SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: RedFox

Date: November 15th, 2021

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed — upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for RedFox.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Bridge
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/RFL-Valt/eth-bridge-contract
Commit	61d530d50229e57f73434eae421abd23cb32685c
Deployed contract	https://etherscan.io/address/0xf60ab4a139fca6bb30b20c5ac5bbbff64bf1b080#code
Technical Documentation	NO
JS tests	YES
Website	Redfoxlabs.io
Timeline	22 NOVEMBER 2021 - 15 DECEMBER 2021
Changelog	24 NOVEMBER 2021 - Initial Audit 02 DECEMBER 2021 - Second Review 15 DECEMBER 2021 - Third Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by RedFox (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between November 22nd, 2021 - November 24th, 2021.

Second review conducted on December 2nd, 2021.

Third review conducted on December 15th, 2021.

Scope

The scope of the project is smart contracts in the repository:

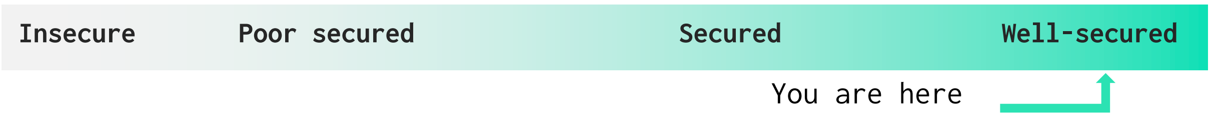
Repository:
https://github.com/RFL-Valt/eth-bridge-contract
Commit:
61d530d50229e57f73434eae421abd23cb32685c
Deployed contract:
https://etherscan.io/address/0xf60ab4a139fca6bb30b20c5ac5bbbff64bf1b080#code
Technical Documentation: No
JS tests: Yes (included in the repo: “/test/”)
Contracts:
ECDSA.sol
test/Token.sol
ETHWAXBRIDGE.sol
interfaces/ERC20Interface.sol
Owned.sol
libraries/Math.sol
libraries/Endian.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.



Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** medium and **7** low severity issues.

After the second review security engineers found that all issues were addressed.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

Audit overview

🟥 🟥 🟥 🟥 Critical

No critical issues were found.

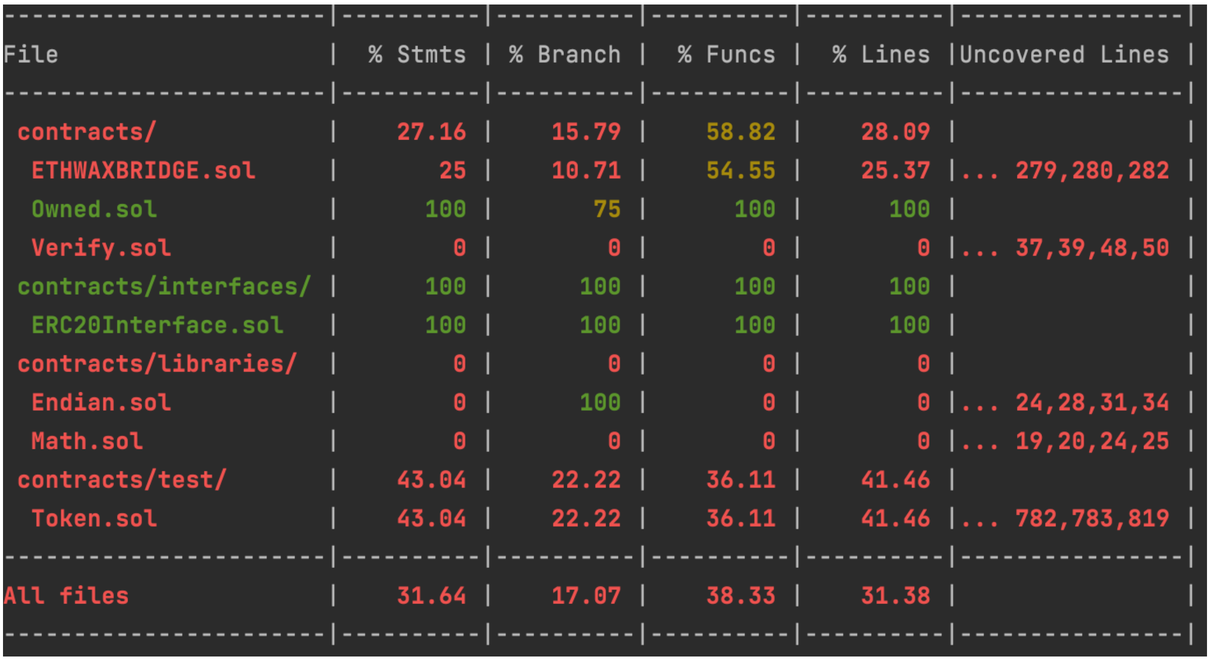
🟨 🟨 🟨 High

No high severity issues were found.

🟩 🟩 Medium

Very low tests coverage.

The overall test coverage is about 31% for statements and 17% for code branches which is too low. Many aspects of the code and logic branches remain untested.



```
-----------------------|----------|----------|----------|----------|----------------|
File                   |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-----------------------|----------|----------|----------|----------|----------------|
 contracts/            |    27.16 |    15.79 |    58.82 |    28.09 |                |
  ETHWAXBRIDGE.sol     |       25 |    10.71 |    54.55 |    25.37 |... 279,280,282 |
  Owned.sol            |      100 |       75 |      100 |      100 |                |
  Verify.sol           |        0 |        0 |        0 |        0 |... 37,39,48,50 |
 contracts/interfaces/ |      100 |      100 |      100 |      100 |                |
  ERC20Interface.sol   |      100 |      100 |      100 |      100 |                |
 contracts/libraries/  |        0 |        0 |        0 |        0 |                |
  Endian.sol           |        0 |      100 |        0 |        0 |... 24,28,31,34 |
  Math.sol             |        0 |        0 |        0 |        0 |... 19,20,24,25 |
 contracts/test/       |    43.04 |    22.22 |    36.11 |    41.46 |                |
  Token.sol            |    43.04 |    22.22 |    36.11 |    41.46 |... 782,783,819 |
-----------------------|----------|----------|----------|----------|----------------|
All files              |    31.64 |    17.07 |    38.33 |    31.38 |                |
-----------------------|----------|----------|----------|----------|----------------|
```

Recommendation: Please write more tests and ensure that you have covered the code for at least 95%.

Status: Fixed. Test coverage is 100%

🟦 Low

1. Inconsistency of naming and declaring.

   The contract’s name is set to “ERC20Interface” while it is not declared as an interface but an abstract contract.

Contract: ERC20Interface.sol

Recommendation: Fix inconsistency in the naming.

Status: Fixed

2. The imported contract was never used.

   While ETHWAXBRIDGE imports ERC20Interface but it never uses anything declared there.

Contract: ERC20Interface.sol

Recommendation: Remove excess import statement.

Status: Fixed

3. Missing events.

   updateThreshold has no event so it is hard to track these changes off-chain.

Contract: ETHWAXBRIDGE.sol

Function: updateThreshold

Recommendation: Please emit an event on threshold change.

Status: Fixed

4. Boolean equality.

   Boolean constants can be used directly and do not need to be compared to **true** or **false**.

Contract: Oracled.sol

Functions: unregOracle, onlyOracle

Recommendation: Remove the equality to the boolean constant.

Status: Fixed

5. A public function that could be declared external.

   public functions that are never called by the contract should be declared **external** to save gas.

Contract: ETHWAXBRIDGE.sol

Functions: receiveApproval, regOracle, unregOracle, bridge, claim, updateThreshold, transferAnyERC20Token

Recommendation: Use the **external** attribute for functions never called from the contract.

Status: Fixed

6. A public function that could be declared external.

public functions that are never called by the contract should be declared **external** to save gas.

Contract: Owned.sol

Functions: transferOwnership, acceptOwnership

Recommendation: Use the **external** attribute for functions never called from the contract.

Status: Fixed

7. A public function that could be declared external.

   public functions that are never called by the contract should be declared **external** to save gas.

Contract: ERC20Interface.sol

Functions: totalSupply, balanceOf, allowance, transfer, approve, transferFrom

Recommendation: Use the **external** attribute for functions never called from the contract.

Status: Fixed

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **1** medium and **7** low severity issues.

After the second review security engineers found that all issues were addressed.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.