SMART CONTRACT AUDIT

ZOKYO.

Feb 8, 2021 | v. 2.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges

TECHNICAL SUMMARY

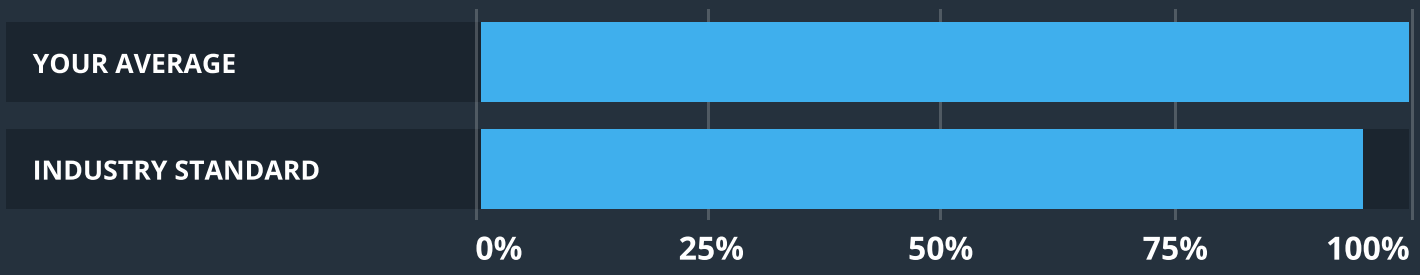
This document outlines the overall security of the xBTC smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the xBTC smart contract codebase for quality, security, and correctness.

Contract Status

There was one critical issue found during the audit, which was successfully resolved.

Testable Code



Testable code is 100%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the xBTC team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Disclaimer. The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

Overview of the audit . The project has 1 flattened file. It contains approx 672 lines of Solidity code. All the functions and state variables are well commented, but that does not create any vulnerability.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from:

- Repository: https://github.com/ipo/wrapped-xBTC;
- Commit: c778273d5dcf6aa77b2a024e5f53b9a6526d3314.

Requirements:
tech spec overview – https://github.com/ipo/wrapped-xBTC/blob/master/LITEPAPER.md.

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Bondly smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

SUMMARY

The contract is well-designed & follows all known best practices & standards. The smart contract code itself is well written and has good quality.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

HIGH | RESOLVED

Method `deposit` contains `payable` modifier.

Recommendation:
Remove modifier `payable` as method is not intended to receive Ethers.

MEDIUM | RESOLVED

Methods rawToWrapAmount, wrapToRawAmount & deposit are not using SafeMath for all arithmetic operations.

Recommendation:
Use SafeMath for all arithmetic operations.

Vulnerabilities Checklist

	WrappedXBTC
Re-entrancy	Not affected
Access Management Hierarchy	Not affected
Arithmetic Over/Under Flows	Not affected
Unexpected Ether	Not affected
Delegatecall	Not affected
Default Public Visibility	Not affected
Hidden Malicious Code	Not affected
Entropy Illusion (Lack of Randomness)	Not affected
External Contract Referencing	Not affected
Short Address/ Parameter Attack	Not affected
Unchecked CALL Return Values	Not affected
Race Conditions / Front Running	Not affected
General Denial Of Service (DOS)	Not affected
Uninitialized Storage Pointers	Not affected
Floating Points and Precision	Not affected
Tx.Origin Authentication	Not affected
Signatures Replay	Not affected
Pool Asset Security (backdoors in the underlying ERC-20)	Not affected

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by xBTC team (original test coverage)

Basic functionality

when brand new

- ✓ original xBTC uFragments should have the correct symbol (40ms)
- ✓ original xBTC uFragments should have the correct name (93ms)
- ✓ original xBTC uFragments should have the correct decimals
- ✓ wrap should have the correct symbol
- ✓ wrap should have the correct name
- ✓ wrap should have the correct decimals
- ✓ wrapped token should be available
- ✓ 0 deposit should work without effects (134ms)
- ✓ 0 withdraw should work without effects (123ms)
- ✓ shouldn't allow user to deposit more than they have (236ms)
- ✓ shouldn't allow eth to be sent to the contract (66ms)

with swaps from multiple users

- ✓ should swap back correctly (574ms)
- ✓ should swap back correctly in jumbled order (587ms)

with a swap done

- ✓ should swap back correctly (105ms)
- ✓ should round down when swapping back (107ms)
- ✓ shouldn't allow user to withdraw more than they have (59ms)
- ✓ should allow call to rawToWrapAmount
- ✓ should allow call to wrapToRawAmount (51ms)
- ✓ should allow a user other than the depositer to withdraw (279ms)

after a positive rebase

- ✓ should withdraw the correct amount (131ms)
- ✓ should withdraw the correct amount (with another depositer) (699ms)

after a negative rebase

- ✓ should withdraw the correct amount (172ms)
- ✓ should withdraw the correct amount (with another depositer) (795ms)

23 passing (20s)

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	100.00	70.00	100.00	100.00	
iDetailedERC20.sol	100.00	100.00	100.00	100.00	
WrappedXBTC.sol	100.00	70.00	100.00	100.00	
All files	**100.00**	**70.00**	**100.00**	**100.00**	

Tests written by Zokyo Secured team

As part of our work assisting xBTC in verifying the correctness of their contract code, our team was responsible for writing additional tests using the Truffle testing framework.

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

```
Contract: WrappedXBTC
  original details
    ✓ should have correct name (53ms)
    ✓ should have correct symbol
    ✓ should have correct decimals
  wrap details
    ✓ should have correct name
    ✓ should have correct symbol
    ✓ should have correct decimals
    ✓ should return correct wrapped token address
  wrong transfers
    ✓ user balances shouldn't be changed after 0 deposit by 0 amount (111ms)
    ✓ withdraw by 0 amount should work correct (118ms)
    ✓ shouldn't deposit when amout more than user balance (157ms)
  correct transfers
    ✓ rawToWrapAmount should work correctly
    ✓ should deposit correctly (242ms)
    ✓ should swap back correctly by parts (352ms)
  with swaps from multiple users
    ✓ wrap the raw tokens for multi users (1206ms)
    ✓ should swap back correctly for multi users (1166ms)
  afret rebasing
    ✓ should deposit correctly (206ms)
    ✓ withdraw should work correctly (303ms)

17 passing (12s)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	100.00	70.00	100.00	100.00	
iDetailedERC20.sol	100.00	100.00	100.00	100.00	
WrappedXBTC.sol	100.00	70.00	100.00	100.00	
All files	**100.00**	**70.00**	**100.00**	**100.00**	

We are grateful to have been given the opportunity to work with the xBTC team.

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.

Zokyo's Security Team recommends that the xBTC team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.