QuillAudits

AUDIT REPORT

August 2025

For

alkimi

Table of Content

Executive Summary

Project Name	Alkimi
Protocol Type	Token
Project URL	https://www.alkimi.org/
Overview	ALKIMI token with 1 billion total supply and 9 decimals, featuring secure treasury management through a ProtectedTreasury that stores the TreasuryCap in dynamic object fields. The contract implements role-based governance with an owner and up to 2 admins, including a two-step ownership transfer process and functions to add/remove admins or make the contract permanently immutable. Core functionality includes a sophisticated burn mechanism that reduces total supply, handles partial burns with change return, and tracks cumulative burned tokens with overflow protection. The contract emphasizes security through extensive validation, categorized error codes, and comprehensive event emission for all state changes including minting, burning, and administrative actions.
Audit Scope	The scope of this Audit was to analyze the Token gen Smart Contracts for quality, security, and correctness.
Source Code link	https://github.com/Alkimi-Exchange/sui_contracts/blob/main/token_gen_move_contract/sources/token.move
Contracts in Scope	token_gen_move_contract/sources/token.move
Branch	main
Commit Hash	00dfc26d2a81f8c6fdffe744ddca92ff9a62bb1e
Language	Move
Blockchain	Sui
Method	Manual Analysis, Functional Testing, Automated Testing

Executive Summary

Review 1	1st August 2025 - 4th August 2025
Updated Code Received	5th August 2025

Verify the Authenticity of Report on QuillAudits Leaderboard:

https://www.quillaudits.com/leaderboard

Number of Issues per Severity

0

Total Issues

🟥	Critical	0 (0%)
🟥	High	0 (0%)
🟧	Medium	0 (0%)
🟨	Low	0 (0%)
🟪	Informational	0 (0%)

Severity

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

Summary of Issues

No Issues Found

Checked Vulnerabilities

- ☑ Transaction-ordering dependence
- ☑ Timestamp dependence
- ☑ Denial of service / logical oversights
- ☑ Timestamp dependence
- ☑ Access control
- ☑ Code clones, functionality duplication
- ☑ Witness Type
- ☑ Integer overflow/underflow by bit operations
- ☑ Number of rounding errors
- ☑ Business logic contradicting the specification
- ☑ Number of rounding errors
- ☑ Gas usage
- ☑ Unchecked CALL Return Values
- ☑ Centralization of power

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.

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. While not 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		
		🟥 High	🟧 Medium	🟨 Low
Likelihood	🟥 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.

Functional Tests

Some of the tests performed are mentioned below:

- ✓ Unauthorised Use of Shared Objects
- ✓ Mint Integrity (Only Owner can mint once during initialization)
- ✓ Burn Integrity (Only Admin or Owner can burn tokens, must not exceed coin value)
- ✓ Add Admin (Only Owner can call this function, admin must not be owner or already exist)
- ✓ Remove Admin (Only Owner can call this function, admin must exist)
- ✓ Transfer Ownership (Two-step process: Owner initiates, new owner accepts)
- ✓ Cancel Ownership Transfer (Only Owner can cancel before acceptance)
- ✓ TreasuryCap Supply Integrity (Post-burn, supply must match system-reported total supply)
- ✓ Max Admin Limit Enforcement (Adding admin fails when limit reached)

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.

Threat Model

Contract	Function	Threats
alkimi::alkimi	burn()	• Griefing: Admin burns tokens from vaults they control • Economic Attack: Coordinated large burns to signal scarcity • Event Spamming: Zero-value burns to inflate logs
alkimi::alkimi	add_admin() / remove_admin()	• Admin Lock-out: Owner removes all admins and loses key • Social Engineering: Malicious actor convinces owner to add them
alkimi::alkimi	transfer_full_ownership( ) / accept_ownership()	• Ownership Hijack: Compromised owner transfers • Zombie Transfer DoS: Stale transfer requests remain pending
alkimi::alkimi	safe_add_burned()	• Burn Stats Pollution: High-velocity burns inflate logs and metrics
alkimi::alkimi	get_current_supply()	• Supply Desync Assertion: If TreasuryCap supply and coin::total_supply() mismatch, contract panics • Denial of Service: Any mismatch bricks the contract

Closing Summary

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

Code looks good,No Issues Found.

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