Tetragaze

Smart contract audit

Tetragaze

May 22, 2022 | v 3.0

Pass

Tetragaze Security Team has concluded that there are no issues that can have an impact on contract security. The contract is well written and is production-ready.

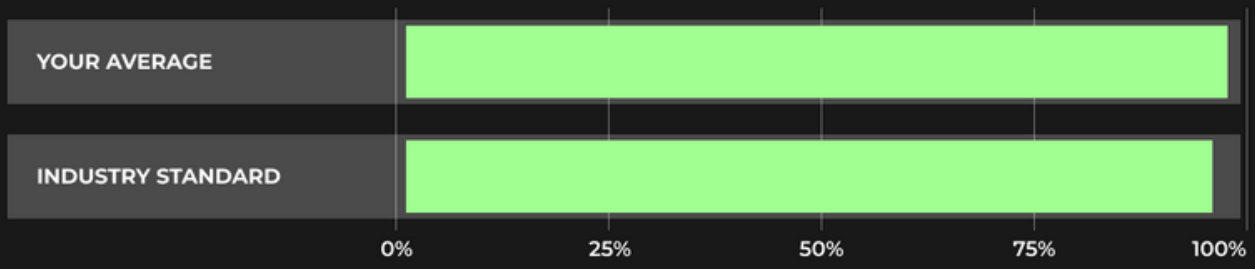
Technical summary

This document outlines the overall security of the Allbridge smart contracts, evaluated by Tetragaze Blockchain Security team.
The scope of this audit was to analyze and document the Allbridge smart contract codebase for quality, security, and correctness

Contract Status

There were no critical issues found during the audit.

Testable Code



Testable code is 99%, which is close to 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 Tetragaze recommend that the Allbridge team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table of content

Auditing Strategy and Techniques Applied

Requirements: FIP8 spec

During the review process, we made sure to carefully check that the token contract:

- Adheres to established Token standards and ensures effective implementation;
- Ensures that documentation and code comments accurately reflect the logic and behavior of the code;
- Ensures that token distribution is in line with calculations;
- Utilizes best practices to efficiently use gas, avoiding unnecessary waste;
- Employs methods that are safe from reentrance attacks and immune to the latest vulnerabilities;
- Follows best practices for code readability

Tetragaze team of expert pentesters and smart contract developers carefully evaluated the repository for security issues, code quality, and compliance with specifications and best practices. They conducted a line-by-line review, documenting any issues that were identified during the process. Special care was taken to ensure that the review was thorough and comprehensive.

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

Summary

The Allbridge EVM Contracts project includes contracts such as the Bridge, Farming, FeeOracle, Staking, and WrappedToken.

The code quality is generally good with self-explanatory functions and informative comments. However, the Readme.md file is incomplete and does not fully describe all aspects of the project.

While the majority of the code has unit tests, a few functions are not covered. During the audit, the majority of issues found were low-level and informational, with a few medium severity issues specific to the Bridge Solana Contract. There are also a few small issues with the Bridge info server that could potentially impact the security of the system, but overall the service is safe to use.

The Bridge info server provides information on transaction confirmation, supported tokens and networks, contract balance, token information, and staking pool information.

The current version supports Solana, Terra, Near, and EVM networks.

The file structure of the project is well organized and the code is divided into logical files. The code style is acceptable, but there are suggestions from linters that could improve readability.

Despite not using any frameworks to express account constraints, the developers took care to perform all necessary checks to minimize the potential for attacks. The mapping to Ethereum contracts was clear, making the auditing process straightforward.

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

Denial of locks and unlocks is possible

Medium

The CreateLock and CreateUnlock instructions of the validator include creating an account with a PDA as the final step. However, before this occurs, the validator checks to ensure that the PDA does not already correspond to an existing account. If the account holds lamports, this check will fail. An attacker could potentially exploit this by sending the minimum amount of lamports required for specific lock_ids to the PDA.

New validators can repeat the process of unlocking previous validators' accounts.

Medium

To prevent replay attacks, the create_unlock instructions utilize a program derived address (PDA) generated from the validator_account key and a lock_id as seeds. Additionally, the validator's key is not included in the signed message during signature validation. If the validator for the bridge is changed via the SetValidator instruction, the new validator could potentially validate previous unlocks as valid signatures.

Complete Analysis

Instead of including code logic, make a comment about it.

Low

The code should include the logic that is necessary for a stable workflow, rather than just mentioning it in comments as a warning.

Precision loss during staking due to casting

Low

The calculation for the amount to be transferred during a withdrawal or deposit of stakes involves multiplying two u64 types and dividing by a third u64 type, which are all converted to u128. However, at the end of the calculation, the result is simply converted back to u64 without any checks, potentially causing a loss of precision.

During the removal of a token, the account data is not reset to zero.

Low

When tokens are removed, the Asset accounts associated with them are also deleted. This process involves transferring the remaining lamports from these accounts. The solana runtime will delete these accounts at the end of the transaction, but instructions within the same transaction can still access and utilize the data in these Asset accounts before they are fully deleted.

Complete Analysis

Address validation is missing for zero addresses.

Low

To maintain contract ownership, it is important to properly initialize the smart contract. To do this, it is necessary to verify that the recipient of ETH is not the zero address, as any ETH sent to the zero address will be lost. This operation should be included in the smart contract to ensure that the contract ownership is maintained.

Attacks that involve reentering a system or function multiple times.

Low

Smart contracts may be vulnerable to reentrancy attacks when they incorporate untrusted data in a function call without proper validation or the ability to exit the function. This can occur when external calls or ETH transfers are made before storage modification or when an event is emitted. If the external contract or ETH recipient is untrusted, these attacks may result in the theft of funds.

An unlock may be created for an unsupported token.

Low

Before checking if the token is supported, the function first calls an external contract to create a message hash and validate the signature.

Complete Analysis

Weaknesses in dependent software that can be indirectly accessed through another software.

Low

There are some transitive dependencies with known vulnerabilities: follow-redirects version <=1.14.7 may expose sensitive information, json-schema version <0.4.0 may have prototype pollution, node-fetch version <2.6.7 may expose sensitive information to an unauthorized actor, node-forge version <1.0.0 may have prototype pollution, shelljs version <0.8.5 may have improper privilege management, and simple-get version <2.8.2 may expose sensitive information.

The frequent use of "find_program_address."

Informational

According to the Solana documentation, the find_program_address function may not be suitable for programs that need high performance as it may take a significant amount of time. Alternative methods exist for validating PDAs without the use of find_program_address.

We are thankful for the opportunity to collaborate with the Allbridge team.

This document's statements should not be taken as investment or legal advice, and the authors cannot be held responsible for any decisions made based on them.

It is suggested by the Tetragaze Security Team that the Allbridge team implement a bug bounty program in order to encourage external parties to scrutinize the smart contract further.