QuillAudits

AUDIT REPORT

May , 2025

For

HETRACOIN

Table of Content

Executive Summary

Project name	Hetra Coin
Project URL	https://www.hetracoin.io/
Overview	HetraCoin provides a robust, secure token infrastructure with multiple integrated components: HetraCoin: Core token implementation with dynamic admin management Governance: Secure administration and token supply management Treasury: Timelock-protected fund management Escrow: Secure peer-to-peer transactions with dispute resolution Staking: Token staking system with rewards LiquidityPool: Decentralized token exchange functionality Proposal: On-chain governance voting mechanism
Audit Scope	The scope of this Audit was to analyze the Hetra Coin Smart Contracts for quality, security, and correctness.
Source Code link	https://github.com/Hetrafi/hetracoin-sui
Contracts in Scope	Hetracoin.move Governance.move Treasury.move
Branch	Main
Commit Hash	1319210850818316e85bf8aa5a9e4adb8807a8fd
Language	Move
Blockchain	Sui
Method	Manual Analysis, Functional Testing, Automated Testing

Review 1	6th may 2025
Updated Code Received	9th May 2025
Review 2	9th May 2025
Fixed In	hetracoin:136a49f5566505ccb3299bf9b63ed23a16397b98 Governance: 941b1c92975404d43633bb540b0ad8ee5d14c091

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] Style guide violation
- [x] Unsafe type inference
- [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.

High Severity Issues

Broken Two-Step Governance Transfer.

Resolved

Path

Governance.move

Function

initiate_governance_transfer(),accept_governance_transfer()

Description

The initiate_governance_transfer function only sends a request object to the intended new_admin, failing to transfer the necessary capabilities(TreasuryCap, AdminCap) alongside it.

Consequently, the accept_governance_transfer function requires the new_admin (the caller) to provide these capabilities, which they never received. This makes it impossible for the new_admin to complete the transfer, making the two-step mechanism unusable as designed.

Recommendation

Simplify governance transfer by making it two distinct manual steps:
First: New admin calls accept_governance_transfer to update AdminRegistry only
Second: Old admin calls transfer_treasury_cap to move TreasuryCap and AdminCap (implement the transfer_treasury_cap completely)

Medium Severity Issues

initiate_governance_transfer(),accept_governance_transfer()

Resolved

Path

HetraCoin.move

Function

secure_transfer()

Description

The HetraCoin::secure_transfer function, intended for general token transfers, does not check the EmergencyPauseState. This allows users to transfer tokens even when the system is globally paused, bypassing the intended emergency stop functionality.

Recommendation

Add the EmergencyPauseState as a parameter and include an assert!(!pause_state.paused, E_PAUSED); check

Missing Pause Check in burn Function

Resolved

Path

Governance.move

Function

burn()

Description

The Governance::burn function allows token burning even when the system is paused via EmergencyPauseState. This undermines the emergency stop.

Recommendation

Add the EmergencyPauseState as a parameter and include an assert!(!pause_state.paused, E_PAUSED); check

Low Severity Issues

Missing Zero-Amount Check in mint and burn Functions

Resolved

Path

Governance.move

Function

mint() , burn()

Description

Both Governance::mint and Governance::burn functions lack validation to ensure amount values are greater than zero. This differs from secure_transfer which explicitly checks with assert!(amount > 0, E_ZERO_AMOUNT).
Zero-amount operations waste gas, pollute event logs.

Recommendation

Add zero-amount validation to both functions.

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 Hetra Coin. We performed our audit according to the procedure described above.

issues of High , Medium and Low severity were found. Hetra Coin 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

QuillAudits

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

HETRACOIN

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com