SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Allbridge
Date: Feb 09, 2023

Document

Name	Smart Contract Code Review and Security Analysis Report for Allbridge
Approved By	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU
Type	Cross-chain, Bridge
Platform	Near
Language	Rust
Methodology	Link
Website	https://allbridge.io/
Changelog	31.01.2023 - Initial Review 09.02.2023 - Second Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Allbridge (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 contracts.

Scope

The scope of the project is review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/allbridge-io/near-stellar-state-contract
Commit	4cd67d596
Whitepaper	https://docs.allbridge.io/
Functional Requirements	https://docs.allbridge.io/
Technical Requirements	https://github.com/allbridge-io/near-stellar-state-contract
Contracts	File: ./contract/src/lib.rs SHA3:d2c1d792b23c0e38d3555dfb867b85e827a6461bc714b88fdd2e014734d89641 File: ./contract/src/types.rs SHA3:92c4986b727fc07d1cc846c4d52b48f92a76aad26baad72bcaa001fdff9fd938 File: ./contract/src/utils.rs SHA3:6625f1cbef6932fe8da8c12b31877bc6c988acfc368b624ccfafa54e635912b3 File: ./contract/src/validator.rs SHA3:10632891c7be10dc6964256e4b47452898b43af132380192223b0dafc47ddede File: ./contract/src/contract_core/create_transfer.rs SHA3:834ba23b359a2b42f75c9c8937b1b666e94085410f87e9936bba1ebc46c65e60 File: ./contract/src/contract_core/init.rs SHA3:2fc9287fcc0ad1d07efe9b0387947ba0884771b3cada76ea99dcee886ec8f205 File: ./contract/src/contract_core/mod.rs SHA3:921f718c96fc634391558b455c21a1ba2b4aad040d0e75ad45c80bee604e9bf3 File: ./contract/src/contract_core/remove_transfer.rs SHA3:cc9e64ad7b92763faa5e2b57aa12f2b222face9c138fc3b836827b9be3840058 File: ./contract/src/contract_core/set_authority.rs SHA3:b40b3ae49955efc63ba5671fa2b731979d3fa99a5247a83acc3af22bf2a8879b File: ./contract/src/contract_core/set_last_sequence.rs SHA3:ed48f827a0f00d82c64ef16a28eadb72f8fea09dee26d5f70885daaf675d2df0 File: ./contract/src/contract_core/transfer_result.rs SHA3:22eeaddad78562f4e9279c5e1adb9d07238872e49fe4c63b08173b3585c9e37d File: ./contract/src/contract_core/views.rs SHA3:094b1ec7f8aa023511bff5fd7d869369693b27b9c28b7abc0ae55ea052d7bd32

Second review scope

Repository	https://github.com/allbridge-io/near-stellar-state-contract
Commit	3fe7adc49
Whitepaper	https://docs.allbridge.io/
Functional Requirements	https://docs.allbridge.io/
Technical Requirements	https://github.com/allbridge-io/near-stellar-state-contract
Contracts	File: ./contract/src/lib.rs SHA3:dbe443fe4fdda5de91c3801b66ea1fb9e1703a6164d094c957a235c06d748f02 File: ./contract/src/types.rs SHA3:425fd88f6022e6797c4114d268813bef26ca57a2e9423ed0428fe0e82a450049 File: ./contract/src/utils.rs SHA3:28fea372bfebeae489cda8d7c87a894dd10233f3016172ba260441981ee75a7f File: ./contract/src/validator.rs SHA3:512160e4c4a2cad606cdf8272fb635b11843d3f395f5f12beac9539ce2904c0b File: ./contract/src/contract_core/create_transfer.rs SHA3:5217d3c48335f69adeb7a509e179b4566c2ef702e7b280cc3a88558293a63da9 File: ./contract/src/contract_core/init.rs SHA3:55f7603dcd48e8fc841412c8c3d215deaf04b1033373338ddd173c42f3c56c31 File: ./contract/src/contract_core/mod.rs SHA3:47f6051aa9276a7a3b8b64838938bbe2c6968d0b9ac27339cd72a6c5b8d66003 File: ./contract/src/contract_core/remove_transfer.rs SHA3:4f334a19373344c8dc636a4db5d55bedcb9531a53a999ad901a09599dd553eac File: ./contract/src/contract_core/set_authority.rs SHA3:0b1e93149d76f9688b30b09334d6a70e3f60edb96c4da13048978889c77fe084 File: ./contract/src/contract_core/set_last_sequence.rs SHA3:2d21c3eb6ea84870d9bdd18e1ac7d29e73e35f5326e08ce49e7acb784d1bf405 File: ./contract/src/contract_core/transfer_result.rs SHA3:db84cb2d8bedf2c106572961d734181a8c244535584cbf0affbde03df1585bd3 File: ./contract/src/contract_core/views.rs SHA3:c0f0850aa4b42a42660d98156da84239bfe1a8c02cab1ba6d60793b689b79822

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution but affect code quality

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **6** out of **10**.

- Functional requirements are superficial.
- Technical description is superficial.

Code quality

The total Code Quality score is **8** out of **10**.

- Development environment is configured.
- Several template code patterns were found.
- Code is not covered with comments

Test coverage

Code coverage of the project is **100%** (branch coverage).

- Deployment and authority interactions are covered with tests.
- Negative cases are covered.

Security score

As a result of the second audit, the code does not contain any severity issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.2**

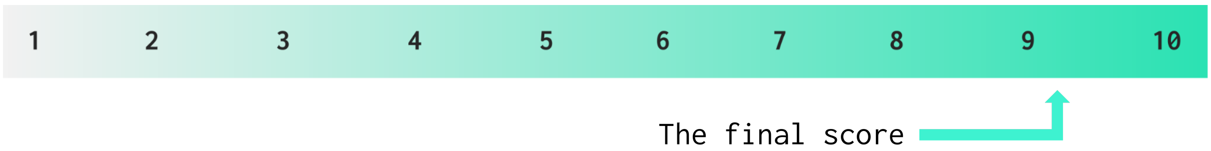

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
31 JAN 2023	1	0	0	0
09 FEB 2023	0	0	0	0

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Description	Status
Default Visibility	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Unchecked Call Return Value	The return value of a message call should be checked.	Not Relevant
Access Control & Authorization	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
Assert Violation	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Rust Functions	Deprecated built-in functions should never be used.	Passed
DoS (Denial of Service)	Execution of the code should never be blocked by a specific contract state unless required.	Passed
Race Conditions	Race Conditions and Transactions Order Dependency should not be possible.	Not Relevant
Shadowing State Variable	State variables should not be shadowed.	Passed
Weak Sources of Randomness	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Calls Only to Trusted Addresses	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	The code should not contain unused variables if this is not justified by design.	Passed
Near Standards Violation	EIP standards should not be violated.	Passed
Assets Integrity	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed

User Balances Manipulation	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply Manipulation	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Not Relevant
Gas Limit and Loops	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style Guide Violation	Style guides and best practices should be followed.	Passed
Requirements Compliance	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Passed
Tests Coverage	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed

System Overview

Allbridge is a simple, modern, and reliable way to transfer assets between different networks. It is a bridge between both EVM (Ethereum, Polygon, BSC, etc.) and non-EVM compatible (Solana, Terra, etc.) blockchains, that aims to cover L2 (like Arbitrum, Optimism) solutions and NFT transfers in the future.
Allbridge's mission is to make the blockchain world borderless and provide a tool to freely move assets between different networks.

The system contains the following contracts:

- *StateContract* — The main contract for creating/removing transfers, setting authorities,querying sequences and more.

Privileged roles

- `***admin_authority_one***` - one of the two admin authority roles
- `***admin_authority_two***` - one of the two admin authority roles
- `**add_transfer_authority**` - is the only role that can add transfer
- `***primary_oracle***` - adding a transfer requires the signature of this authority
- `***secondary_oracle***` - adding a transfer requires the signature of this authority

Risks

- Project uses a near-sdk pre-release version, which may not contain the required security or other bug fixes. An update to a higher stable version of Near sdk is recommended.

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

No high severity issues were found.

■■ Medium

No medium severity issues were found.

■ Low

1. Unformatted Code

Source code is not formatted using the Rust formatter.

Formatted code increases readability, improves maintenance and extensibility even after the original code has been modified.

Path: ./contract/contract_core/init.rs : new()

Recommendation: Format source code using Rust fmt

Status: Fixed (Revised commit: 3fe7adc)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with 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 report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

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.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

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 Consultant cannot guarantee the explicit security of the audited smart contracts.