transactions from the pool will be included in the block, typically ordered by the gasPrice of each transaction.
- An attacker can watch the transaction pool for transactions that may contain solutions to problems, and modify or revoke the solver’s permissions or change state in a contract detrimentally to the solver. The attacker can then get the data from this transaction and create a transaction of their own with a higher gasPrice so their transaction is included in a block before the original.
  For more details, refer - https://rb.gy/q2v3ob
- Remediation: Away to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks, and to accounts owned by the people you may trust. Although there are multiple workarounds for this vulnerability. For more details visit the link below,
  https://rb.gy/xyxshw

Status: Open

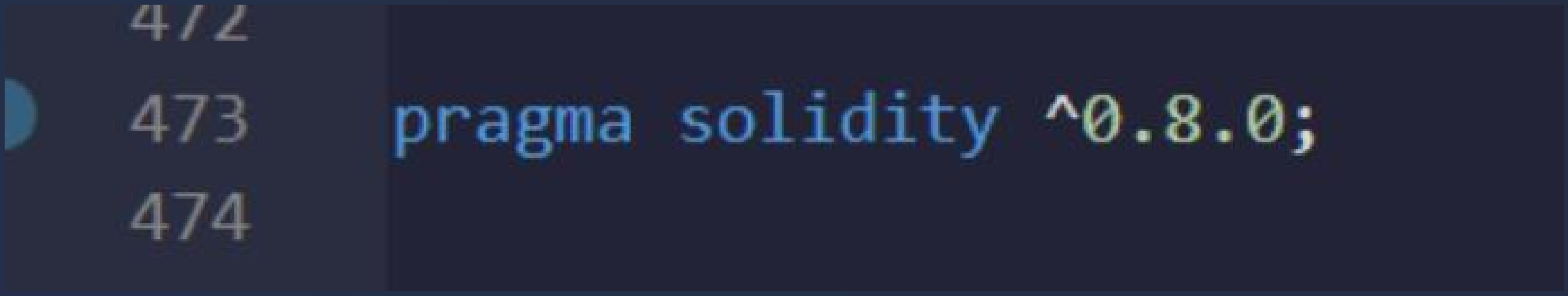


```
    */
    function approve(address spender, uint256 amount) external returns (bool);

    /**
     * @dev Moves `amount` tokens from `sender` to `recipient` using the
     * allowance mechanism. `amount` is then deducted from the caller's
     * allowance.
```

Informational Issues

- Type: Unlocked Pragma
- Description: Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.
- Remediation:Here all the in-scope contracts have an unlocked pragma, it is recommended to lock the same. Moreover, we strongly suggest not to use experimental Solidity features (e.g., pragma experimental ABIEncoderV2) or third-party unaudited libraries. If necessary, refactor the current code base to only use stable features.
- Status: Open



```
pragma solidity ^0.8.0;
```

Functional Testing

Some of the tests performed are mentioned below

Should mint initial balance to msg.sender on deployment
Should be able to mint tokens only by contract owner
Should update balances of sender and recipient when token transferred.
Should be able to approve tokens
Should be able to increase allowance
Should be able to decrease allowance
Should be able to spend approved tokens
Should be able to burn approved tokens
Reverts while minting if totalsupply + amount to mint exceeds the cap limit
Reverts while minting if _mintingFinished is true
Reverts on transfer to zero address
Reverts on approve to zero address
Reverts if sender doesn’t holds enough token balance for sending
Reverts if spender doesn’t holds enough approval to spend someone’s tokens

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the Equinox Business Smart Contract. We performedour audit according to the procedure described above.

Disclaimer

Secureverse smart contract audit is not a security warranty, investment advice, or an endorsement of the Equinox Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Equinox Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About Secureverse

We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

SECUREVERSE