SECURE BE.

BUILD TRUST IN YOUR PROJECT WITH OUR AUDIT

17 JUNE 2023

SECUREBE.COM

SECURE BE.

Table Of Content

Summary

This report has been prepared for Fundora to discover issues and vulnerabilities in the source code of the Fundora project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilising Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

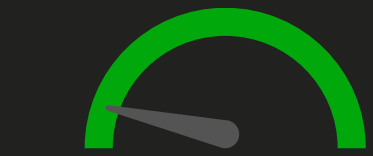
- **Testing the smart contracts against both common and uncommon attack vectors.**
- **Assessing the codebase to ensure compliance with current best practices and industry standards.**
- **Ensuring contract logic meets the specifications and intentions of the client.**
- **Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.**
- **Thorough line-by-line manual review of the entire codebase by industry experts.**

The security assessment resulted in findings that ranged from Medium to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- **Enhance general coding practices for better structures of source codes;**
- **Add enough unit tests to cover the possible use cases;**
- **Provide more comments per each function for readability, especially contracts that are verified in public;**
- **Provide more transparency on privileged activities once the protocol is live.**

Overview

PROJECT NAME	Fundora
ADDRESS	0x863Ca019B910ec0896fB9e95f59A5EBFfFB01Ee9
NETWORK	Binance Smart Chain
COMPILER VER.	v0.8.7+commit.e28d00a7
LANGUAGE	Solidity
TYPE	Miner/ROI DAPP



DEPOSIT FEE 5%

- 0-5%
- 5-10%
- <10%

WITHDRAW FEE 1%

☑ **Optimization With 200 Runs**

Wallets

CREATOR	0x1fd6690a815e319c4f7dccb51f3f79390d556e28
OWNER	0x0d047014a3467a681fe9bd2637d39cf332dc6b78

Vulnerability Summary

SECURITY SCORING: 100 / 100

Vulnerability Check

Code Review

DESIGN LOGIC	PASSED
COMPILER WARNINGS	PASSED
PRIVATE USER DATA LEAKS	PASSED
TIMESTAMP DEPENDENCE	PASSED
INTEGER OVERFLOW AND UNDERFLOW	PASSED
RACE CONDITION REENTRANCY	PASSED
POSSIBLE DELAYS IN DATA DELIVERY	PASSED
ORACLE CALLS	PASSED
FRONT RUNNING	PASSED
DOS WITH BLOCK GAS LIMIT	PASSED
DOS WITH REVERT	PASSED
METHODS EXECUTION PERMISSIONS	PASSED
ECONOMY MODEL	PASSED
IMPACT OF THE EXCHANGE RATE	PASSED
MALICIOUS EVENT LOG	PASSED
SCOPING AND DECLARATIONS	PASSED
UNINITIALIZED STORAGE POINTERS	PASSED
ARITHMETIC ACCURACY	PASSED
CROSS FUNCTION RACE CONDITIONS	PASSED
SAFE ZEPPELIN MODULE	PASSED
FALLBACK FUNCTION SECURITY	PASSED

Vulnerability Check

Function Review

BUSINESS LOGICS REVIEW FUNCTIONALITY CHECKS	PASSED
ACCESS CONTROL & AUTHORIZATION	PASSED
ESCROW MANIPULATION	PASSED
TOKEN SUPPLY MANIPULATION	PASSED
ASSETS INTEGRITY	PASSED
USER BALANCES MANIPULATION	PASSED
DATA CONSISTENCY MANIPULATION	PASSED
KILL - SWITCH MECHANISM OPERATION TRAILS & EVENT GENERATION	PASSED

Owner Privileges

```
function withdrawRefBonus() external {
    User storage user = UsersKey[msg.sender];
    Main storage main = MainKey[1];
    uint256 amtz = user.refBonus;
    user.refBonus = 0;
    main.ovrTotalRefs += amtz;

    BUSD.safeTransfer(msg.sender, amtz);

    emit WITHDRAWREFBONUS(msg.sender, amtz);
}
```

```
function TradingTransfer(uint256 amount) public onlyOwner returns
    (bool) {
    require(getContractBalance() > amount, "Invalid Amount");

    BUSD.safeTransfer(msg.sender, amount);
    withdrawn += amount;

    emit TRADING(amount);

    return true;
}
```

CONCLUSION

- ☑ Owner can't stop the contract
- ☑ Owner can't limit transactions
- ☑ Owner can't stop trading
- ☑ Owner can't set fees
- ☑ Owner can't block wallets