QuillAudits
Web3 Security

AUDIT REPORT

December, 2024

For

cSIGMA

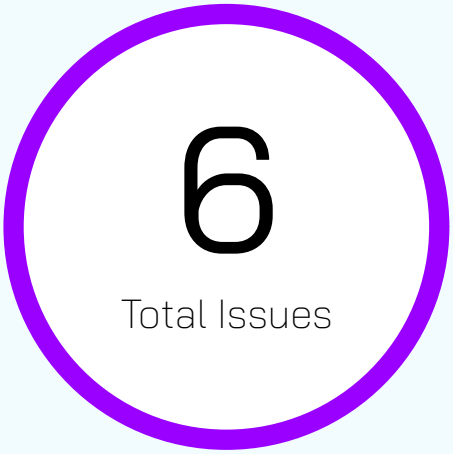
Table of Content

Executive Summary

Project name	CSigma
Overview	The system is a comprehensive financial protocol designed for real-world asset tokenization, fund management, and staking. It features modular contracts for creating and managing investment pools, staking assets, and distributing rewards. The system leverages AccessControlUpgradeable for role management and supports upgradeable contracts for flexibility and future enhancements. Its functionality includes secure asset deposits, governance-driven configuration, and user-focused operations like staking, unstaking, and reward withdrawals.
Method	Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives.
Blockchain	Ethereum, Arbitrium
Audit Scope	The Scope of the Audit is to Analyse the Security,COde Quality and Correctness of csigma Contracts
Contracts In Scope	https://github.com/csigma-labs/upgradeable-proxy-contracts/tree/release-v2.1/contracts/v2 a) Factory b) Pool c) Staking pool d) Staking pool extension e) Fund manager
Fixed In	https://github.com/csigma-labs/upgradeable-proxy-contracts/commit/c5a853cfb42b4f032e4b7d4c3da1158c8e3c01bf
Project URL	https://csigma.finance

Language	Solidity
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	7th November 2024 - 28th November 2024
Updated Code Received	4th December 2024
Review 2	4th December 2024 - 5th December 2024

Number of Issues per Severity



High	0 (0.00%)
Medium	0 (0.00%)
Low	0 (0.00%)
Informational	6 (100.00%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	0	0	0	**4**
Acknowledged	0	0	0	**2**
Partially Resolved	0	0	0	0

Checked Vulnerabilities

- ☑ Re-entrancy
- ☑ Timestamp Dependence
- ☑ Gas Limit and Loops
- ☑ Exception disorder
- ☑ Gasless send
- ☑ Use of tx.origin
- ☑ Compiler version not fixed
- ☑ 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
- ☑ Malicious libraries
- ☑ 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 Issues

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

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.

Issue Status

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.

Informational Severity Issues

Missing Validation for Percentage in updateNonReservePercentage

Resolved

Path

contracts/v2/CsigmaV2Pool.sol

Function

updateNonReservePercentage

Description

The updateNonReservePercentage function allows setting any value for _percentage without validation. This value is used in the deposit function to calculate the portion of assets transferred to the fundManager as (assets * nonReservePercentage) / 10000. If nonReservePercentage is set to an excessively high value (e.g., greater than 10000, representing 100%), it can result in all deposited assets being transferred to the fundManager, leaving no reserves in the pool.

Recommendation

Add a validation check in updateNonReservePercentage to ensure _percentage remains within a valid range.

Missing Validation for pauseDuration in updatePause-Duration

Resolved

Path

contracts/v2/CsigmaV2Pool.sol

Function

updatePauseDuration

Description

The updatePauseDuration function allows the admin to set an arbitrary value for pauseDuration without validation. This could result in excessively long or inappropriate pause durations, potentially causing unnecessary disruptions to the system's availability and user operations.

Recommendation

Add validation to ensure pauseDuration is within reasonable bounds.

Missing Event Emission in CsigmaV2Pool for Critical State Updates

Resolved

Path

contracts/v2/CsigmaV2Pool.sol

Function

updateInitialDepositTime, updatePauseDuration, updatePauseStartTime, updateNonReservePercentage, updatePoolSize

Description

The CsigmaV2Pool contract lacks event emissions for several critical state-changing functions, including updateInitialDepositTime, updatePauseDuration, updatePauseStartTime, updateNonReservePercentage, and updatePoolSize. This omission reduces transparency and makes it difficult to monitor or audit changes to the pool's configuration. Without events, off-chain systems and stakeholders cannot track these updates effectively.

Recommendation

Ensure all critical actions, especially those involving state changes or administrative controls, emit appropriate events. For example, emit events for role changes, configuration updates, and fund transfers to enhance traceability and facilitate off-chain monitoring.

Centralization Risk in Pool Status Management

Acknowledged

Path

contracts/v2/CsigmaV2StakingPool.sol,
contracts/v2/CsigmaV2StakingPoolExtension.sol

Function

updatePoolStatus

Description

The updatePoolStatus function allows the admin to arbitrarily change the pool's status to CLOSED or PENDING, which disables key user actions like staking or unstaking. When the pool is not in an ACTIVE state, users cannot withdraw their funds, even if their assets are unlocked. This introduces a centralization risk where an admin could unintentionally or maliciously block user funds.

Recommendation

Implement additional safeguards such as:

1. Requiring a governance vote or multi-signature approval for status changes.
2. Ensuring users can always withdraw unlocked funds, regardless of the pool's status.
3. Adding a time delay to status changes, allowing users to act before the pool status is updated.

Teams Comment

Csigma Team's Comment: The DEFAULT_ADMIN_ROLE, responsible for managing pool status updates, is controlled through a multi-signature (multi-sig) approval process. This setup ensures that no single entity can arbitrarily or maliciously alter the pool's status, thereby minimizing the risk of user funds being locked. Any status change requires approval from multiple trusted parties, enhancing both security and accountability. Additionally, governance rules outline clear scenarios for status updates, providing transparency to users.
Thus, the current design balances security and operational flexibility without compromising user control over their funds.

Missing Validation in updateMinMaxDurationAllowed

Resolved

Path

contracts/v2/CsigmaV2StakingPool.sol,
contracts/v2/CsigmaV2StakingPoolExtension.sol

Function

updateMinMaxDurationAllowed

Description

The updateMinMaxDurationAllowed function allows the admin to set arbitrary values for _min and _max without validation. This can result in configurations where minStakeDurationAllowed is greater than maxStakeDurationAllowed, creating invalid or contradictory staking conditions.

Recommendation

Add a validation check to ensure _min is always less than or equal to _max.

Potential DoS Risk in deployFunds with Excessive Pools

Acknowledged

Path

contracts/v2/CsigmaV2FundManager.sol

Function

deployFunds, addPool

Description

The deployFunds function iterates through all v1Pools to distribute funds based on their allocation. If the number of pools becomes too large, this loop may exceed gas limits, causing the function to fail (DoS risk). While the total allocation is capped at 10000, there is no restriction on the number of pools that can be added. This could result in operational issues and blocked

Recommendation

Implement a cap on the maximum number of pools allowed in v1Pools during addPool.

Teams Comment

CSigma Team's Comment: While the function currently has no hard cap on the number of pools, operational constraints, and governance controls inherently limit pool creation. Pools are segregated based on the risk profile, APR or yield, geographic region and industry, and no Master Pool will have a large number of child pools to be managed by this Fund Manager Smart Contact method. In real-world scenarios, the number of active pools managed by this function in a single call is strategically managed to remain low (typically five to seven). .

In the event that pool numbers grow beyond operational thresholds, the poolManager retains the ability to dynamically manage and remove excess pools, ensuring that capital can be redeployed without disruption. This flexibility ensures seamless fund deployment regardless of pool count.

The current approach balances scalability and efficiency, and no additional storage or hard limits are necessary given the governance oversight and operational safeguards already in place.

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

Some issues of informational severity were found. Some suggestions, gas optimizations and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Csigma. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Csigma. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of Csigma to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

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

6+ Years of Expertise	**1M+** Lines of Code
$30B+ Secured in Digital Assets	**1K+** Projects

Follow Our Journey

AUDIT REPORT

December, 2024

For

cSIGMA

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

mailto:audits@quillhash.com