QuillAudits

AUDIT REPORT

August 2025

For

alkimi

Table of Content

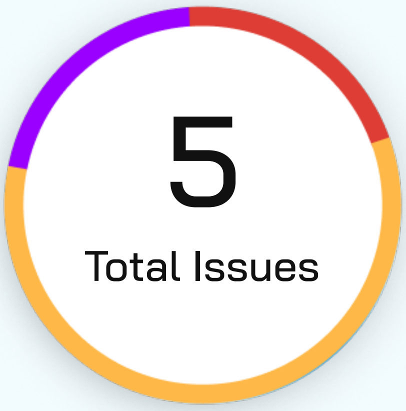
Executive Summary

Project Name	Alkimi
Project URL	https://www.alkimi.org/
Overview	`**AlkimiSuiTransfer**' is a smart contract that accepts $ADS token deposits and emits events for cross-chain transfers, featuring dual authorization modes (whitelist or signature verification), configurable daily limits based on noon-to-noon periods, a multiplier system for weighted deposits, emergency pause capabilities, and rescue functions for tokens, ETH, and NFTs to ensure flexibility and security in operations.
Audit Scope	The scope of this Audit was to analyze the Alkimi Smart Contracts for quality, security, and correctness.
Source Code link	https://github.com/Alkimi-Exchange/sui_contracts/blob/feature/transfer-and-claim-audit/token_transfer_eth/contracts/AlkimiSuiTransfer.sol
Branch	feature/transfer-and-claim-audit
Contracts in Scope	AlkimiSuiTransfer.sol
Commit Hash	2e958eada3b1144dff42ba54e3b93f24dd70736d
Language	Solidity
Blockchain	EVM
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	4th August 2025 - 11th August 2025
Updated Code Received	13th August 2025
Review 2	13th August 2025 - 14 th August 2025
Fixed In	4fa5a38a43defcca16226ccf68a08b9650fd45ab

Verify the Authenticity of Report on QuillAudits Leaderboard:

https://www.quillaudits.com/leaderboard

Number of Issues per Severity



🟥	Critical	0 (0%)
🟥	High	1 (20.0%)
🟧	Medium	0 (0.0%)
🟨	Low	3 (60.0%)
🟪	Informational	1 (20.0%)

Severity

Issues	🟥 Critical	🟥 High	🟧 Medium	🟨 Low	🟪 Informational
Open	0	0	0	0	0
Acknowledged	0	1	0	1	1
Partially Resolved	0	0	0	0	0
Resolved	0	0	0	2	0

Summary of Issues

Issue No.	Issue Title	Severity	Status
1	Missing Per-User Daily Transfer Limit	High	Acknowledged
2	Incorrect Sui Wallet Validation Leading to Locked Funds	Low	Resolved
3	Mismatch Between Signature Verification Implementation and Code Comments Leading to Incorrect Behavior	Low	Resolved
4	Potential Sell Pressure via Exploitation of Daily Limit Time Window	Low	Acknowledged
5	Redundant Balance Checks in Token Transfer Logic	Informational	Acknowledged

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**
- ✓ **Using inline assembly**
- ✓ **Style guide violation**
- ✓ **Unsafe type inference**
- ✓ **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 Static Analysis.

Types of Severity

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

Critical: Immediate and Catastrophic Impact

Critical issues are the ones that an attacker could exploit with relative ease, potentially leading to an immediate and complete loss of user funds, a total takeover of the protocol's functionality, or other catastrophic failures. Critical vulnerabilities are non-negotiable; they absolutely must be fixed.

High (H): Significant Risk of Major Loss or Compromise

High-severity issues represent serious weaknesses that could result in significant financial losses for users, major malfunctions within the protocol, or substantial compromise of its intended operations. While exploiting these vulnerabilities might require specific conditions to be met or a moderate level of technical skill, the potential damage is considerable. These findings are critical and should be addressed and resolved thoroughly before the contract is put into the Mainnet.

Medium (M): Potential for Moderate Harm Under Specific Circumstances

Medium-severity bugs are loopholes in the protocol that could lead to moderate financial losses or partial disruptions of the protocol's intended behavior. However, exploiting these vulnerabilities typically requires more specific and less common conditions to occur, and the overall impact is generally lower compared to high or critical issues. Whilenot as immediately threatening, it's still highly recommended to address these findings to enhance the contract's robustness and prevent potential problems down the line.

Low (L): Minor Imperfections with Limited Repercussions

Low-severity issues are essentially minor imperfections in the smart contract that have a limited impact on user funds or the core functionality of the protocol. Exploiting these would usually require very specific and unlikely scenarios and would yield minimal gain for an attacker. While these findings don't pose an immediate threat, addressing them when feasible can contribute to a more polished and well-maintained codebase.

Informational (I): Opportunities for Improvement, Not Immediate Risks

Informational findings aren't security vulnerabilities in the traditional sense. Instead, they highlight areas related to the clarity and efficiency of the code, gas optimization, the quality of documentation, or adherence to best development practices. These findings don't represent any immediate risk to the security or functionality of the contract but offer valuable insights for improving its overall quality and maintainability. Addressing these is optional but often beneficial for long-term health and clarity.

Types of Issues

Open Security vulnerabilities identified that must be resolved and are currently unresolved.	**Resolved** These are the issues identified in the initial audit and have been successfully fixed.
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.

Severity Matrix

Impact

Likelihood	🟥 High	🟧 Medium	🟨 Low
🟥 High	Critical	High	Medium
🟧 Medium	High	Medium	Low
🟨 Low	Medium	Low	Low

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- **Low** - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

High Severity Issues

Missing Per-User Daily Transfer Limit

Acknowledged

Description

According to the provided specifications, the transfer mechanism should enforce two distinct restrictions:

1. **Global Daily Limit** – A maximum amount of ADS tokens that can be locked by all users collectively within a single day.
2. **Per-User Daily Limit** – A maximum amount of ADS tokens that any individual holder can lock within a single day, ensuring that no single user monopolizes the global daily limit.

While the global daily limit appears to be implemented in the **`AlkimiSuiTransfer`** contract, the per-user daily limit is absent. This omission allows a single account to potentially consume the entire global daily limit, preventing other users from locking tokens that day.

Impact

Failure to enforce individual-level restrictions violates a core business requirement and may lead to unfair usage, reduced participation, and potential service disruption for other users.

Recommendation

Implement a per-user daily limit tracking mechanism to enforce individual caps in addition to the global daily limit. This will ensure fair distribution of daily transfer capacity and compliance with the intended business logic.

Low Severity Issues

Incorrect Sui Wallet Validation Leading to Locked Funds

Resolved

Description

In token_transfer`_eth/contracts/AlkimiSuiTransfer.sol#L634`, the if statement used to validate the length of the provided `suiWallet` does not strictly enforce the required length of 66 characters. This means that invalid Sui wallet addresses can pass the validation check when the `deposit` function is called.

If an invalid wallet address is provided, the funds will be transferred to the treasury wallet but will remain inaccessible on the Sui blockchain without manual intervention. As a result, the affected user will be unable to claim their funds, effectively locking them.

Similar problem exists in the `verifyDepositSignature` function at `token_transfer_eth/contracts/AlkimiSuiTransfer.sol#L1110`

Impact

Funds sent with an invalid Sui wallet address become irretrievable without human intervention, leading to user dissatisfaction and potential loss of trust in the system.

Recommendation

We recommend replacing the current if condition at `token_transfer_eth/contracts/AlkimiSuiTransfer.sol#L634` and at `token_transfer_eth/contracts/AlkimiSuiTransfer.sol#L1110` with the following strict length check to ensure only valid addresses are accepted:

```
if (bytes(suiWallet).length != 66) {
    revert InvalidSuiWallet(suiWallet);
}
```

Mismatch Between Signature Verification Implementation and Code Comments Leading to Incorrect Behavior

Resolved

Description

In **`token_transfer_eth/contracts/AlkimiSuiTransfer.sol#L655`**, the contract implements EIP-191 style signature verification. However, according to the code comments at **`token_transfer_eth/contracts/AlkimiSuiTransfer.sol#L598`**, the expected approach is EIP-712 style signature verification.

This inconsistency can cause confusion for developers and backend systems. If the backend generates signatures in the EIP-712 format (as suggested by the comments) while the contract verifies using EIP-191, signature validation will fail. As a result, valid transactions could be incorrectly rejected, leading to operational disruptions and misaligned expectations between the off-chain and on-chain components.

Impact

A mismatch in signature formats can prevent legitimate user operations from being processed, causing failed transactions and potential downtime.

Recommendation

Ensure consistency between the implemented signature verification style and the documented expectations. Either:

- Update the code comments to reflect the actual implemented EIP-191 verification, or
- Update the implementation to align with the intended EIP-712 standard, and ensure backend systems generate the correct type of signature accordingly.

Potential Sell Pressure via Exploitation of Daily Limit Time Window

Acknowledged

Description

In **`token_transfer_eth/contracts/AlkimiSuiTransfer.sol (lines 718-731)`**, the contract implements a 24-hour epoch-based cycle to reset **`currentDailyAmount`**. This refresh cycle operates continuously, regardless of whether any user has locked ADS tokens during the period.

This design can be exploited by a malicious depositor who performs a maximum deposit immediately before the 24-hour refresh, and then another maximum deposit immediately after the refresh. This effectively allows them to lock and subsequently unlock twice the intended daily limit in a very short timeframe.

If the attacker chooses to sell both unlocked batches at once, it could cause significant sell pressure on the ALKIMI token in the Sui market. While the attacker would also face lower execution prices due to their selling activity, this behaviour breaks the intended daily limit assumption of the protocol.

Impact

- Does not directly cause guaranteed financial loss, but allows circumvention of the intended token flow controls.
- May lead to temporary market disruption and price volatility.

Recommendation

Consider implementing a per-user rolling time window or timestamp-based cooldown mechanism to ensure that deposit limits are enforced over any given 24-hour period per user, preventing back-to-back limit circumvention.

Informational Issues

Redundant Balance Checks in Token Transfer Logic

Acknowledged

Description

In the **`AlkimiSuiTransfer`** contract, multiple instances include explicit checks to verify that the sender holds a sufficient ADS token balance before initiating a transfer. These validations are redundant because the ERC20 **`transfer`** and **`transferFrom`** functions inherently perform balance checks and revert if the sender's balance is insufficient.

Maintaining these unnecessary checks increases gas costs slightly and adds extra code complexity without providing additional security guarantees.

Impact

While the issue does not introduce any functional vulnerabilities, it results in marginally higher transaction costs and reduced code clarity.

Recommendation

Remove the redundant balance checks and rely on the native ERC20 token transfer behavior. This will improve gas efficiency and simplify the codebase.

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 Alkimi. We performed our audit according to the procedure described above.

Issues of high, low, and informational severity were found. The Alkimi team resolved two and acknowledged the remaining.

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 leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With seven years of expertise, we've secured over 1400 projects globally, averting over $3 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

QuillAudits

7+ Years of Expertise	**1M+** Lines of Code Audited
50+ Chains Supported	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

August 2025

For

alkimi

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com