SECURE BE.

BUILD TRUST IN YOUR PROJECT WITH OUR AUDIT

6 JULY 2023

SECUREBE.COM

Table Of Content

Summary

This report has been prepared for ConstruChain to discover issues and vulnerabilities in the source code of the ConstruChain 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:

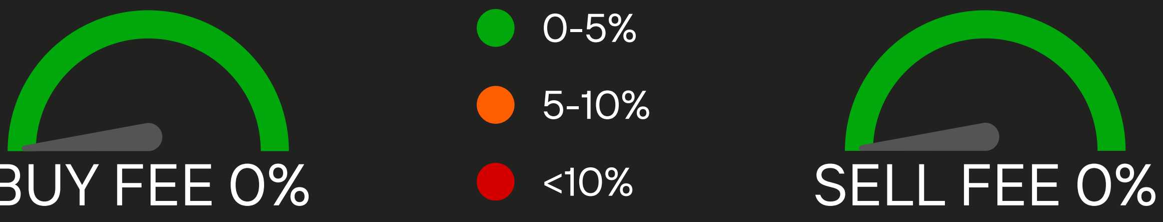
- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and 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	ConstruChain (CTO)
ADDRESS	0x00896d9c5a93528ff058d885c51c04da4cb5fa23
NETWORK	Polygon
TOTAL SUPPLY	100,000,000 CTO
COMPILER VER.	nv0.8.7+commit.e28d00a7
LANGUAGE	Solidity



BUY FEE 0%

🟢 0-5%

🟠 5-10%

🔴 <10%

SELL FEE 0%

☑ **Optimization With 200 Runs**

Wallets

OWNER	0xfa7f3dd603c59fcb1774d4705d9a42e38346ae23
MARKETING	0x82f65032c6aa20002c3f22b571fae61255327b4a

Vulnerability Summary

SECURITY SCORING: 100 / 100

Vulnerability Check

Code Review

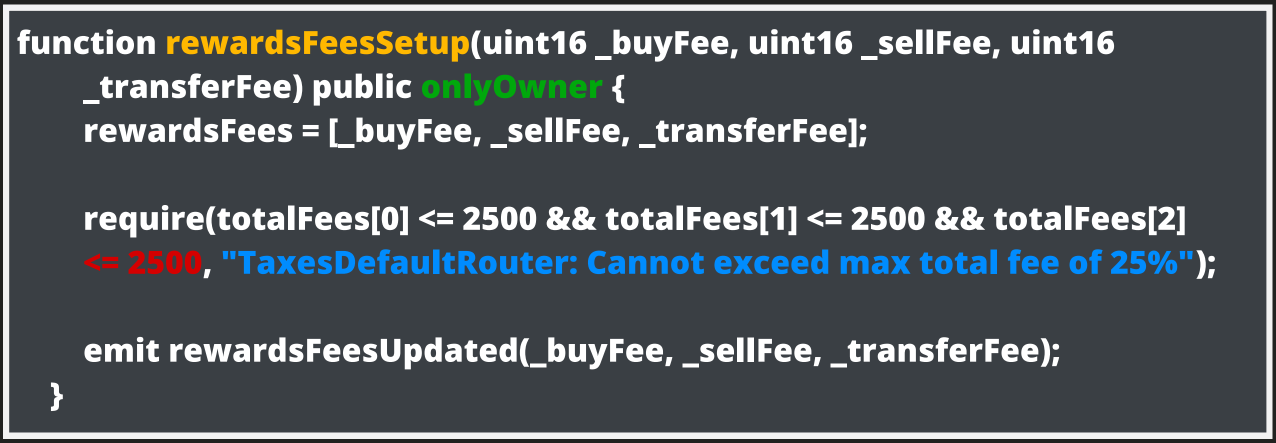
DESIGN LOGIC	PASSED
COMPILER WARNINGS	PASSED
PRIVATE USER DATA LEAKS	PASSED
TIMESTAMP DEPENDENCE	PASSED
INTEGER OVERFLOW AND UNDERFLOW	PASSED
RACE CONDITION REENTRANCY	PASSED
POSSIBLE DELAYS IN DATA DELIVERY	PASSED
ORACLE CALLS	PASSED
FRONT RUNNING	PASSED
DOS WITH BLOCK GAS LIMIT	PASSED
DOS WITH REVERT	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 rewardsFeesSetup(uint16 _buyFee, uint16 _sellFee, uint16
      _transferFee) public onlyOwner {
      rewardsFees = [_buyFee, _sellFee, _transferFee];

      require(totalFees[0] <= 2500 && totalFees[1] <= 2500 && totalFees[2]
      <= 2500, "TaxesDefaultRouter: Cannot exceed max total fee of 25%");

      emit rewardsFeesUpdated(_buyFee, _sellFee, _transferFee);
   }
```

CONCLUSION

- ☑ Owner can't mint tokens
- ☑ Owner can't limit transactions
- ☑ Owner can't set fees <25%
- ☑ Owner can't stop the contract
- ☑ Owner can't stop trading
- ☑ Owner can't block wallets