ALPHA APE LLC

Equilibrium BSC EQLM

Audit Report
May.22.2022

Table of Contents

SUMMARY OF AUDIT

DETAILS OF AUDITED PROJECT

Audited Project:	Equilibrium BSC - EQLM
Source Code:	https://bscscan.com/address/0xe45e041b2fb97c932285d59c1f6e20a102fc9c6c#code
Solidity File:	equilibrium.sol
Security Audit Date:	May. 22 - 2022
Revisions:	None
Auditing Methods:	Automatic review + Manual review

AUDITING METHODS AND COVERING SECTORS

Evaluation objective for the security audit:

- Quality of smart contracts code
- Issues and vulnerabilities with security
- Documentation, project specifics and commenting on smart contract
- Correctness of specifications regarding the use-case
- Following the best practices on smart contract

Audit covers these sectors of smart contract for possible vulnerabilities, issues and recommendations for better practices in case of severe or medium issues:

- Dependence Transaction Order
- Single and Cross-Function Reentrancy
- Time Dependency
- Integer Overflow
- Integer Underflow
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Number rounding errors
- Insufficient gas issues
- Logical oversights
- Access control
- Centralization of power
- Logic-Specification
- Contradiction
- Functionality duplication

- Malicious contract behaviour and abusable functions
- Possible DoS vulnerabilities

The code review conducted for this audit follows the following structure:

1. Review of the specifications, documentation and commenting provided by the project owners regarding the functionality of the smart contract
2. Automated analysis of the smart contract followed by manual, line-by-line analysis of the smart contract
3. Assessment of smart contracts correctness regarding the documentation and commenting compared to functionality
4. Assessment of following the best practices
5. Recommendations for better practises in case severe or medium vulnerabilities

SMART CONTRACT DETAILS

Contract ID	0xE45E041B2Fb97c932285D59c1F6e20A102fC9C6c
Blockchain:	Binance Smart Chain
Language Used:	Solidity
Compiler Version:	v0.8.13+commit.abaa5c0e
Bscscan Verification:	2022-05-18
Type of Smart Contract:	BEP20 Token | Dividend Paying Token
Libraries Used:	IterableMapping
Optimization Enabled:	Yes with 200 runs

SUMMARY OF AUDIT RESULTS

Alpha Ape LLC smart contract audit for Equilibrium BSC - EQLM is marked as **PASSED** result without severe issues on the logic and functions of the contract. Review of the documentation and description of the projects use-case and the smart contract follows the line of good practices. Fees of the contract can be adjusted by the owner, but the auditor does not have concerns about the fee structure as it has been stated that the contract is to be renounced after a 30 days vesting period ends for private investors.

SEVERITY OF RISKS AND VULNERABILITIES

LOW-SEVERITY	MEDIUM-SEVERITY	HIGH-SEVERITY
3	**0**	**0**

REPORTED VULNERABILITIES AND ISSUES

LEVEL OF SEVERITY	DESCRIPTION	FILE	CODELINES AFFECTED
LOW-SEVERITY	Use of "tx.origin" as a part of authorization control.	equilibrium.sol	L: 1273 C: 80
LOW-SEVERITY	Use of "tx.origin" as a part of authorization control.	equilibrium.sol	L: 1377 C: 84
LOW-SEVERITY	Potential use of "block.number" as source of randonmness.	equilibrium.sol	L: 1309 C: 55

FINDINGS IN-DEPTH

LOW-SEVERITY USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL

```
1271  function processRewardsPool(uint256 gas) external {
1272  (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) = rewardsPool.process(gas);
1273  emit ProcessedRewardsPool(iterations, claims, lastProcessedIndex, false, gas, tx.origin);
1274  }
1275
```

Analysis: tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract.

Possible vulnerability: A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

Recommendation: tx.origin should not be used for authorization. Use msg.sender instead. In this case while only processing earned dividends in rewards, the auditor does not have concerns of using tx.origin instead msg.sender.

equilibrium.sol L: 1273 C: 80

LOW-SEVERITY USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL

```
1375
1376 try rewardsPool.process(gas) returns (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) {
1377 emit ProcessedRewardsPool(iterations, claims, lastProcessedIndex, true, gas, tx.origin);
1378 }
1379 catch {}
```

Analysis: tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract.

Possible vulnerability: A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

Recommendation: tx.origin should not be used for authorization. Use msg.sender instead. In this case while only processing earned dividends in rewards, the auditor does not have concerns of using tx.origin instead msg.sender.

equilibrium.sol L: 1377 C: 84

LOW-SEVERITY POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDONMNESS

```
1307 }
1308
1309 if(!tradingActive || tradingActiveBlock + 2 >= block.number){
1310 require(_isExcludedFromFees[from] || _isExcludedFromFees[to], "Trading is always active after deployment");
1311 }
```

Analysis: The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner

Possible vulnerability: For example, use of block.timestamp is insecure, as a miner can choose to provide any timestamp within a few seconds and still get his block accepted by others. Use of blockhash, block.difficulty and other fields is also insecure, as they're controlled by the miner. If the stakes are high, the miner can mine lots of blocks in a short time by renting hardware, pick the block that has required block hash for him to win, and drop all others.

Recommendation: If the contract was to be an gambling app or similar guess to win app, we would recommend not to use block.timestamp variables as it could be manipulated by malicious miners. Auditor does not see problem using this variable as it is called once during the deployment and only used as an indicator at a later time when the trading has been started.

equilibrium.sol L: 1309 C: 55

DOCUMENTATION AND COMMENTING

The code has a decent amount of comments and projects description and interview has been fluid. Projects use-case and purpose of the contract follows the line as described and no concerns rise during the whole process of interview of the project and the use of the contract. Even the fees of the contract are adjustable, the auditor has no concern of abuse in the fees structure, and the contract is to be renounced after a 30 days private investors wallet vesting period ends. Functions of the contract are understandable and during the auditing time while the project has been launched, nothing out of ordinary has been detected on the contracts use.

CORRECTNESS OF SPECIFICATIONS

Smart contract follows the functionality that is stated in the documentation and description of the contract. Use-case of the contract interacting as an community token, has taken its place just as described by the managers of the project. Projects management team has provided an KYC verification with an trusted security company. All specifications stated have been verified as stated and no concerns rise from the auditors side.

FOLLOWING THE BEST PRACTICES

The smart contract follows the best practices in majority. Minor parts that do not follow the best practices, or is affected by possible vulnerabilities, do not rise fear in malicious use of the contract or severe issues in projects use-case. Low-level severities that are noted in the audit are only improper adherence to coding standards, than security vulnerabilities.

HISTORY OF REVISIONS, FUNCTIONS AND VARIABLES

HISTORY OF REVISIONS

Initial audit was performed May. 22-2022. No further revision history.

Team has been provided recommendations for better practices for revision. There is no severe or medium notifications or recommendations for the revision. Contract is deployed and immutable and the possible vulnerabilities has been stated in the security audit. Project is stated to be community driven and security auditor has validated the use-case of the contract to follow its description the best possible way. Projects managers have been interviewed before and during the audit, and the auditor does not have concerns of malicious use of fees or any other functionality of the contract.

Projects management has provided the auditor a list of the vested wallets and the auditor has verified the list to be correct and correctly vested.

Throughout the process of contract creation, audit process and general interaction between the auditor and the projects management, the management has shown capability of leadership and fairness regarding the projects future vision and plans.

KYC verification by the projects management team has been provided and fully verified by the auditor. This brings a second stage of trust and legitimacy. KYC information will be held for 6 months in secure vault by a trusted global security company.