RESONANCE

RHEA

Rhea Finance
Zcash Transparent Verifier Smart Contract Audit Report

Blockchain, Emerging Technology, and Web2
CYBERSECURITY PRODUCT & SERVICE ADVISORY

Document Control

PUBLIC **FINAL**(v2.1)

Audit_Report_RHEA-ZTV_FINAL_21

Nov 28, 2025	○	v0.1	João Simões: Initial draft
Nov 28, 2025	○	v0.2	João Simões: Added findings
Dec 1, 2025	●	v1.0	Charles Dray: Approved
Dec 4, 2025	○	v1.1	João Simões: Reviewed findings
Dec 4, 2025	●	v2.0	Charles Dray: Finalized
Dec 4, 2025	○	v2.1	Charles Dray: Published

Points of Contact	Marco Sun	Rhea Finance	marco@ref.finance
	Charles Dray	Resonance	charles@resonance.security
Testing Team	João Simões	Resonance	joao@resonance.security
	Luis Arroyo	Resonance	luis.arroyo@resonance.security
	George Skouroupathis	Resonance	george@resonance.security

Copyright and Disclaimer

© 2025 Resonance Security, Inc. All rights reserved.

The information in this report is considered confidential and proprietary by Resonance and is licensed to the recipient solely under the terms of the project statement of work. Reproduction or distribution, in whole or in part, is strictly prohibited without the express written permission of Resonance.

All activities performed by Resonance in connection with this project were carried out in accordance with the project statement of work and agreed-upon project plan. It's important to note that security assessments are time-limited and may depend on information provided by the client, its affiliates, or partners. As such, the findings documented in this report should not be considered a comprehensive list of all security issues, flaws, or defects in the target system or codebase.

Furthermore, it is hereby assumed that all of the risks in electing not to remedy the security issues identified henceforth are sole responsibility of the respective client. The acknowledgement and understanding of the risks which may arise due to failure to remedy the described security issues, waives and releases any claims against Resonance, now known or hereafter known, on account of damage or financial loss.

Contents

© 2025 Resonance Security, Inc

Executive Summary

Rhea Finance contracted the services of Resonance to conduct a comprehensive security audit of their smart contracts between November 24, 2025 and December 1, 2025. The primary objective of the assessment was to identify any potential security vulnerabilities and ensure the correct functioning of smart contract operations.

During the engagement, Resonance allocated 2 engineers to perform the security review. The engineers, including an accomplished professional with extensive proficiency in blockchain and smart-contract security, encompassing specialized skills in advanced penetration testing, and in-depth knowledge of multiple blockchain protocols, devoted 5 days to the project. The project's test targets, overview, and coverage details are available throughout the next sections of the report.

The ultimate goal of the audit was to provide Rhea Finance with a detailed summary of the findings, including any identified vulnerabilities, and recommendations to mitigate any discovered risks. The results of the audit are presented in detail further below.

System Overview

The Multichain Account (McA) Protocol is a NEAR-based system that enables seamless multichain interoperability without relying on side-chains or light clients. It leverages NEAR's native on-chain signature verification to link a user's wallets and public keys across multiple blockchains (e.g., EVM networks, Solana, Bitcoin), allowing them to interact with NEAR dApps using their existing wallets—without the need to maintain a NEAR account. Each user is assigned a dedicated McA smart contract, deployed from a shared NEAR Global Contract, which serves as their on-chain identity and multichain wallet abstraction.

The protocol is composed of several key components: the Account Manager (AM), which acts as the factory and registry for McA deployments; the McA contract, which verifies signatures, manages wallet bindings, and executes authorized business actions; Signed Business Relayers (SBRs) that deliver user-signed messages on-chain; and Near Intents, a message asset bridge that enables cross-chain token transfers with embedded instructions. A DAO governs upgrades to the shared McA contract code, ensuring secure, consistent updates across all instances. The system also supports gas abstraction, allowing users to pay transaction fees in tokens such as ETH, USDC, or BTC instead of NEAR.

Core functionalities include user onboarding through cross-chain deposits, wallet management, and execution of signed business messages for operations such as token transfers, swaps, and lending interactions. The protocol enables complex multichain use cases like cross-chain swaps and lending on NEAR (e.g., through Burrow) in a single, user-transparent flow. Security mechanisms include signature verification, nonce and expiry management, DAO-controlled upgrades, and the use of pre-approved "reliable" business aliases for safe execution without explicit signatures.

The Zcash Transparent Verifier contract was introduced to validate Zcash v5 transaction signatures while serving as a gateway between the Zcash blockchain and NEAR's implemented Multichain Account protocol. It acts as a proof-of-ownership validator and introduces comprehensive support for Zcash wallet integration into the Multichain Account (MCA) protocol. This enhancement enables users to create MCAs using Zcash transparent transaction signatures, execute business operations signed with Zcash wallets, and add and manage Zcash wallets to existing MCAs.

Repository Coverage and Quality

Code	Tests	Documentation
9 / 10	6 / 10	9 / 10

Resonance's testing team has assessed the Code, Tests, and Documentation coverage and quality of the system and achieved the following results:

- The code follows development best practices and makes use of known patterns, standard libraries, and language guides. It is easily readable and uses the latest stable version of relevant components. Overall, **code quality is excellent.**
- Unit tests are included. The tests cover technical requirements. Code coverage is undetermined. Overall, **tests coverage and quality is average**.
- The documentation includes the specification of the system, technical details for the code, relevant explanations of workflows and interactions. Overall, **documentation coverage and quality is excellent**.

© 2025 Resonance Security, Inc

Target

The objective of this project is to conduct a comprehensive review and security analysis of the smart contracts that are contained within the specified repository.

The following items are included as targets of the security assessment:

- Repository: `ref-finance/multichain-account/contracts`
- Hash: d3e2ff51e78dd29bd84d89dcd8e302dac45678b4

The following items are excluded:

- External and standard libraries
- Files pertaining to the deployment process
- Financial related attacks

© 2025 Resonance Security, Inc

Methodology

In the context of security audits, Resonance's primary objective is to portray the workflow of a real-world cyber attack against an entity or organization, and document in a report the findings, vulnerabilities, and techniques used by malicious actors. While several approaches can be taken into consideration during the assessment, Resonance's core value comes from the ability to correlate automated and manual analysis of system components and reach a comprehensive understanding and awareness with the customer on security-related issues.

Resonance implements several and extensive verifications based off industry's standards, such as, identification and exploitation of security vulnerabilities both public and proprietary, static and dynamic testing of relevant workflows, adherence and knowledge of security best practices, assurance of system specifications and requirements, and more. Resonance's approach is therefore consistent, credible and essential, for customers to maintain a low degree of risk exposure.

Ultimately, product owners are able to analyze the audit from the perspective of a malicious actor and distinguish where, how, and why security gaps exist in their assets, and mitigate them in a timely fashion.

Source Code Review - Rust NEAR

During source code reviews for Web3 assets, Resonance includes a specific methodology that better attempts to effectively test the system in check:

1. Review specifications, documentation, and functionalities
2. Assert functionalities work as intended and specified
3. Deploy system in test environment and execute deployment processes and tests
4. Perform automated code review with public and proprietary tools
5. Perform manual code review with several experienced engineers
6. Attempt to discover and exploit security-related findings
7. Examine code quality and adherence to development and security best practices
8. Specify concise recommendations and action items
9. Revise mitigating efforts and validate the security of the system

Additionally and specifically for Rust NEAR audits, the following attack scenarios and tests are recreated by Resonance to guarantee the most thorough coverage of the codebase:

- Race conditions caused by asynchronous cross-contract calls
- Frontrunning attacks
- Storage staking
- Potentially problematic storage layout patterns
- Manual state rollbacks in callbacks
- Access control issues

© 2025 Resonance Security, Inc

- Denial of service
- Inaccurate business logic implementations
- Unoptimized Gas usage
- Arithmetic issues
- Client code interfacing

Severity Rating

Security findings identified by Resonance are rated based on a Severity Rating which is, in turn, calculated off the **impact** and **likelihood** of a related security incident taking place. This rating provides a way to capture the principal characteristics of a finding in these two categories and produce a score reflecting its severity. The score can then be translated into a qualitative representation to help customers properly assess and prioritize their vulnerability management processes.

The **impact** of a finding can be categorized in the following levels:

1. Weak **-** Inconsequential or minimal damage or loss
2. Medium **-** Temporary or partial damage or loss
3. Strong **-** Significant or unrecoverable damage or loss

The **likelihood** of a finding can be categorized in the following levels:

1. Unlikely **-** Requires substantial knowledge or effort or uncontrollable conditions
2. Likely **-** Requires technical knowledge or no special conditions
3. Very Likely **-** Requires trivial knowledge or effort or no conditions

		Likelihood		
		Very Likely	Likely	Unlikely
Impact	Strong	Critical	High	Medium
	Medium	High	Medium	Low
	Weak	Medium	Low	Info

© 2025 Resonance Security, Inc

Repository Coverage and Quality Rating

The assessment of Code, Tests, and Documentation coverage and quality is one of many goals of Resonance to maintain a high-level of accountability and excellence in building the Web3 industry. In Resonance it is believed to be paramount that builders start off with a good supporting base, not only development-wise, but also with the different security aspects in mind. A product, well thought out and built right from the start, is inherently a more secure product, and has the potential to be a game-changer for Web3's new generation of blockchains, smart contracts, and dApps.

Accordingly, Resonance implements the evaluation of the code, the tests, and the documentation on a score **from 1 to 10** (1 being the lowest and 10 being the highest) to assess their quality and coverage. In more detail:

- Code should follow development best practices, including usage of known patterns, standard libraries, and language guides. It should be easily readable throughout its structure, completed with relevant comments, and make use of the latest stable version components, which most of the times are naturally more secure.
- Tests should always be included to assess both technical and functional requirements of the system. Unit testing alone does not provide sufficient knowledge about the correct functioning of the code. Integration tests are often where most security issues are found, and should always be included. Furthermore, the tests should cover the entirety of the codebase, making sure no line of code is left unchecked.
- Documentation should provide sufficient knowledge for the users of the system. It is useful for developers and power-users to understand the technical and specification details behind each section of the code, as well as, regular users who need to discern the different functional workflows to interact with the system.

© 2025 Resonance Security, Inc

Findings

During the security audit, several findings were identified to possess a certain degree of security-related weaknesses. These findings, represented by unique IDs, are detailed in this section with relevant information including Severity, Category, Status, Code Section, Description, and Recommendation. Further extensive information may be included in corresponding appendices should it be required.

An overview of all the identified findings is outlined in the table below, where they are sorted by Severity and include a **Remediation Priority** metric asserted by Resonance's Testing Team. This metric characterizes findings as follows:

- **"Quick Win"** Requires little work for a high impact on risk reduction.
- **"Standard Fix"** Requires an average amount of work to fully reduce the risk.
- **"Heavy Project"** Requires extensive work for a low impact on risk reduction.

RES-01	Zcash Verification And Execution Dependent On Relayer Implementation	Heavy Project	Acknowledged
RES-02	Missing Error Handling During Zcash MCA Creation	Standard Fix	Resolved

Zcash Verification And Execution Dependent On Relayer Implementation

Info **RES-RHEA-ZTV01** Architecture **Acknowledged**

Code Section

- zcash_transparent_verifier/src/lib.rs#L131
- zcash_transparent_verifier/src/lib.rs#L187
- zcash_transparent_verifier/src/lib.rs#L241

Description

The Zcash Transparent Validator is implemented to verify transactions passed on to the functions `verify_mca_creation()`, `verify_business()`, and `verify_add_zcash()`. The version 5 transaction is parsed and validated of its fields, where the signature present in the field `scriptSig` is verified against the `pubKey`. The current implementation only supports the most common transaction script type, Pay-to-Public Key Hash (P2PKH). Along with the parameter `target_address`, it is possible to validate that the owner of a Zcash wallet successfully deposited to a pre-calculated deposit address.

The correct and legitimate execution of these functions directly depend on the implementation and actions of the relayer. This effectively means the Zcash verifier only verifies the parameters that are passed in as inputs, making no assumptions of their validity. As such, it may be possible for malicious or compromised relayers to pass in parameters that will result in successful outcomes for these functions.

A few examples include:

- A malicious or compromised relayer may use any private key to forge and sign a transaction without broadcasting it to the network. Since there is no way, on-chain, to verify that said transaction actually exists and was executed, it is possible for the signature of the transaction passed in as an input to still be verified against the public key and the Zcash relevant functions still be executed on the MCA protocol.

- A malicious or compromised relayer may perform a legitimate transaction to the pre-calculated `target_address`, but with a transfer value of 0. While a legitimate transaction now exists and is capable of proving ownership of a certain Zcash wallet, the transaction would still be validated by the verifier and Zcash relevant functions would still be executed on the MCA protocol, while no funds would be transferred to Rhea Finance's vault. E.g., in the function `verify_mca_creation()` no equivalent of a `create_mca_fee` would be deposited and serve as a fee for MCA creation, therefore making the protocol lose money on inexistent fees.

- A malicious or compromised relayer may create numerous MCAs with 0 deposit to the pre-calculated `target_address` and then transfer the initial deposited 0.2N (`mca_init_balance` + `EXTRA_NEAR_AMOUNT`) to NEAR accounts of their own.

It should be noted that the severity of this finding has been reduced due to the fact that the Relayer role, caller of these functions, is controlled by Rhea Finance.

© 2025 Resonance Security, Inc

Recommendation

It is recommended to ensure that the Relayer implementation, being an off-chain component:

- Properly generates and validates legitimate transactions, and that these are properly finalized on the network with a positive deposit value.
- The deposit value into the pre-calculated deposit address should at least cover the initial deposit made during the creation of MCAs (`mca_init_balance` + `EXTRA_NEAR_AMOUNT`).

Additionally, it is recommended to ensure that the account with the Relayer role is a multi-signature wallet to further decrease the chance of being compromised.

Status

The issue was acknowledged by Rhea Finance's team.

Missing Error Handling During Zcash MCA Creation

Info **RES-RHEA-ZTV02** Code Quality **Resolved**

Code Section

- `account_manager/src/interfaces/zcash.rs#L69-L79`

Description

The callback function `create_mca_from_zcash_callback()` does not properly validate the result of the `create_account()` promise, leaving the state variables intact should an error occur. This fact hinders further MCAs from being created with the same Zcash wallet address.

It should be noted that the severity of this finding has been reduced due to the fact that the Relayer role, caller of this function, is controlled by Rhea Finance.

Recommendation

It is recommended to implement error handling code to rollback any previous changes made do the contract's state, in case the promise results in an error.

Status

The issue has been fixed in 24e47cd462edc9a779f59ba459201bbc3e804e1d.

© 2025 Resonance Security, Inc

Proof of Concepts

No Proof-of-Concept was deemed relevant to describe findings in this engagement.

© 2025 Resonance Security, Inc