QuillAudits

AUDIT REPORT

May , 2025

For

ALVARA

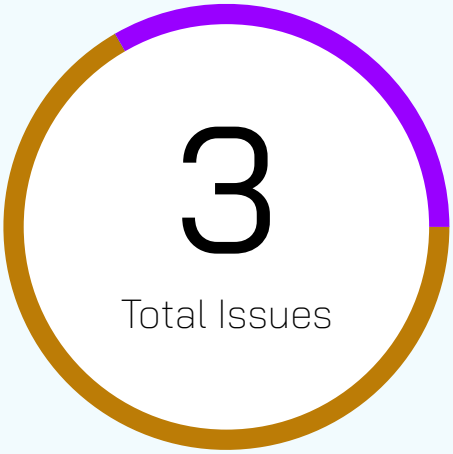
Table of Content

Executive Summary

Project name	Alvara
Project URL	https://www.alvaraprotocol.io
Overview	The Alvara platform enables the creation and self-management of tokenized basket funds. Its factory provides a platform for users to design and mint their unique BTS tokens. In essence, anyone can step into the shoes of a fund manager by incorporating tokens from any supported blockchain into their BTS.
Audit Scope	The scope of this Audit was to analyze the Alvara Smart Contracts for quality, security, and correctness.
Source Code link	https://github.com/Alvara-Protocol/alvara-contracts
Contracts in Scope	AlvaStaking.sol, AlvaraDao.sol, BTS.sol, and Factory.sol
Branch	development
Commit Hash	1aa494124267f80e1ed6b02b6144c244baad45a6
Language	solidity
Blockchain	EVM
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	17th April 2025 to 18th April 2025

Updated Code Received	6th May 2025
Review 2	7th May 2025 - 8th May 2025
Fixed In	542ed90de2c815c790e298b03cc996d7c0b57ef7

Number of Issues per Severity



High	0 (0.00%)
Medium	2 (66.67%)
Low	0 (0.00%)
Informational	1 (33.33%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	0	**2**	0	**1**
Acknowledged	0	0	0	0
Partially Resolved	0	0	0	0

Checked Vulnerabilities

- ☑ Access Management
- ☑ Arbitrary write to storage
- ☑ Centralization of control
- ☑ Ether theft
- ☑ Improper or missing events
- ☑ Logical issues and flaws
- ☑ Arithmetic Computations Correctness
- ☑ Race conditions/front running
- ☑ SWC Registry
- ☑ Re-entrancy
- ☑ Timestamp Dependence
- ☑ Gas Limit and Loops
- ☑ Exception Disorder
- ☑ Gasless Send
- ☑ Use of tx.origin
- ☑ Malicious libraries
- ☑ Compiler version not fixed
- ☑ Address hardcoded
- ☑ Divide before multiply
- ☑ Integer overflow/underflow
- ☑ ERC's conformance
- ☑ Dangerous strict equalities
- ☑ Tautology or contradiction
- ☑ Return values of low-level calls
- ☑ Missing Zero Address Validation
- ☑ Private modifier
- ☑ Revert/require functions
- ☑ Multiple Sends
- ☑ Using suicide
- ☑ Using delegatecall
- ☑ Upgradeable safety
- ☑ Using throw

- [x] Using inline assembly
- [x] Unsafe type inference
- [x] Style guide violation
- [x] Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools And Platforms Used For Audit

Remix IDE, Foundry, Solhint, Mythril, Slither, Solidity statistical analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

Security vulnerabilities identified that must be resolved and are currently unresolved.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/ impact of the security issue, but are not completely resolved.

Medium Severity Issues

Block.timestamp does not equate to instantaneous execution

Resolved

Path

BTS.sol, Factory.sol

Function

-

Description

Using block.timestamp as the expiry or deadline of an operation doesn't enforce immediate execution. Instead, it indicates that the caller is comfortable with the transaction being included in whichever block it is mined in.
However, this opens the door for potential manipulation by malicious miners, who could delay the transaction (for example, through the flashbots mempool used for bundling transactions) until conditions are most advantageous to them.
This delay could lead to maximum slippage under the transaction's slippage tolerance or trigger unfavorable events like liquidations.
To mitigate this risk, expiry times should be determined off-chain and set explicitly by the caller, reducing the potential for unnecessary Miner Extractable Value (MEV).

There are three instances of this issue:
1. contribute()
2. _swapTokensForTokens()
3. createBTS()

Recommendation

Ensure a deadline parameter is passed.

Hardcoded slippage might be problematic

Resolved

Path

BTS.sol

Function

withdraw()

Description

The problem with hardcoded slippage is that it can potentially leads to frozen funds during periods of high volatility.

Consider withdraw function that allows users to withdraw their tokens from basket.
After _withdraw action is executed, the function proceeds to send remaining amount to feeCollector.

```
uint256 ethAmount = _tokensToEth(feeAmounts, payable(feeCollector), 500);
```

Hardcoding slippage 500 i.e., 5% here can be problematic in scenarios of a volatile market causing unnecessary swap revert and causing the revert of entire function.

Recommendation

Separate user withdrawal and swap process such that withdrawals do not revert for users and owner can manually swap fee afterwards.

Informational Severity Issues

Lack of event emission in updateMinBTSCreation-Amount

Resolved

Path

Factory.sol

Function

updateMinBTSCreationAmount

Description

Unlike other state update functions, updateMinBTSCreationAmount does not emit an event.

Recommendation

Emit an event for critical state change.

Functional Tests

Some of the tests performed are mentioned below:

- ✔ Valid bts creation
- ✔ State update check

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of Alvara. We performed our audit according to the procedure described above.

Issues of medium and informational severity were found.
Alvara Team resolved them all

Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event—threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code.

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.

Stay proactive. Stay secure.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem

7+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

May , 2025

For

ALVARA

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com